A3. Unsupervised Learning and Dimensionality Reduction

Jihoon 'Jay' Song
jsong350@gatech.edu

INTRODUCTION

In this section, I briefly re-introduce the two datasets that I will be referencing throughout the paper. I describe the datasets and discuss what makes them interesting from a machine learning perspective. Lastly, I provide the problem statements associated with each dataset.

Dataset #1 – I sourced my first dataset, "Abalone Data Set" from the UCI: Machine Learning Repository. This dataset contained 4176 rows which were described using 8 classes including 'sex', 'length', 'diameter', 'height', 'whole weight', 'shucked weight', 'viscera weight', 'shell weight' and 'rings'. The data type of every class except 'sex' (nominal) and 'rings' (integer) were continuous. This dataset was used to solve a multi-class classification problem with 3 labels: To predict abalones 'sex' (Male, Female, Infant). From here on, I will refer to this dataset as 'dataset #1'.

Dataset #2 – Like the first one, I sourced my second dataset, "Wine Quality Dataset" from the UCI: Machine Learning Repository. This dataset contained 4898 rows which were described using 12 classes including 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', and 'recommend'. The data type of every attribute was continuous except 'recommend' (Boolean). This dataset was used to solve a binary classification problem: To predict if the wine would be recommended (1 for True, 0 for False).

To create the binary classification problem, I modified the widely known Wine Quality Dataset by consolidating the 'quality' (integer between 3 and 9) column in the original version into a binary one which I relabeled 'recommend'. Instances with quality over 6 were reassigned value of 1 (True), and those with quality equal to or less than 6 were reassigned value of 0 (False). From here on, I will refer to this dataset as 'dataset #2'.

PART I

In Part I, I explored two clustering algorithms: *k*-means clustering (KM) and Expectation Maximization (EM). Both algorithms were attained through the scikit-learn library [1][2]. To determine the most optimal *k* values for each algorithm/dataset, I plotted the Silhouette Coefficient (SC) from k=2 to k=100 and found the cut-off point using the elbow method [3].

Once the *k* values were chosen, I assessed the algorithms by evaluating their clustering outputs. For each combination of algorithm/dataset, I calculated and analyzed the Homogeneity Score (HS) and Rand Index (RI). The values of SC, HS, and RI were all plotted on the same space for easy comparison. The purpose was to compare the *k* value suggested by the unsupervised metric, SC against the values suggested by the supervised metric, HS, and RI. The final *k* value for both algorithms, however, were determined using only the value of their SC values to avoid "cheating" by peeking into the true y-labels.

Silhouette Coefficient (SC) – Intuitively, better clustering should result in clusters which have larger inter-cluster distances and higher densities. Therefore, I chose SC, an unsupervised performance metric, since it expresses both the distances between the clusters and their densities [4]. SC has a maximum value of 1 and a minimum value of -1. A value closer to 1 indicates that the clusters are well separated, and highly dense. A value closer to -1 indicates the opposite.

Homogeneity Score (HS) – I chose to analyze the effect of HS because unlike SC, it uses information from the target class distribution as part of its assessment. Intuitively, better clustering should result in clusters which are more homogenous. Using the distribution of labels in the target class and the cluster designations of each sample, it calculates a score between 0 and 1. A value closer to 1 indicates clusters are more homogenous and a value closer to -1 indicates the opposite [5].

Rand Index (RI) – I chose RI because even more so than HS, its evaluation criteria rely on knowing the target class. Intuitively, better clustering should result in clusters which assign instances which have the same true labels grouped into the same cluster. RI indicates this measurement through a score between 0 and 1. A value closer to 1 indicates clusters are more inclusive, and a value closer to 0 indicates the opposite [6].

1.1 Dataset #1 – K-Means Clustering

When performing KM on dataset #1, as shown in *Figure 1*, *k=20* resulted in the highest value of SC, which was 0.566. This value indicates that KM was successful in generating distinct, meaningful clusters. The value indicates the resulting clusters demonstrated high inter-cluster distances and cluster densities. Interestingly, as demonstrated by the red (HS) and green (RS) lines in *Figure 1*, label information had minimal impact on the algorithm's *k* value selection.

As shown in *Table 1*, the HS value was 0.181, indicating the resulting clusters were weakly uniform. The RI value was 0.656, indicating clusters contained instances which have the same true labels. Given HS ranges [-1, 1] and RI ranges [0, 1], both measurements are consistent in their assessment that

the clustering outputs were satisfactory but has significant room for improvement.

1.2 Dataset #1 – Expectation Maximization

When performing EM on dataset #1, as shown in *Figure 2*, *k=16* resulted in the highest value of SC, which was 0.514. Like the results of the KM experiment, this value indicates EM resulted in clusters with high inter-cluster distances and cluster densities. Although the difference in SC value is small, it shows that EM's soft clustering had a slightly negative impact on clustering. In addition, like KM, as shown by the HS and RI values, the usage of label information had minimal impact on the algorithm's *k* value selection.

As shown in *Table 1*, the HS value was 0.146, indicating the resulting clusters were weakly uniform. The RI value was 0.646, indicating more time than not, instances within the same clusters had the same true labels. Like that of KM, both measurements were consistent in their assessment that the clustering outputs were satisfactory but has significant room for improvement.

1.3 Dataset #2 – K-Means Clustering

When performing KM on dataset #2, as shown in *Figure 3*, *k=2* resulted in the highest value of SC, which was 0.506. Like the results of the KM experiment on dataset #1, this value indicates KM resulted in clusters with high inter-cluster distances and cluster densities, signaling that the clustering effort was successful. I found it interesting that neither SC nor HS improved as the number of clusters increased. HS improved slightly but plateaued quickly around *k=15*. This made sense intuitively since this is a binary classification problem, and therefore, the two clusters potentially represented the 0 or 1 for the "recommend" column. Consistent with the previous two experiments, as shown by the HS and RI values, usage of label information had minimal impact on the algorithm's *k* value selection.

As shown in *Table 1*, the HS value was 0.022, indicating the resulting clusters were very weakly uniform. The RI value was 0.520, indicating more times than not, instances within the same clusters had the same true labels. These values made sense intuitively, since there were only 2 clusters, compared to 20 and 16 in the previous experiments using dataset #1.

