CS7641 A3 – Unsupervised Learning and Dimensionality Reduction

Lirui Xiao (lxiao75)

Overleaf link: https://www.overleaf.com/read/xbdxkycmpmnm271fdb

1 DATASET DESCRIPTION

1.1 Dataset 1

Dataset 1 is the result of a chemical analysis of breast cancer patients, sourced from OpenML.org datasets. The analysis determined the quantities of 30 continuous features related to breast cancer. The classification task is binary, predicting whether a patient has breast cancer. The original dataset has 569 entries. Using the features, I previously tried to apply algorithms to predict the diagnosis of each patient. The dataset is reasonably balanced, with no class dominating the other.

This dataset presents four significant features. First, it is a binary classification problem, focusing on whether a patient has breast cancer or not. Second, the number of features is substantial, with 30 different attributes analyzed for each patient. Third, the dataset is well-balanced, ensuring that the model training and evaluation processes are fair and unbiased. For the hyper-parameter optimization process, I used the weighted F1 score as the evaluation metric. The weighted F1 score calculates the F1 score for all classes and then takes their weighted average according to their data count, providing a comprehensive measure of model performance.

1.2 Dataset 2

Dataset 2 is the one I experimented with in Assignment 1. It conducted a market survey to collect the prices of many mobiles, which helps to price its newly produced mobile phones. This dataset comprises 2001 entries, and each details various features of the mobile phones, such as RAM, memory, battery power, and Front Camera mega pixels. It is a multi-class problem, as the phone's price range, the data target, is categorized from 0 to 3 (0 represents the lowest price range and 3 represents the highest). Also, the number of features is 20, which is substantial. I guess it might be quite helpful to see the influence of dimensionality reduction in supervised learning. For the hyper-parameter optimization process, I also used the weighted F1 score as the evaluation metric, just what I have mentioned before in the Dataset 1.

2 HYPOTHESIS

In terms of clustering methods, I hypothesize that the Expectation Maximization (EM) algorithm with 'tied' and 'full' covariance types will better capture the complex data structures in the Breast Cancer dataset and provide superior clustering results. The EM algorithm's strength lies in its ability to model the data distribution more flexibly, particularly when the data has varied shapes and orientations. 'Tied' covariance allows for the clusters to share the same shape but differ in size, while 'full' covariance can adapt to any cluster shape, making it versatile for complex datasets. For the Mobile Price dataset, I hypothesize that the K-Means algorithm will provide better clustering results with an appropriate number of clusters (k). K-Means is effective in partitioning data into well-separated groups, and determining the optimal k by maximizing the Silhouette score and Calinski-Harabasz index ensures the balance between intra-cluster cohesion and inter-cluster separation.

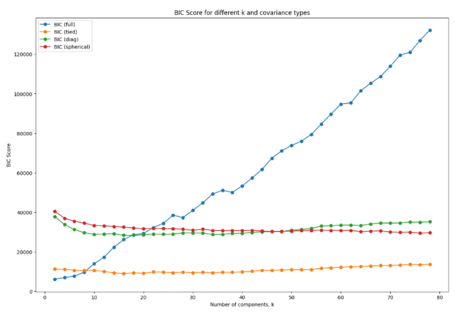
In terms of dimensionality reduction impacts on clustering performance, I hypothesize that the application of PCA in the Breast Cancer dataset will lead to significantly lower BIC scores when using the EM algorithm. This is because PCA reduces the number of dimensions while retaining the most critical information, thereby simplifying the data structure and improving model fit. For the Mobile Price dataset, I anticipate that using Random Projection (RP) for dimensionality reduction will better preserve the data structure, resulting in higher Silhouette scores when applying the K-Means algorithm. RP maintains the distance relationships in the data, which is crucial for effective clustering, ensuring that the essential patterns and structures within the data are preserved even after dimensionality reduction.

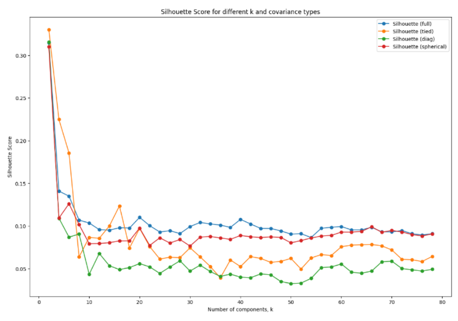
3 CLUSTERING ALGORITHMS

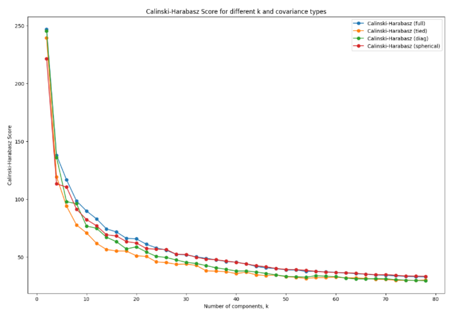
3.1 Expectation Maximization (EM)

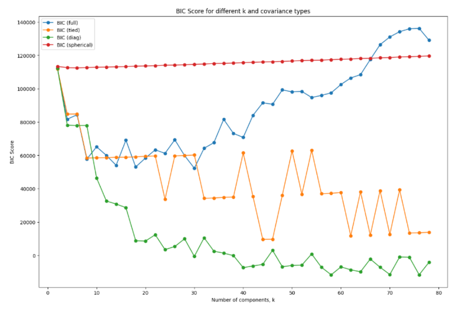
The Expectation-Maximization (EM) algorithm is a powerful tool used for finding maximum likelihood estimates in the presence of latent variables. It involves two key steps: the Expectation step (E-step) and the Maximization step (M-step). During the E-step, the algorithm estimates the expected value of the latent variables given the observed data and current parameter estimates. In the M-step, these expected values are then used to compute an updated estimate of the model parameters that maximizes the expected log-likelihood.

