Unsupervised Learning and Dimensionality Reduction Analysis

John Seon Keun Yi

Introduction

This project explores various clustering and dimensionality reduction algorithms. The two datasets used for this analysis are

- **Heart Disease Dataset**

  This is the dataset used in Assignment 1. This dataset contains 13 attributes on various health conditions of the subject and a target field of 0 (no heart disease) and 1. (presence of the disease) This dataset has 303 instances, of which 138 (46%) are 0 and 165 (54%) are 1. The binary outputs of the target data were changed to 'yes' and 'no' to adjust to the needs of the WEKA Explorer.

- **Breast Cancer Wisconsin Database**

  This dataset works to predict breast cancer from features computed from digitized images of a fine needle aspirate of a breast mass. This dataset contains 30 attributes such as mean radius and mean smoothness. All values in the attributes are numerical. They were normalized using the WEKA normalize tool. It has a total of 569 instances with no missing values present. This dataset was retrieved from the UCI machine learning repository.

I chose to use these two datasets because I wanted to compare results of binary classification datasets with different number of attributes. The breast cancer dataset had more than twice the number of attributes than the heart disease dataset. I thought it would be interesting to compare the clustering results of the two datasets both before and after applying dimensionality reduction.

1. Clustering

K-means Clustering

K-means clustering algorithm (K-means in short) clusters samples into k groups of equal variances. It randomly picks k centers which claim closest points based on the mean squared errors. Then it recomputes the centers by averaging the points in each cluster.

Although we already know the number of clusters that should form for both datasets, (two because binary classification) since this is an unsupervised learning problem, we attempt to find the optimal k-value, which is the number of clusters. For both datasets I varied the value of k from 2 to 20, and observed how the sum of squared error (SSE) turned out for different k values. I used the KValid package from WEKA to plot the results. Figure 1 and 2 are the plotted graphs for k value to SSE. To evaluate the performance of the clustering, first I attempted to use the elbow method. The elbow method is a visual identification method where an elbow -a point where the

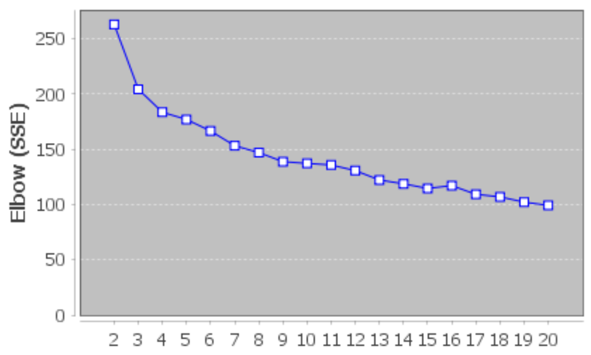


Figure 1. Sum of squared error (SSE) to k-value of the Heart Disease dataset.

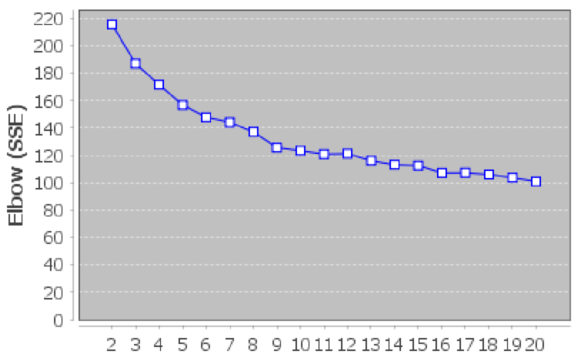


Figure 2. Sum of squared error (SSE) to k-value of the Breast Cancer dataset.