1.4 Dataset #2 – Expectation Maximization

Like KM, when performing EM on dataset #2, as shown in *Figure 4*, *k=2* resulted in the highest value of SC, which was 0.093. I was very surprised by this SC value since it was significantly lower compared to those of the previous three experiments. Given this value was non-negative, it indicated that the clustering effort was successful but barely so. Interestingly, as *k* increased, SC eventually became negative, which was not seen in previous experiments. However, consistent with previous experiments, as shown by the HS and RI values, usage of label information had minimal impact on the algorithm's *k* value selection. Consistent with the results of section 1.2, EM's soft clustering negatively affected the algorithm's clustering output. However, the effect was much more pronounced in dataset #2.

As shown in *Table 1*, the HS value was 0.044, indicating the resulting clusters were very weakly uniform. The RI value was 0.580, indicating more time than not, instances within the same clusters had the same true labels. Although both KM and EM performed poorly when clustering dataset #2, EM outperformed KM's. I call out this information since it is referenced again in Part V.

1.5 Conclusion

Algorithmically, EM is very similar to KM: KM seesaws back and forth between finding the mean/center and assigning instances to clusters. Similarly, EM seesaws back and forth between finding the mean of the Gaussian distribution and the likelihood that the data comes from a specific cluster. The only difference is that EM allows "soft clustering". While KM iteratively minimizes the error metric, EM iteratively maximizes the probabilistic metric.

Interestingly, due to this "slight difference", KM outperformed EM when evaluated on their SCs. Although the difference was minimal in dataset #1 it was significant in dataset #2. My hypothesis is that there exists some important split value in dataset #2 which suffers significantly if soft clustering is used.

It was fascinating to see the power of both algorithms in action. Even without being provided any label information, one can determine an ideal, practical value of *k* for clustering: As shown throughout all four experiments, in most cases, SC, HS and RS, are optimized at a similar *k* value, even though SC does not make use of label information and is therefore completely unsupervised.

Evaluation of Two Clustering Techniques					
Dataset	**Algorithm**	**K**	**SC**	**HS**	**RI**
1	KM	20	0.566	0.181	0.656
1	EM	16	0.514	0.146	0.646
2	KM	2	0.506	0.022	0.520
2	EM	2	0.093	0.044	0.580

TABLE 1. K, SC, HS, RI VALUES RESULTING FROM KM AND EM ON DATASETS 1 AND 2 ARE SHOWN.

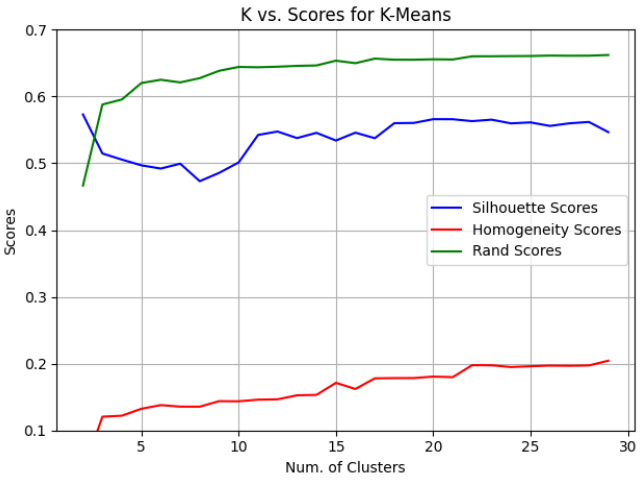

FIGURE 1. *FOR DATASET #1 - K-MEANS, SC* SCORE WAS HIGHEST AT K=20.

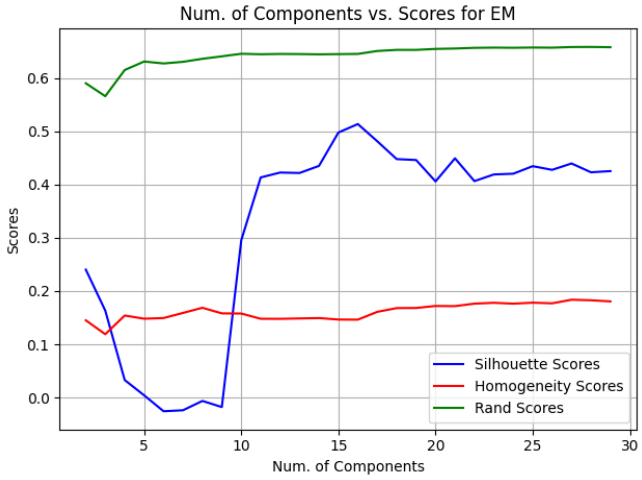

FIGURE 2. *FOR DATASET #1 - EM, SC* SCORE WAS HIGHEST AT K=16.

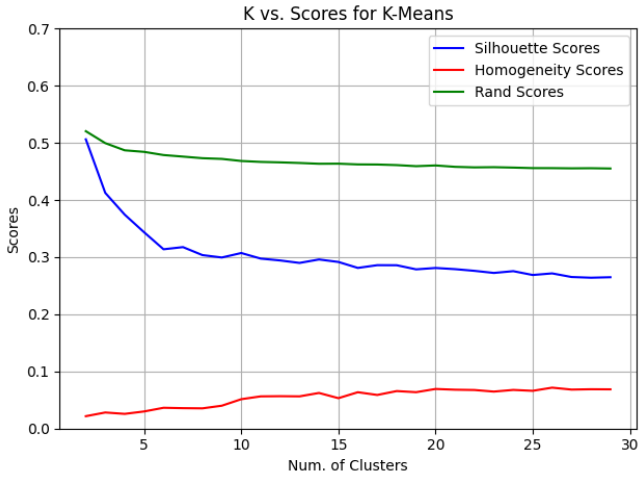

FIGURE 3. *FOR DATASET #2 - K-MEANS, SC* SCORE WAS HIGHEST AT K=2.

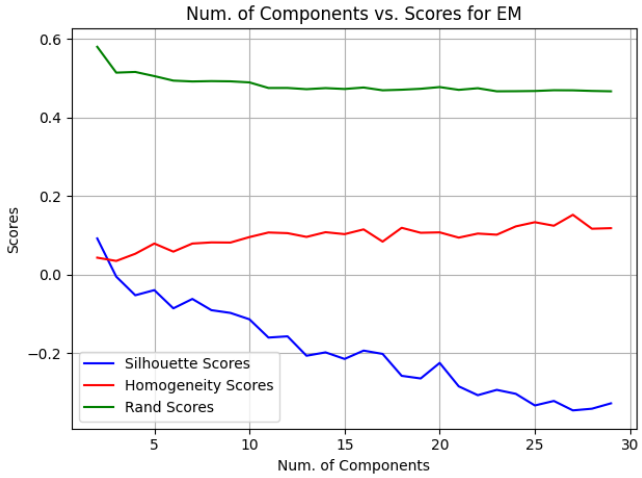

FIGURE 4. *FOR DATASET #2, EM, SC* SCORE WAS HIGHEST AT K=2.