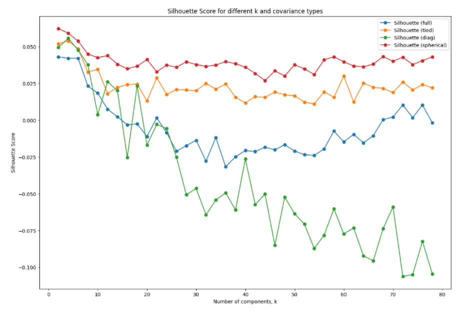
For my implementation, I used GaussianMixture to fit a mixture of Gaussian distributions to the data. This approach models the data as a combination of several Gaussian clusters, with the algorithm iteratively refining the parameters of these clusters. To evaluate the performance of the models, which also means to prove they make sense, I employed the Bayesian Information Criterion (BIC). BIC is a metric that balances the model's likelihood against its complexity. It penalized models with more parameters to avoid overfitting. The model with the lowest BIC score is considered the best, as it provides a good fit with minimal complexity. Also, to control other variables, I experimented with four covariance constraints (spherical, diagonal, tied, and full).

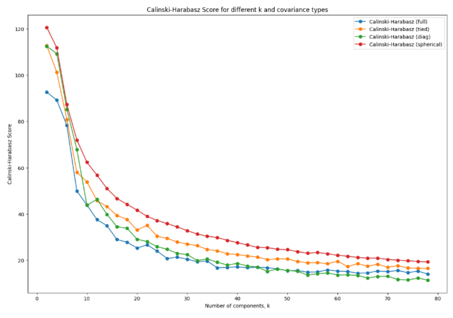












I presented the results of the EM algorithm for both datasets. The 'full' covariance type of the Breast Cancer dataset has a rapid increase in scores as the number of components increases, while the 'diag' and 'spherical' types have relatively low and smooth scores, suggesting that they are more effective in dealing with complex data structures. The Mobile Price dataset has its lowest BIC at n = 30, with a BIC of approximately 55000 and covariance type of "diag". For the first dataset, both the silhouette and Calinski-Harabasz scores are highest at k = 2, indicating this as the optimal number of clusters, which is not consistent with the Distortion score. This inconsistency may be due to the Distortion score's sensitivity to intra-cluster variance, which can be misleading if clusters are not well-separated. As k increases, these scores drop and stabilize, suggesting that additional clusters do not significantly improve clustering quality. For the second dataset, the scores also peak at k = 2 but then fluctuate more, indicating a more complex clustering structure. This variability suggests that while k = 2 may be optimal initially, the underlying data may have more nuanced patterns requiring further investigation. Overall, the Breast Cancer dataset shows better clustering stability with increasing k, whereas the Mobile Price dataset indicates more complexity and potential noise affecting the clustering quality.

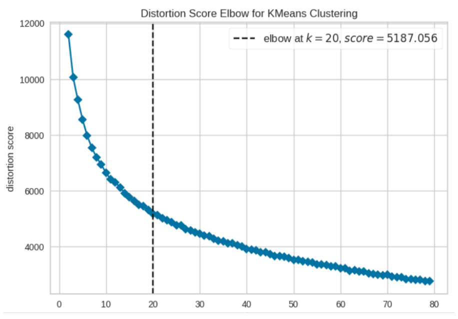
3.2 K-Means

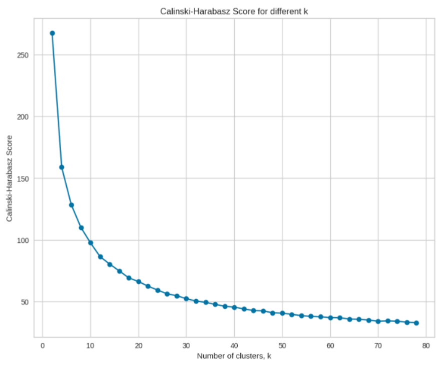
K-Means is a heuristic clustering algorithm that works by minimizing the error, defined as the distance from a point within a cluster to the cluster's center. Initially, clusters are assigned, and the algorithm iteratively adjusts the centers of the clusters and reassigns points to reduce the error. I implemented this using K-Means and KElbow Visualizer.

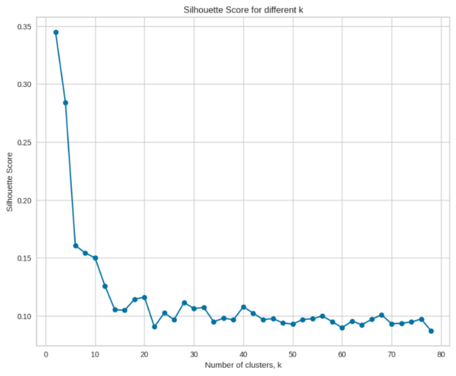
I evaluated the performance of my K-Means algorithm using several metrics. One key metric is distortion, which represents the sum of squared errors that K-Means aims to minimize. This score typically decreases as the number of clusters (k) increases. To interpret this, I look for the "elbow" in the graph, where the rate of decrease slows down. This point, marked by the maximum curvature on my distortion graphs, suggests the number of clusters that most effectively capture the data. Another important metric is the silhouette score. This score considers both intra-cluster and inter-cluster distances. It reaches its peak at 1.0 when points are close to their cluster mates and far from points in other clusters. Finally, I used the Calinski-Harabasz index, which evaluates the variance between and within clusters. The higher the score, the better the clusters are defined.