PART II

In Part II, I performed dimensionality reduction (DR) on both datasets using 3 different unsupervised algorithms: Principal Component Analysis (PCA), Independent Component Analysis (ICA), Random Projection (RP) and 1 supervised algorithm: Random Forest (RF). Below, I describe my feature selection methodology. Then, in each subsequent section, I report the outcomes of the dimensionality reduction using each algorithm. To quickly determine the effect of DR, I input the reduced dataset into a dummy classifier (Decision Tree with default hyperparameters) and evaluate the results.

PCA – The ideal combination of features was determined by evaluating the significance of each principal component indicated by their Eigenvalues [7]. To perform the evaluation, their contribution % to the total combined Eigenvalues of each component were plotted on a pie chart. I selected the combination of components whose combined Eigenvalues explained at least 90% of the total using the smallest number of components.

ICA – The ideal number of components was determined by calculating the average kurtosis of all components at each quantity [8]. To perform the evaluation, the average kurtosis at each number of components was plotted on a line chart. I selected the number of components which resulted in the highest average kurtosis value.

RP – The ideal number of components was determined by calculating the mean squared root error (RMSE) between the original training data and the randomly projected data at each quantity [9]. To perform the evaluation, the RMSE at each number of components was plotted on a line chart. I selected the number of components which resulted in the lowest overall RMSE.

RF – The ideal combination of features was determined by evaluating the mean decrease in impurity due to each principal component [10]. To perform the evaluation, their contribution % to the total combined mean decrease in impurity of each component were plotted on a pie chart. I selected the combination of components whose combined mean decrease in impurity exceeded 90% of the total using the smallest number of components. If multiple combinations satisfied this requirement, I chose the combination resulting in the highest reduction in impurity.

2.1 Dataset #1 – PCA

As shown in *Figure 5*, results indicated that when using PCA, only two features: length and diameter were found to be significant. Together, they explained 95.1% of the total variance. Therefore, all others were discarded as a result of DR. As shown in *Table 2*, although accuracy decreased by 0.2%, train time was cut down by 53%. Given 0.2% accuracy isn't critical to Dataset #1's problem domain, the reduction in training time indicates DR using PCA was successful.

2.2 Dataset #1 – ICA

As shown in *Figure 6*, results indicated that when using ICA, using 7 components was most ideal, as it resulted in the highest average kurtosis. Therefore, 1 component was discarded as a result of DR. As shown in *Table 2*, accuracy improved significantly (+3.7%) while training time remained about the same. Given the significant boost in accuracy with no trade-off in train time, DR using ICA was very successful.

2.3 Dataset #1 – RP

As shown in *Figure 7*, results indicated that when using RP, using 2 components was most ideal, as it resulted in the lowest overall reconstruction error. Therefore, 6 components were discarded as a result of DR. As shown in *Table 2*, the results were very similar to that of PCA. But unlike PCA, accuracy increased by 0.4% while train time was cut down by 60%. Given its ability to increase accuracy with no trade-off, DR using RP was very successful.

2.4 Dataset #1 – Random Forest

As shown in *Figure 8*, results indicated that when using RF, a combination of all components except "height" resulted in the highest mean decrease in impurity with a makeup over 90%. Therefore, "height" was discarded as a result of DR. As shown in *Table 2*, the results were very similar to those of ICA. Like ICA, accuracy improved significantly (+3.0%) while train time was about the same. Given the significant boost in accuracy with no trade-off in train time, DR using RF was very successful.

2.5 Dataset #2 – PCA

As shown in *Figure 9*, results indicated that when using PCA, only 3 components, volatile acidity, citric acid, and residual sugar, were found to be significant. Together, they explained 96.4% of the total variance. All others were discarded as a result of DR. As shown in *Table 2*, the results were fascinating. Accuracy increased by 27% while train time was cut by 50%. Given the significant boost in accuracy and reduction in train time, DR using PCA was very successful.

2.6 Dataset #2 – ICA

As shown in *Figure 10*, results indicated that when using ICA, using 10 components was most ideal, as it resulted in the highest average kurtosis. Therefore, only 1 component was discarded as a result of DR. As shown in *Table 2*, accuracy improved significantly (+32.8%). However, training time did increase by 64.6%. I found it interesting that although the dummy classifier had one less feature it needed to classify, training time had increased. It may be that the DR resulted in a better split strategy, which provide additional information gain. In this case, it likely resulted in a higher number of nodes needing to be formed. Given the significant boost in accuracy with acceptable trade-off in train time, DR using ICA was very successful.

2.7 Dataset #2 – RP

As shown in *Figure 11*, results indicated that when using RP, using 10 components was most ideal, as it resulted in the lowest overall reconstruction error. Therefore, only 1 component was discarded as a result of DR. As shown in *Table 2*, the results showed that accuracy increased by 28.7% while train time increased by 47.8%. Given its ability to increase accuracy with acceptable level of trade-off in train time, DR using RP was very successful.

2.8 Dataset #2 – Random Forest

As shown in *Figure 12*, results indicated that when using RF, a combination of all components except "sulphates" resulted in the highest mean reduction in impurity with makeup over 90%. Therefore, "sulphates" was discarded as a result of DR. As shown in *Table 2*, the accuracy improved significantly (+33.16%) while train time was nearly the same. Given the significant boost in accuracy with minimal trade-off in train time, DR using RF was very successful.

2.9 Conclusion

This insightful exercise demonstrated the immense power of DR. As shown in *Table 2*, DR almost always resulted in improved accuracy. In addition, I found it interesting that despite the advantage of having access to the target class while others did not, RF didn't perform much better than others. In fact, in dataset #1, RF performed worse than ICA.

Curse of Dimensionality (COD) – DR's ability to reduce the effect of COD is indicated by the significant improvement of accuracy in dataset #2, compared to that of dataset #1. According to COD's definition, every additional dimension result in an exponentially large amount of data required to supplement the learning process. Since dataset #2 had 11 features, opposed to dataset #1's 8, reducing even a single component helped it significantly.

Computational Complexity (CC) – DR's ability to reduce the effect of CC is indicated by the significant reduction in train time when significant number of dimensions were able to be reduced. This was the case when DR was performed using PCA. As a result, train time was able to be reduced by over 50%.

In some cases, such as when DR was performed using PCA on both datasets, there was no trade-off between accuracy and train time as both were able to be improved. The improvement in accuracy can be largely attributed to the reduction of the effect of the curse of dimensionality. The improvement in train time can be attributed to DR's ability to reduce the computational complexity of the classification problem.

Evaluation of Different Dimensionality Reduction Techniques using Dummy Classifier (Decision Tree)				
Dataset #1	**Without DR**		**With DR**	
Algorithm	**Accuracy**	**Train Time**	**Accuracy**	**Train Time**
PCA	48.2%	0.015s	48.0%	0.007s
ICA			51.9%	0.014s
RP			48.6%	0.006s
RF			51.2%	0.013s
Dataset #2	**Without DR**		**With DR**	
Algorithm	**Accuracy**	**Train Time**	**Accuracy**	**Train Time**
PCA	45.9%	0.015s	73.0%	0.009s
ICA			78.8%	0.022s
RP			74.6%	0.022s
RF			79.1%	0.014s

TABLE 2. EVALUATION OF DIFFERENT DIMENSIONALITY REDUCTION TECHNIQUES WERE CONDUCTED BY COMPARING ACCURACY AND TRAIN

TIME USING A DECISION TREE CLASSIFIER.