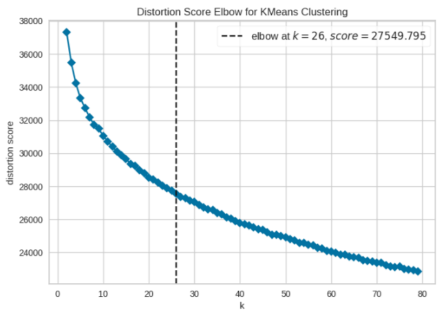
The figures above show the silhouette scores across different values of k for these two datasets. For the first dataset (Figure 1), the silhouette score is highest at k = 2 but then gradually decreases. This indicates that while increasing the number of clusters, the quality of clustering does not significantly improve. Notably, there are small plateaus at k = 19, k = 22, and k = 36. These mini-plateaus are worth investigating because typically, adding clusters should either increase the clustering quality until the optimal number or decrease, but not flatline. For the second dataset, the silhouette score is also highest at k = 2, then drops sharply and gradually rises with higher values of k. This phenomenon may suggest that the clustering structure of this dataset is more complex, and a larger number of clusters might better capture the data's intrinsic structure.

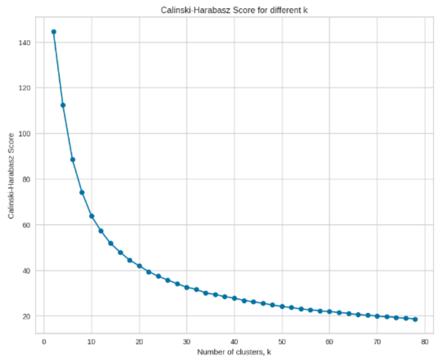
Overall, the first dataset shows a clear optimal point at k = 2, while the second dataset exhibits more complex clustering behavior, which is reasonable since it has multiple class to identify. The silhouette scores indicate that different datasets perform differently across various k values, reflecting the inherent complexity and diversity of the data itself. And it is nice to see that the results are broadly consistent with EM's. But the F1 scores are quite low (below 0.6), so **I decide to try dimension reduction to improve the performance in my following analysis**.

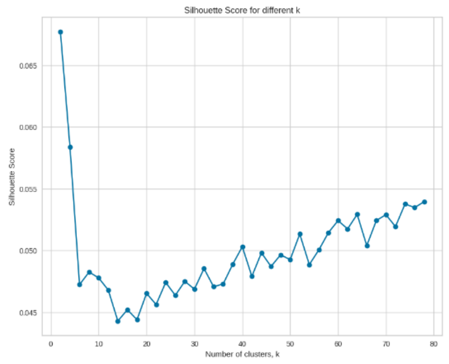












K-Means is a heuristic clustering algorithm that works by minimizing the error, defined as the distance from a point within a cluster to the cluster's center. Initially, clusters are assigned, and the algorithm iteratively adjusts the centers of the clusters and reassigns points to reduce the error. I implemented this using K-Means and KElbow Visualizer.

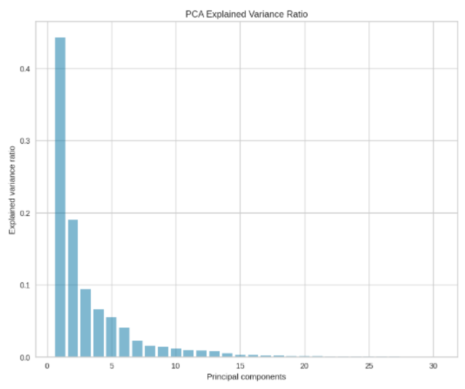
I evaluated the performance of my K-Means algorithm using several metrics. One key metric is distortion, which represents the sum of squared errors that K-Means aims to minimize. This score typically decreases as the number of clusters (k) increases. To interpret this, I look for the "elbow" in the graph, where the rate of decrease slows down. This point, marked by the maximum curvature on my distortion graphs, suggests the number of clusters that most effectively capture the data. Another important metric is the silhouette score. This score considers both intra-cluster and inter-cluster distances. It reaches its peak at 1.0 when points are close to their cluster mates and far from points in other clusters. Finally, I used the Calinski-Harabasz index, which evaluates the variance between and within clusters. The higher the score, the better the clusters are defined.