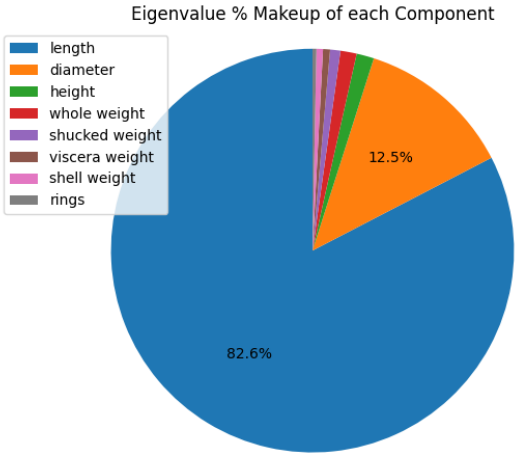

FIGURE 5. *PER PCA, LENGTH AND DIAMETER EXPLAINED 90% OF THE TOTAL VARIANCE*

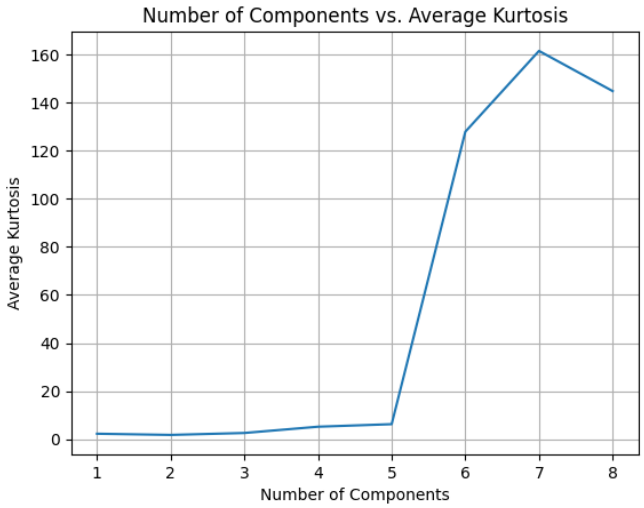

FIGURE 6. *PER ICA, 7 COMPONENTS RESULTED IN THE HIGHEST AVERAGE KURTOSIS*

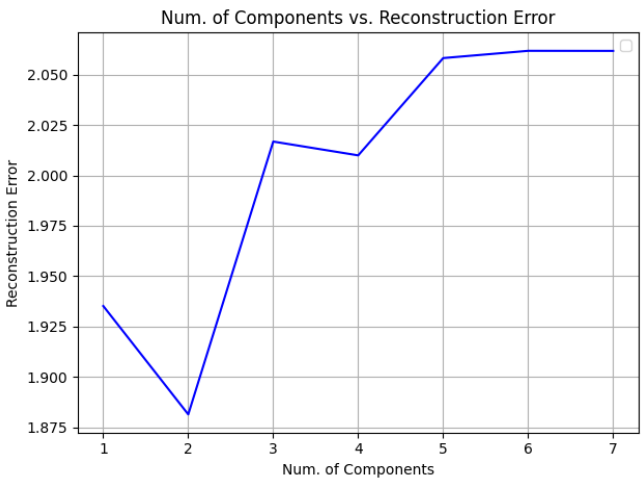

FIGURE 7. *PER RP, 2 COMPONENTS RESULTED IN LOWEST OVERALL RECONSTRUCTION ERROR*

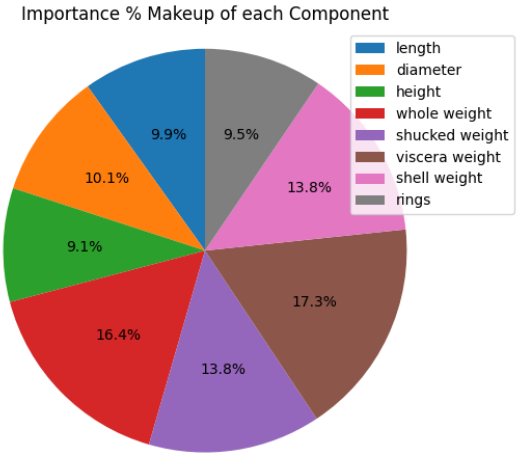

FIGURE 8. *PER RF, A COMBINATION OF ALL COMPONENTS EXCEPT HEIGHT ACHIEVED THE HIGHEST IMPORTANCE % MAKEUP OVER 90*

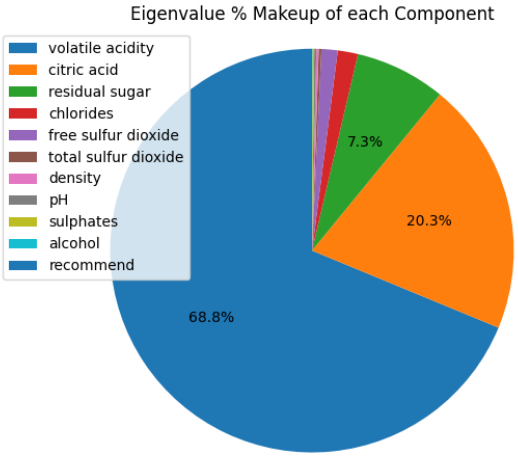

FIGURE 9. *PER PCA, VOLATILE ACIDITY, CITRIC ACID AND RESIDUAL SUGAR EXPLAINED 96.4% OF THE TOTAL VARIANCE*

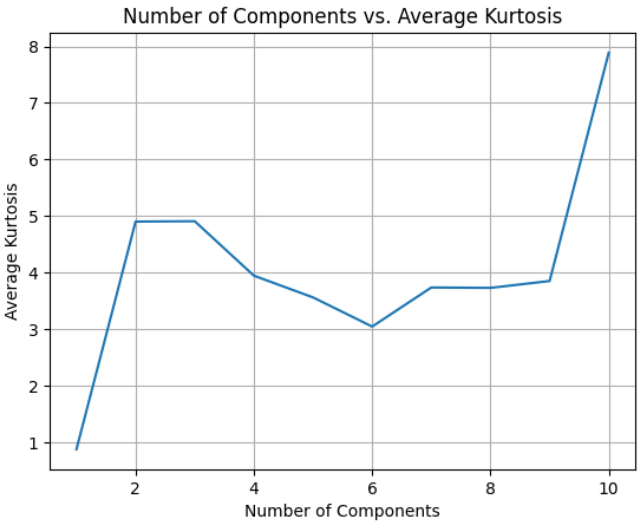

FIGURE 10. *PER ICA, 10 COMPONENTS RESULTED IN THE HIGHEST AVERAGE KURTOSIS*

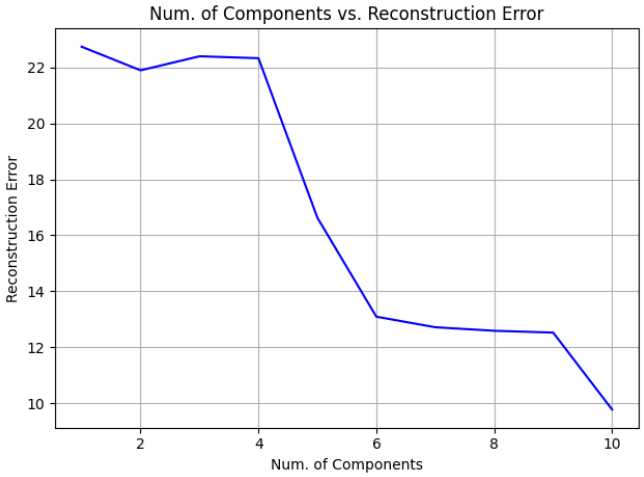

FIGURE 11. *PER RP, 10 COMPONENTS RESULTED IN LOWEST OVERALL RECONSTRUCTION ERROR*

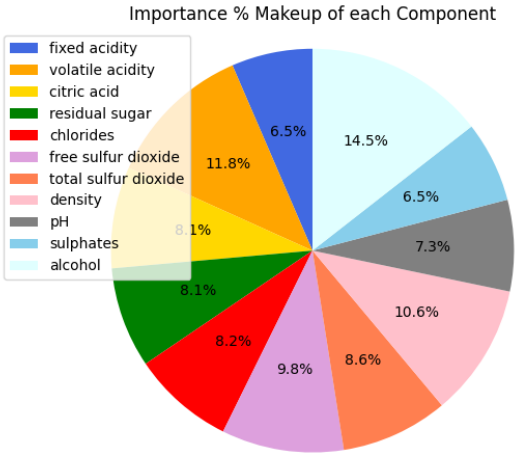

FIGURE 12. *PER RF, A COMBINATION OF ALL COMPONENTS EXCEPT SULPHATES ACHIEVED THE HIGHEST IMPORTANCE % MAKEUP OVER 90*

PART III

In Part III, I effectively repeated Part I in that I again performed the two clustering algorithms, KM and EM. This time, however, I performed the two clustering algorithms on the 8 dimensionally reduced datasets from Part II. Like Part I, I provide the 16 artifacts in table format (*Table 3*). Unlike in Part I, however, I do not speak to each in detail. Instead, I focus on only the parts I found to be most important and interesting.

3.1 Clustering Behaviors within each DR Algorithm

PCA – In PCA reduced dataset #1, clusters formed using both KM and EM had noticeably greater values of SC, HS, and RI. In other words, the resulting clusters were farther apart from each other and were denser. In addition, the clusters were more homogeneous and contained instances which have the same true labels. Given the evaluation using dummy classifier showed similar accuracy using only 2 of 8 features, it made sense intuitively that the reduced dataset resulted in better clustering. Similarly, in dataset #2, clusters formed using KM were also of higher quality. Interestingly, this was not the case when using EM. The reduced dataset likely benefits from hard clustering (opposed to soft clustering) as only the important features were kept.

ICA – In ICA reduced dataset #1, using KM and EM both resulted in significantly smaller *k*-values. Given only 1 feature was able to be removed, it made sense intuitively that significant reduction in number of clusters would result in much less homogenous clusters. Interestingly, despite the significant reduction in *k*-values, the RI remained relatively unchanged, which made sense intuitively with the increased accuracy. Dataset #2 showed consistent behavior, especially when EM was used.

RP – Both RP reduced dataset #1 and #2, showed consistent behavior with that of ICA. I found it fascinating that there were so many similarities in clustering behavior even though the ICA and RP methods were completely different in their approach to feature selection. This was made even more interesting by the fact that their *k*-values did not match in dataset #1. It is likely that although KM separated ICA reduced dataset #1 into different 8 different clusters, there likely existed two groups which were much closer to each other in respect to their inter-cluster distances and consistency.

RF – Both RF reduced dataset #1 and #2 resulted in much higher quality clusters using both KM and EM. It made intuitive sense that the SC, HS, and RI values were all relatively unchanged since only one feature was removed using RF for both datasets. I found it encouraging that the clustering behavior supported the results from Part II, where I observed a significant increase in accuracy in dataset #2. I was afraid there may have been a chance the result was due to overfitting. I address this concern during the Neural Network HP tuning portion of Part IV.

Evaluation of Different Clustering Techniques on Datasets with Reduced Dimensions					
Dataset #1					
Clustering Algorithm	**DR Algorithm**	**K**	**SC**	**HS**	**RI**
KM	Original	20	0.566	0.181	0.656
	PCA	27	0.596	0.198	0.661
	ICA	8	0.199	0.143	0.613
	RP	2	0.470	0.082	0.536
	RF	2	0.573	0.038	0.467
EM	Original	16	0.514	0.146	0.646
	PCA	18	0.577	0.147	0.645
	ICA	2	0.170	0.103	0.559
	RP	2	0.427	0.047	0.491
	RF	19	0.477	0.154	0.648
Dataset #2					
Clustering Algorithm	**DR Algorithm**	**K**	**SC**	**HS**	**RI**
KM	Original	2	0.506	0.022	0.520
	PCA	2	0.507	0.022	0.520
	ICA	10	0.115	0.138	0.480

	RP	2	0.542	0.016	0.516
	RF	2	0.506	0.022	0.520
EM	Original	2	0.093	0.044	0.580
	PCA	2	0.068	0.001	0.502
	ICA	2	0.245	0.040	0.578
	RP	2	0.206	0.032	0.562
	RF	3	0.134	0.048	0.525

TABLE 3. K, SC, HS, RI VALUES RESULTING FROM KM AND EM ON DATASETS 1 AND 2 WITH VARIOUS DIMENSIONALITY REDUCTION METHODS PERFORMED ARE SHOWN.

3.2 Conclusion

It was uncanny that both unsupervised and supervised learning methods behaved so similarly in response to dimensionality reduction. However, after stepping through it, the results made sense intuitively: In a way, the technique in which PCA, ICA, RP and RF perform analysis before dimensionality reduction resembles the act of clustering. Therefore, by performing them, it's as if some form of "pre-clustering" has already been done before the actual clustering using KM and EM even occurs. In conclusion, Part III further demonstrated the tremendous power of dimensionality reduction. It showed that performance of both unsupervised and supervised learning methods can be improved significantly by using it.

PART IV

In Part IV, I gathered the outputs of the dimensionality reduction on dataset #2 from Part II. Using each (PCA, ICA , RP, and RF) transformed dataset, I trained a Neural Network. Keeping consistent with Assignment 1, I tuned each HP using their learning curves to prevent overfitting/underfitting and the GridSearch optimizer to maximize their respective performance. I then compared their performances against that of Assignment 1. For experimental purposes and per instructors' recommendations, I aimed to reduce the number of neurons and size of the input/hidden layers.

The objective of this experiment was to evaluate the power of dimensionality reduction by measuring the performance of neural network classifiers. Keeping consistent with Assignment 1, performance was evaluated primarily on model's cross-validation accuracy and training wall-clock time. Within each section, the NN's general convergence behavior and number of iterations required for convergence were also discussed since they were relevant and/or interesting. For reference, the performance metrics of the original NN from Assignment 1 is listed in *Table 4* under the method: "Original". In this section, "control" refers to the same, original dataset #2 from Assignment 1.

4.1 NN Performance on Dataset #2 DR via PCA

This experiment was very interesting. As was the case when I used the dummy classifier (i.e., Decision Tree) in Part II, the resulting NN performed noticeably worse compared to the control in respect to accuracy (*See: Table 4*). However, due to its considerable reduction (8 out of 11 features were removed) in dimensional space using PCA, the fully tuned NN only required 10 neurons with only a single layer. This was a stark contrast from the NN from Assignment 1, which required 2 layers with 500 neurons each. In turn, due to the significant reduction in computational complexity, its train time was a mere 0.473 seconds, which is significantly better than that of the control, which was 29.223 seconds. Likely due to significant reduction in noise as a byproduct of the dimensionality reduction, as shown in *Figure 13*, its loss curve also showed minimal oscillation, resulting in a very aesthetically pleasing convergence trend.

4.2 NN Performance on Dataset #2 DR via ICA

Compared to the results of NN produced via PCA, I found NN produced via ICA much more difficult to tune. The difference was night and day, and it reminded me of the painstaking HP tuning that the original dataset required to optimize. Interestingly, its performance in respect to both accuracy and training time was able to be reduced significantly. Although DR using ICA only resulted in removal of 1 feature, it was likely one which was causing significant noise as its removal resulted in a much smoother loss curve as shown in *Figure 14*. The higher number of iterations it required to converge likely can be attributed to its ability to find additional, more informative patterns it previously could not find due to noise.

4.3 NN Performance on Dataset #2 DR via RP

This experiment was interesting because both DR using ICA and RP resulted in a reduction of 1 feature. However, the resulting, transformed dataset behaved very differently when put to the test using NN. As shown in *Table 4*, despite its having 1 fewer feature compared to the control, the NN required more than twice as long to train with respect to both the number of iterations and train time. As shown in *Figure 15*, no matter how I tuned its HPs, loss curve was not ideal and showed just as much, if not more oscillation compared to the control, indicating the persistence of noise. The likely explanation is that unlike ICA's, reduction performed using RP failed to remove the feature which contributed to the noise. In addition, given its reduced accuracy, it is possible that instead, RP removed an informative feature.

4.4 NN Performance on Dataset #2 DR via RF

I found the results of this experiment surprising. The reason this experiment was supposed to be interesting is because my chosen algorithm, Random Forest, is a supervised learner. In other words, unlike PCA, ICA and RP it has an unfair advantage because it utilizes labels to determine which features can be removed. In other words, RF had the answer key to the test and still scored less than ICA in respect to accuracy. To be fair, RF was able to beat PCA and RP in respect to accuracy and was able converge in significantly less time and iterations as shown in *Figure 16*.

4.5 Conclusion

This experiment further proved that dimensionality reduction is very powerful. Its use-case is not limited to simple learners such Decision Tree and can be used on very complex learners

such as multi-layer Neural Networks. Furthermore, it was an informative, practical demonstration which showed that not all dimensionality reduction techniques/algorithms are created equal, and each have different use cases

As evidenced by the train time, iterations and oscillation in *Table 4*, dimensionality reduction can have a significant effect on the resulting learner's convergence behavior. It also has a significant impact on the model's accuracy. I can see why machine learning engineers in the real-world would use dimensionality reduction techniques, as it saves them time and computational resources often with little to no trade-off. In some cases, not only is there no trade-off, as demonstrated by the datasets reduced using ICA and RF, accuracy may even increase as a result of the reduction. It seems almost too good to be true, but the results of my experiments indicate that it is.