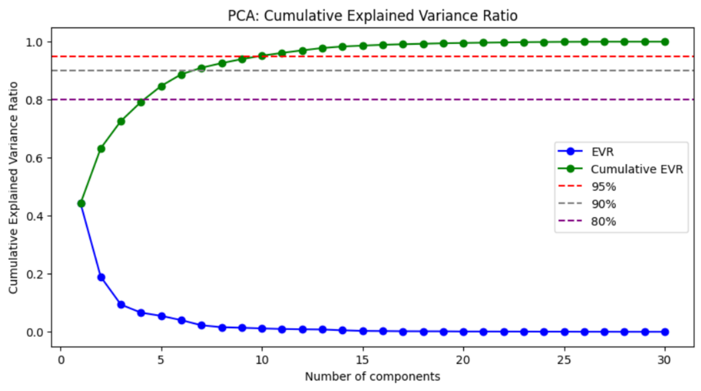
The figures above show the silhouette scores across different values of k for these two datasets. For the first dataset (Figure 1), the silhouette score is highest at k = 2 but then gradually decreases. This indicates that while increasing the number of clusters, the quality of clustering does not significantly improve. Notably, there are small plateaus at k = 19, k = 22, and k = 36. These mini-plateaus are worth investigating because typically, adding clusters should either increase the clustering quality until the optimal number or decrease, but not flatline. For the second dataset, the silhouette score is also highest at k = 2, then drops sharply and gradually rises with higher values of k. This phenomenon may suggest that the clustering structure of this dataset is more complex, and a larger number of clusters might better capture the data's intrinsic structure.

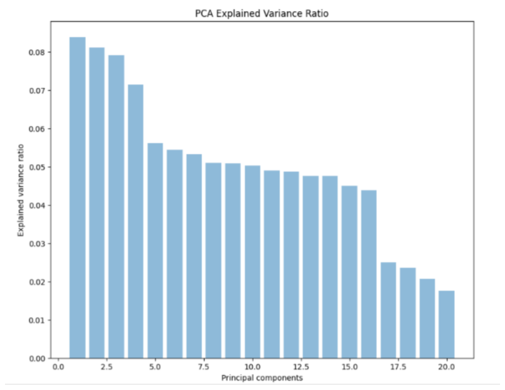
Overall, the first dataset shows a clear optimal point at k = 2, while the second dataset exhibits more complex clustering behavior, which is reasonable since it has multiple class to identify. The silhouette scores indicate that different datasets perform differently across various k values, reflecting the inherent complexity and diversity of the data itself. And it is nice to see that the results are broadly consistent with EM's. But the F1 scores are quite low (below 0.6), so **I decide to try dimension reduction to improve the performance in my following analysis**.

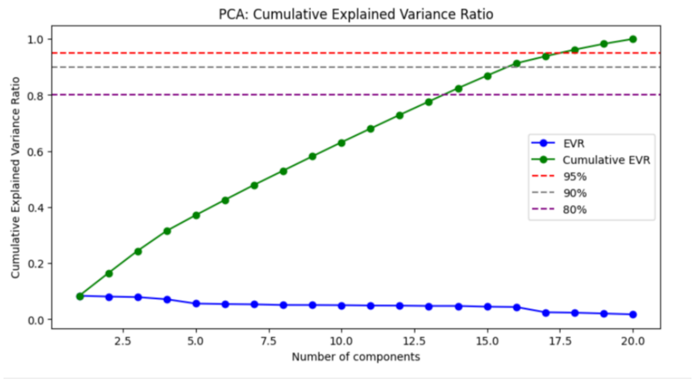
4 DIMENSIONALITY REDUCTION

4.1 Principal Component Analysis (PCA)









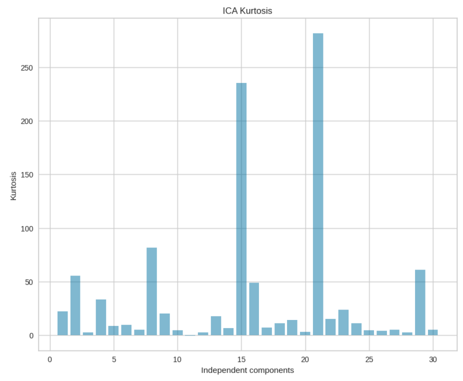
Principal Component Analysis (PCA) is a widely used dimensionality reduction technique that aims to reproject data points onto a new basis that maximizes the variance. In this new basis, each principal component is orthogonal to the others. The first principal component captures the highest variance in the data, while the last principal component captures the least variance. I implemented PCA using scikit-learn's PCA module. The process involves computing the covariance matrix of the data, followed by performing an eigenvalue decomposition to find the new orthogonal basis vectors, known as principal components. These principal components are ranked according to the amount of variance they explain. By selecting the top principal components, I can represent the original data with fewer dimensions while preserving most of its variability.

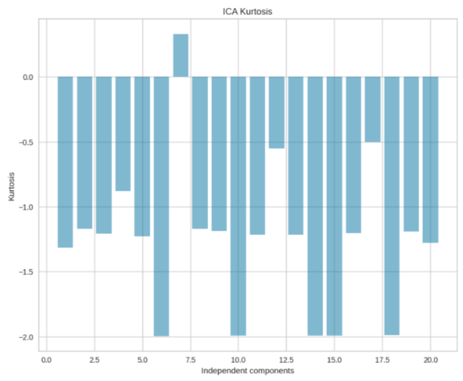
I displayed the explained variance ratio (EVR) and the cumulative explained variance ratio below. These plots determine the number of principal components that are needed to retain most of the information in the data, which allows me to effectively reduce the data's dimensionality to a manageable level while maintaining its essential characteristics. For the first dataset, it shows that around 10 components are needed to retain 95% of the variance, and about 8 components are needed to capture 90% of the variance. We can also observe the distribution of it. The first principal component explains the largest amount of variance, about 40% or more of the total variance. In contrast, the second dataset requires more components to capture the same amount of variance. Approximately 18 components are needed to retain 95% of the variance, and around 15 components are needed to capture 90% of the variance. This indicates that the second dataset has more intrinsic variability and complexity, requiring more principal components to adequately summarize the data. But this dataset has relatively average principal component values, with no particularly large components.