Evaluation of Different Dimensionality Reduction Methods using Neural Network (MLP Classifier)				
Dataset #2				
Method	**Accuracy**	**Train Time**	**Iterations**	**Oscillation**
Original	72.2%	29.223s	137	High
PCA	68.6%	0.282s	70	Low
ICA	75.7%	5.332s	250	Med
RP	71.1%	65.159s	290	High
RF	73.9%	1.165s	150	Low

TABLE 4. EVALUATION OF DIFFERENT DIMENSIONALITY REDUCTION TECHNIQUES USING NEURAL NETWORKS.

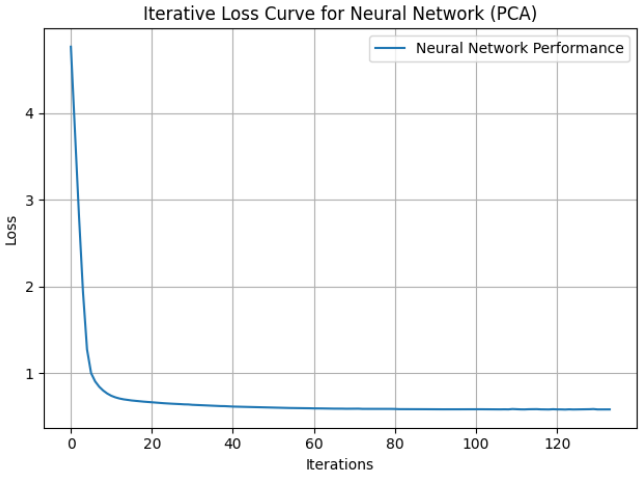

FIGURE 13. *NN TRAINED USING DATASET #2 REDUCED VIA PCA SHOWED A BEAUTIFUL, SMOOTH LOSS CURVE*

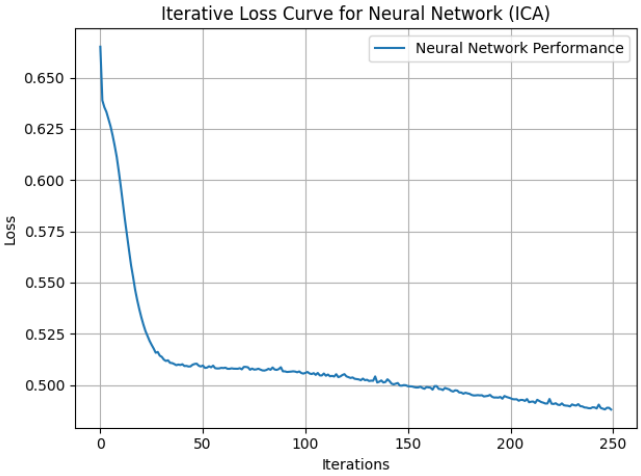

FIGURE 14. *NN TRAINED USING DATASET #2 REDUCED VIA ICA SHOWED A SMOOTHER LOSS CURVE*

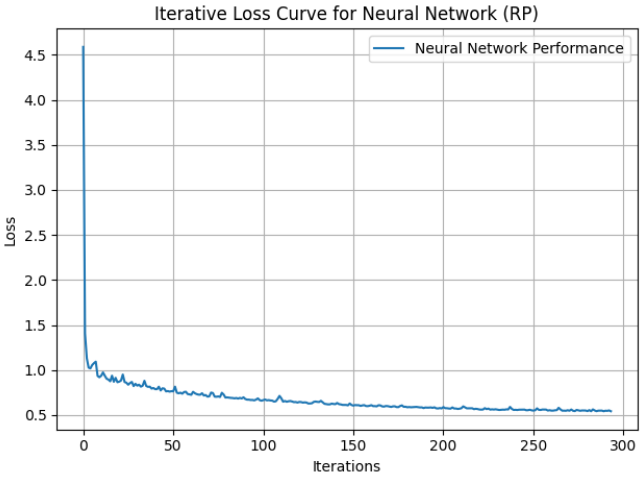

FIGURE 15. *NN TRAINED USING DATASET #2 REDUCED VIA RP SHOWED A LOSS CURVE WITH A HIGH NUMBER OF OSCILLATIONS*

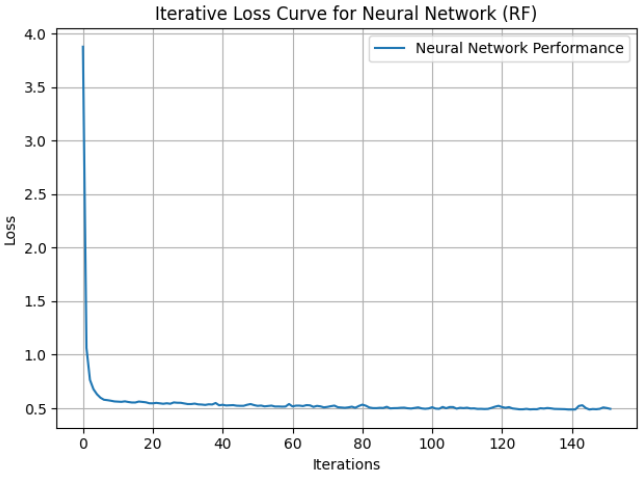

FIGURE 16. *NN TRAINED USING DATASET #2 REDUCED VIA RF SHOWED A FAIRLY SMOOTH LOSS CURVE WITH VERY LITTLE OSCILLATIONS*

PART V

In Part V, I again utilized my findings from Part II. I repurposed the cluster labels generated from performing KM and EM on dataset #2 by using them as new features. I began by making two copies of the X-training data from the original dataset #2. To the first copy, I added the KM cluster labels under a new column, "kmc". To the second copy, I added the EM clusters labels under a new column, "emc".

The objective of this experiment was to evaluate the effect of the two clustering methods using Neural Network classifiers. Like I did in Part IV, I trained multiple Neural Networks. The first NN was trained using the original dataset #2 with the "kmc" column, and the second NN was trained using the original dataset #2 with the "emc" column. The same strategies described in Part IV were used to tune the HPs and evaluate the final neural networks. For reference, the performance metrics of the original NN from Assignment 1 is listed in Table 5 under "Original".

5.1 NN Performance on Dataset #2: Original + K-Means

To reemphasize, to compare the effects of using the KM cluster labels as columns on the performance of Neural Networks, the "kmc" column was added in addition to the 11 original features. Given the addition of another dimension, I acknowledged that the positive effect of the clustering method may not be as pronounced or may even be completely negated due to the curse of dimensionality.