4.2 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a computational technique used to separate a multivariate signal into additive, independent components. This method assumes that the observed data are linear mixtures of independent sources, and its goal is to identify these sources. I implemented ICA using scikit-learn's FastICA.

I use the kurtosis of the points projected into the new space to measure the independence of the separated components. Kurtosis is the fourth moment of the data distribution, which indicates how heavy or light the tails are compared to a normal distribution. Higher kurtosis values suggest longer tails and the presence of outliers, signifying non-Gaussian components. This property is useful because ICA relies on the non-Gaussianity of the data to achieve separation.





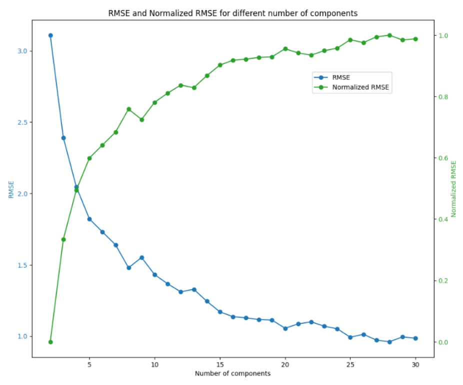
In my analysis, I presented these kurtosis values. A kurtosis value of 0 corresponds to a Pearson kurtosis of 3, which is the kurtosis of a standard Gaussian distribution. Therefore, any value above 0 indicates longer tails compared to the normal distribution, highlighting the presence of non-Gaussian components which ICA aims to isolate. The first graph shows the kurtosis values of the independent components, some of which are very high, especially the 15th and 20th independent components. Thus, the ICA projection axis in the first dataset captures meaningful information, especially on the independent components with these two high kurtosis values. The second graph shows independent components with mostly negative kurtosis values, and all the values are small. This indicates that these components in the second dataset do not have significant non-Gaussian properties and may not contain important feature information. When comparing the results of the two datasets, a noticeable discrepancy arises in the kurtosis values of the independent components. In the first dataset, several components exhibit significantly high positive kurtosis values, and conversely, the second dataset shows mostly negative kurtosis values across the components. This inconsistency may stem from differences in the underlying distributions of the datasets, preprocessing steps, or noise levels, leading to variations in the separation of non-Gaussian components by ICA.

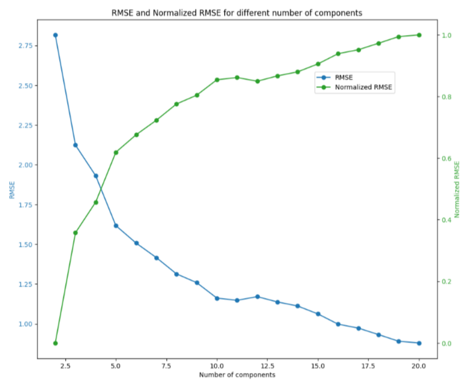
4.3 Randomized Projection (RP)

Random Projection (RP) involves projecting the data onto a randomly chosen lower-dimensional subspace. The idea is to preserve the structure and dependencies of the original data in the new projection as much as possible. To evaluate the effectiveness of RP, I calculated the reprojection error by projecting the data back from the reduced subspace to the original space and measuring the deviation of the points from their original values. Naturally, if all components are retained, the reprojection error would be zero. However, reducing the number of components results in information loss and introduces error. In my implementation, I used scikit-learn's GaussianRandomProjection, and the error is measured as the root-mean-squared error (RMSE), which remains non-zero even when n = 30 (all features are used).

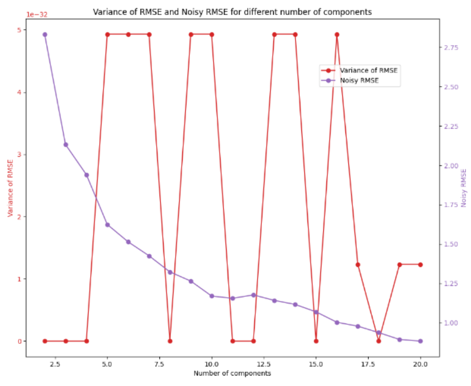
To further interpret the results, I used normalized RMSE, which is the percentage difference between each component's RMSE compared to the maximum and minimum RMSE values. In this scale, the maximum RMSE corresponds to a normalized value of 0.0, while the minimum RMSE corresponds to a normalized value of 1.0, providing a clear view of how the error changes with different dimensionality reductions.

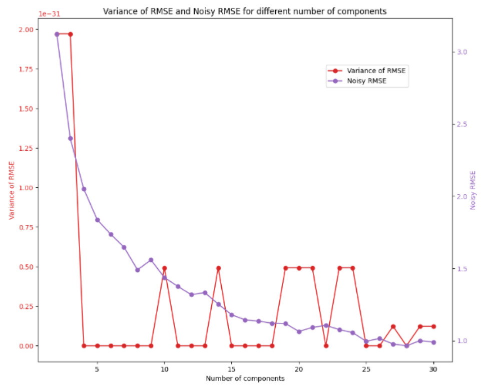
When comparing the two datasets' results, a noticeable difference can be observed. In the first dataset, the RMSE decreases sharply as the number of components increases. Conversely, in the second dataset, although the trend is similar, the RMSE starts at a slightly lower value and decreases more gradually, with the normalized RMSE reflecting a slower rate of change. This inconsistency might be due to variations in the intrinsic dimensionality and distribution of the datasets. Also, I guess the nature of the features, or potential preprocessing differences will also affect the result and how well the random projection preserves the original data structure.

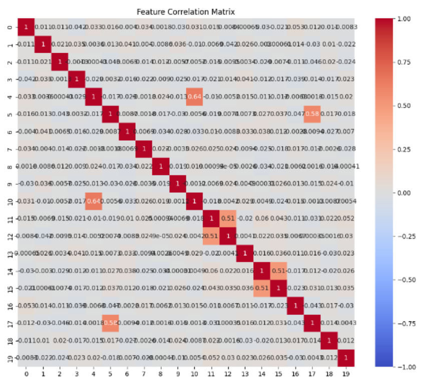


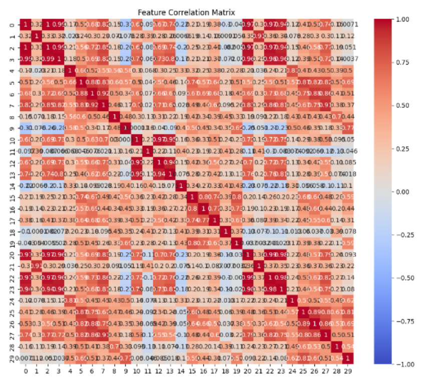


The data rank for the first dataset is 30, and the second dataset is 20, which are the same number as their features. From the Noisy RMSE curves, they decrease as the projection dimension k increases. This indicates that even with noise added to the data, the algorithm is still able to reconstruct the data better as the projection dimension increases. However, the value of RMSE with noise is always higher than the value of RMSE without noise, which indicates that noise has a negative impact on the reconstruction results. Noise makes the algorithm perform worse in low-dimensional space, but in high-dimensional space, the effect of noise is relatively reduced. As for the Feature Correlation Matrix, it shows the correlation between features from the dataset is very low, while the correlations in dataset 2 is larger with more complex correlations, which suggests that there is a covariance problem in the data. So, the RP algorithms is not suitable to this dataset.



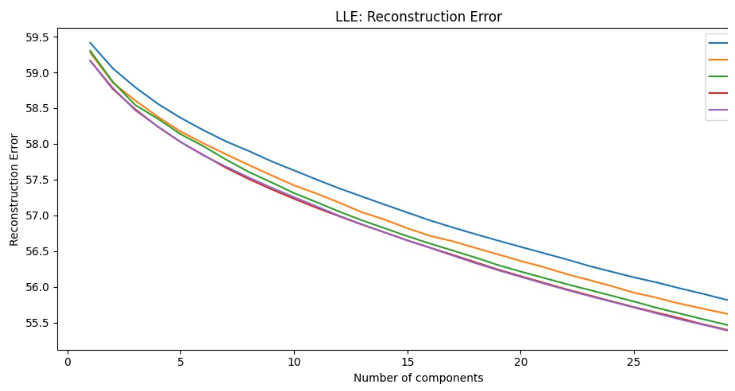


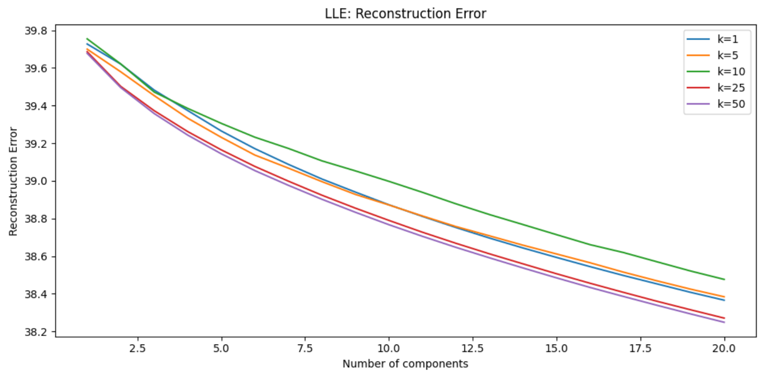




4.4 Locally Linear Embedding (LLE)- Extra Credit

Locally Linear Embedding (LLE) is a Non-linear dimensionality reduction technique that aims to preserve the local linear relationships between points in a high-dimensional space when mapping them to a lower-dimensional space. The process starts by constructing a graph where each point is connected to its k nearest neighbors. LLE then finds a lower-dimensional embedding by minimizing the reconstruction error, which seeks to maintain these local linear relationships as faithfully as possible.