I was pleasantly surprised to find that despite the curse of dimensionality, the addition of "kmc" column resulted in an astounding 2.2% increase in accuracy and 97% reduction in time required to train the model. The significant reduction in train time was due to the model needing fewer input/hidden layers, and fewer neurons within the layer compared to the original. An intuitive explanation was that the clustering method did a remarkable job separating out the two true label groups. In other words, I believe due to the successful cluster labels that KM was able to generate, the classification problem was made much simpler, resulting in the NN requiring much fewer iterations and train time to find a convergence point. This explanation is also consistent with the behavior of the loss curve shown in *Figure 17*. The high degree of oscillation throughout the graph can be attributed to the noise from the original dataset.

5.2 NN Performance on Dataset #2: Original + Exp. Max.

Given the great performance of NN using cluster labels generated using KM, I had even higher hopes for that of EM. As I mentioned in Part I (*Table 1)*, when evaluating the quality of clusters, those generated using EM exhibited higher homogeneity and rand scores compared to those generated using KM. Intuitively, scoring higher on those two scores indicated to me that the clusters were better at separating out the datasets. Furthermore, given the ideal *k* value of 2, it closely resembled binary classification. In other words, EM was better at "classifying" the dataset, compared to KM.

Consistent with my hypothesis, results showed that the addition of "emc" column produced an NN which performed even better than the previous one which used "kmc". It resulted in a 3.0% increase in accuracy and 95.6% reduction in the time required to train the model. Again, as discussed in section 5.1, I believe this can be attributed to the successful cluster labels generated using EM. The classification problem became much simpler, resulting in the NN requiring less iterations and comparable train time to find an even better convergence point. Again, its loss curve shown in *Figure 18* supports this explanation. The smaller degree of oscillation is likely due to the more accurate cluster labels which were provided to the NN.

5.3 Conclusion

These two experiments further proved the usefulness of clustering methods. While the experiments in Part I demonstrated the power and resourcefulness of the clustering methods, Part V showcased their versatility and practicality. I was thoroughly impressed with the ingenuity behind both algorithms and how effectively they were able to gain information, even without being provided any labels.

Curious to see how the cluster labels fared alone, without any help from the other 11 features from the original dataset, I ran two additional tests (outside of the requirements of the assignment). As shown in the last two rows of *Table 5*, using only the labels provided by KM or EM, NN's were able be trained to perform with accuracy of 66.5% and 70.0% respectively. Although the performance isn't as high as the NN's trained with the original features, the fact that using only the cluster labels resulted in NN's with such high accuracy is truly astounding.

Evaluation of Two Clustering Methods using Neural Network (MLP Classifier)				
Dataset #2				
Method	**Accuracy**	**Train Time**	**Iterations**	**Oscillation**
Original	72.2%	29.223s	137	High
Original Features + KM	74.4%	0.905s	113	High
Original Features + EM	75.2%	1.315s	89	Med
KM only	66.5%	0.061s	20	Low
EM only	70.0%	0.320s	24	Low

TABLE 5. EVALUATION OF TWO DIFFERENT CLUSTERING TECHNIQUES USING NEURAL NETWORKS.

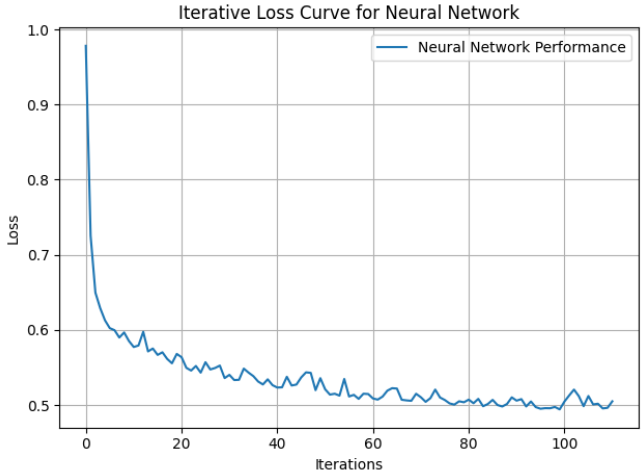

FIGURE 17. *NN PRODUCED USING DATASET #2 WITH THE ADDED 'KMC' COLUMN SHOWED A LOSS CURVE WITH A HIGH DEGREE OF OSCILLATION*

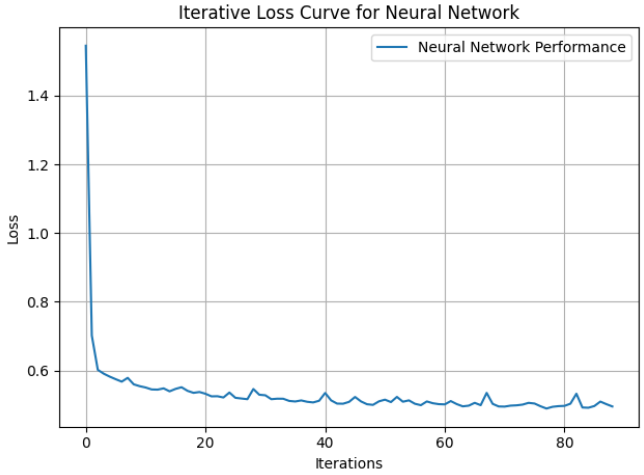

FIGURE 18. *NN PRODUCED USING DATASET #2 WITH THE ADDED 'EMC' COLUMN SHOWED A LOSS CURVE WITH A MEDIUM DEGREE OF OSCILLATION*

REFERENCES

[1] Sklearn.cluster.kmeans. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[2] Sklearn.mixture.gaussianmixture. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture

[3] Wikimedia Foundation. (2022, July 20). Elbow method (clustering). Wikipedia. Retrieved November 3, 2022, from https://en.wikipedia.org/wiki/Elbow_method_(clustering)

[4] Sklearn.metrics.silhouette_score. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[5] Sklearn.metrics.homogeneity_score. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html

[6] Sklearn.metrics.rand_score. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.rand_score.html

[7] SKLEARN.DECOMPOSITION.PCA. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[8] Scipy.stats.kurtosis#. scipy.stats.kurtosis - SciPy v1.9.3 Manual. (n.d.). Retrieved November 3, 2022, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kurtosis.html

[9] SKLEARN.METRICS.MEAN_SQUARED_ERROR. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

[10] Feature importances with a forest of trees. scikit. (n.d.). Retrieved November 3, 2022, from https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html