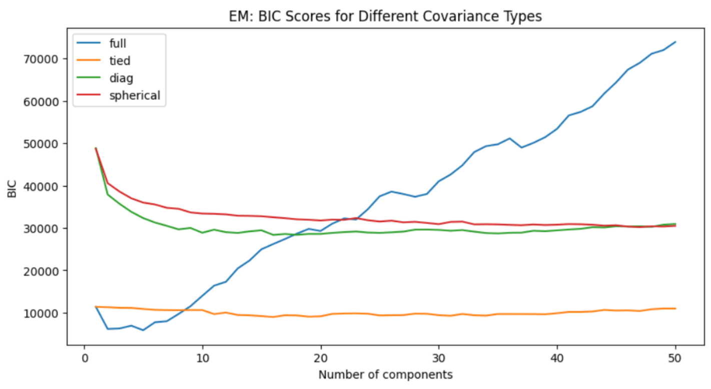


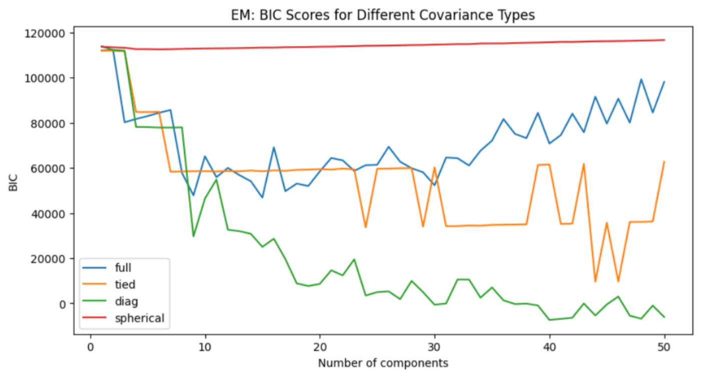


I chose Locally Linear Embedding (LLE) for dimensionality reduction due to its ability to preserve local linear relationships within the data, making it well-suited for datasets where local structure is more informative than global structure. LLE constructs a neighborhood graph and seeks a low-dimensional embedding that minimizes reconstruction error, providing a meaningful representation of high-dimensional data in fewer dimensions. In the first dataset, the reconstruction error decreases as the number of components increases for all values of k. Lower values of k generally result in higher reconstruction errors, suggesting that very local structures might not be sufficient to capture the overall data structure effectively. In the second dataset, the reconstruction error also decreases with an increase in the number of components for all values of k, but the impact of k is more pronounced, indicating that the choice of k significantly affects the quality of the embedding.

5 DIMENSIONALITY REDUCTION + CLUSTERING

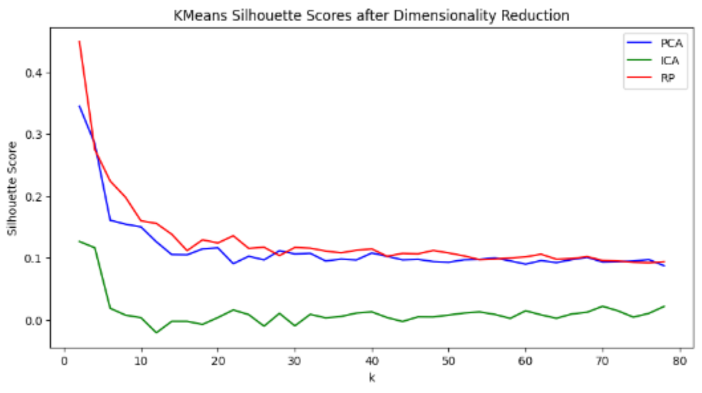
5.1 Expectation Maximization (EM)

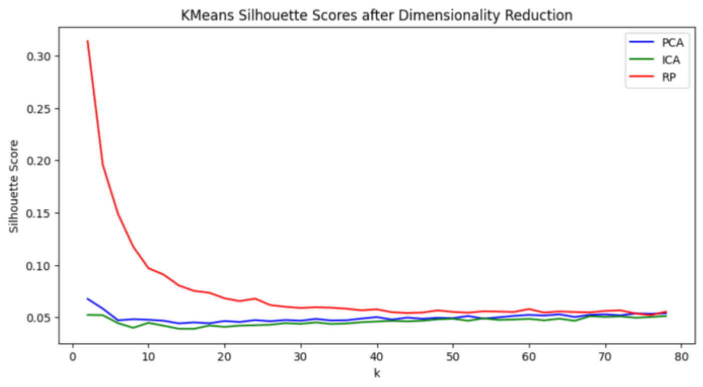




For Breast Cancer Dataset, it shows that the overall BIC scores are reduced after dimensionality reduction, especially the scores of 'tied' covariance types are significantly reduced and even negative. This indicates that the model complexity and fitting effect are improved after dimensionality reduction.' full' covariance type still exhibits high BIC scores after dimensionality reduction, but the growth trend has moderated.' diag' and 'spherical' types show little change in BIC scores, indicating that they are less sensitive to dimensionality reduction. Overall, stochastic projection dimensionality reduction effectively reduces model complexity and improves the fit of certain covariance types. For Mobile Price Dataset, after applying random projection (RP) dimensionality reduction, the BIC scores of the EM algorithm are significantly reduced, especially the scores of 'tied' and 'full' covariance types are substantially reduced and more stable. Whereas, before dimensionality reduction, the BIC scores of 'diag' and 'spherical' covariance types are higher and less variable. After dimensionality reduction, the BIC scores of all covariance types tend to be negative, indicating that the model complexity is reduced significantly and the fitting effect is improved.

5.2 K-Means





For K-means, I use Silhouette scores as my examples. And I observed interesting differences in the silhouette scores after applying dimensionality reduction techniques. Compared to the techniques without dimension reduction, they both improved their scores (from 0.35 to 0.45; from 0.07 to 0.33). The first figure displays the silhouette scores for the breast cancer dataset. In the RP plot, the silhouette score starts high and declines sharply with increasing k, stabilizing around k=30. For ICA, the silhouette scores remain relatively low across different values of k, indicating that the clustering is less effective after ICA. The baseline clustering shows moderate performance compared to RP and ICA. In the mobile price dataset, the RP plot again starts with a high silhouette score at k=2 and declines steeply, stabilizing around k=20. This shows that RP has superiority in this specific dataset. The baseline and ICA results are similar, showing lower silhouette scores overall, with ICA having the lowest scores across most values of k. This suggests that RP may be better at preserving the clustering structure for the mobile price dataset compared to ICA. I guess the differences in performance between RP and ICA across these datasets could be due to how each method handles the underlying structure. RP is a linear transformation that preserves distances, which might be beneficial for the mobile price dataset. On the other hand, ICA aims to find statistically independent components, which might not align well with the clustering structure, leading to lower silhouette scores.

Here I test whether the clusters line naturally with my labels using F1 scores, and I saw they are all over 0.9, suggesting quite good results compared to the simple clustering algorithms (only about 0.6). I guess this is because the original dataset has much noise, and dimensionality reduction techniques simplify the data structure, remove noise and redundant features, thus allowing clustering algorithms to run on purer and more relevant data, improving clustering results.

6 DIMENSIONALITY REDUCTION + NEURAL NETWORK

Algorithm	Train F1	Test F1	Prep Time	Train Time	Adj Train Time	Train Iters	Test Time
Base	1.000000	0.973621	0.013133	2.564694	2.577827	234	0.002842
PCA	1.000000	0.964912	0.028800	1.734711	1.763511	200	0.000338
ICA	0.988976	0.964738	0.158563	2.555681	2.714244	602	0.000308
RP	0.995599	0.982369	0.001451	1.201965	1.203416	332	0.000317

6.1 Baseline Neural Network

The neural network I utilized as a baseline was fine-tuned in stages, starting broadly and becoming more precise, ultimately configured with 100 hidden layers and running for 1000 iterations. This serves as a reference point for evaluating the dimensionality reduction algorithms. Although it exhibits some overfitting due to the lack of bias and presence of variance, it still achieves a relatively high performance with a final F1 score of 0.979.

6.2 Dimension Reduction

Based on the provided table, RP achieved the highest test F1 score of 0.982 with the lowest prep time of 0.0015, showing its efficiency in both preparation and performance. PCA also performed well with a test F1 score of 0.965 and a relatively low prep time of 0.028800, indicating its effectiveness in capturing variance and relationships in the data. However, ICA, despite its non-Gaussian information capturing capability, had a slightly lower test F1 score of 0.964738 and the longest prep time of 0.158563, suggesting it might not be as suited for Gaussian-distributed data. The base model had a high test F1 score of 0.973621, but the highest adjusted train time, implying a trade-off between training time and performance. Overall, RP and PCA show strong performance with efficient preparation times, making them suitable for quick and effective dimensionality reduction.

7 CLUSTERING + NEURAL NETWORK

Algorithm	Train F1	Test F1	Prep Time	Train Time	Adj Train Time	Train Iters	Test Time
Base	1.0	0.973621	0.008242	0.921838	0.930079	234	0.000353
KMeans	1.0	0.964912	0.331261	0.899430	1.230692	194	0.000421
EM	1.0	0.973621	0.241323	1.002368	1.243691	198	0.000397

I used the Breast Cancer Dataset for my experiment. I generated cluster centers using just the training data and then one-hot encoded the clusters as additional features to augment the existing ones, hoping that this extra information would help the network. For K-Means, I used 41 clusters based on my analysis. For EM, I used 40 clusters. I was initially concerned that these extra 40 features on top of the 30 existing ones would cause the training times to increase drastically, but they did not. However, K-Means performed worse than the base network in both F1 score and preparation time. EM performed almost the same as the base algorithm in terms of F1 score but took much longer to prepare and train. I think it is because the extra cluster information didn't add any useful signal and instead injected noise, as evidenced by the longer training time and iterations. I'm not sure why this is the case compared to K-Means, as I would expect them to add similar levels of extra information. Perhaps the difference in the number of clusters contributed to this outcome, with the existing clusters in K-Means providing just enough additional information to be detrimental. However, more datasets would help further explore and validate these findings.

8 CONCLUSIONS

In this experiment, I used the Breast Cancer Dataset and the Mobile Price Dataset to explore clustering and dimensionality reduction. The results showed that 'full' covariance type had high BIC scores, while 'diag' and 'spherical' were more stable and effective in dealing with complex data structures. For K-Means clustering, I observed that increasing the number of clusters initially improved the silhouette scores, but the quality plateaued or even declined with further increases. Principal Component Analysis (PCA) and Independent Component Analysis (ICA) were used for dimensionality reduction. PCA effectively reduced the data dimensionality while retaining most of the variance, with the first dataset requiring fewer components than the second. ICA identified non-Gaussian components, but the presence of noise affected its performance. Random Projection (RP) demonstrated varying degrees of effectiveness, with the first dataset showing a sharper decrease in RMSE compared to the second. After adding dimension reduction (remove noises), the clustering methods lines more closely with the actual labels by improving F1 scores. In clustering and neural network integration, the base model performed better than K-Means and EM in terms of F1 score and preparation time. K-Means added extra features that did not significantly enhance the network's performance, and EM, while almost matching the base model's F1 score, incurred longer training times.