SSE starts to even out and stop changing rapidly- is found. From the plot for the Heart Disease dataset in Figure 1 we can easily identify the 'elbow point', at $k = 2$. However, it is hard to visually distinguish the elbow point for the Breast Cancer dataset in Figure 2. As an alternative, I used the Silhouette method for this dataset. The silhouette method is a cluster performance evaluation method where the silhouette coefficient is compared. The silhouette coefficient s is calculated as follows:

$$s = \frac{b - a}{\max(a, b)}$$

'a' is the mean distance between a sample and other points in the same cluster, and 'b' is the mean distance between the sample and all other points in the nearest next cluster. This measures how closely the sample is matched to the data within its cluster and how loosely it is matched to the data of the next closest cluster. s ranges between 1 and -1, where an s close to 1 indicate that the sample is in the appropriate cluster.

I applied the silhouette method using the KValid package downloaded to WEKA. As a result, the optimal k value for the Breast Cancer dataset turned out to be when $k = 2$. When run with the found k values, the K-means algorithm incorrectly clustered 38% of instances on the Heart Disease dataset, and 7.2% of the instances on the Breast Cancer dataset. K-means successfully clustered much more of its instances that the other dataset. I theorized that the reason for this is that the Breast Cancer dataset has less noise and distinguishable instances, allowing hard clustering algorithms like k-means to cluster them effectively.

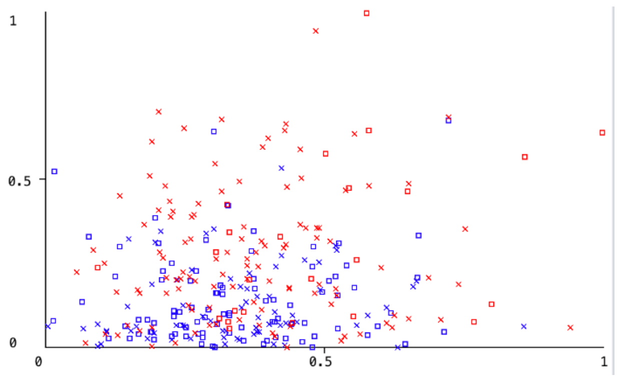


Figure 3. K-means cluster on Heart Disease dataset.

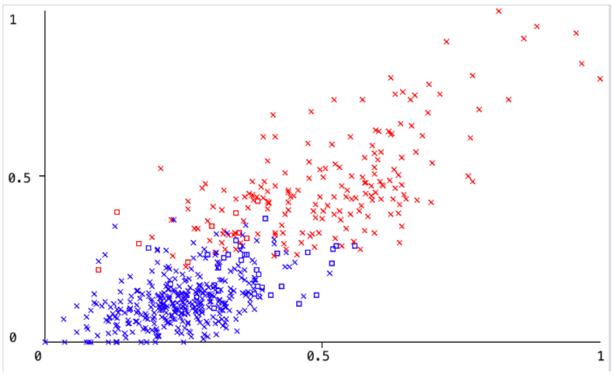


Figure 4. K-means cluster on Breast Cancer dataset.

Figures 3 and 4 display the clusters generated between two attributes for both datasets. Unlike the Breast Cancer dataset which display a clear distinction between the two clusters, the Heart Disease dataset does not show a clear boundary. Since we use labeled data, we also look if the clusters line up with the labels. The first dataset (Figure 5) do not show two clear distinctions on the target-cluster plot compared to the other. (Figure 6) This explains the higher error rate the Heart Disease dataset has.

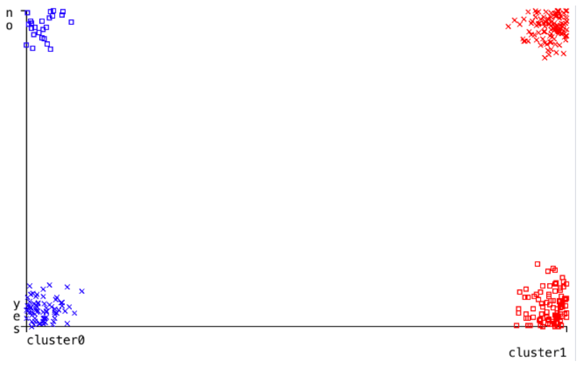


Figure 5. Cluster-target plot of dataset 1.

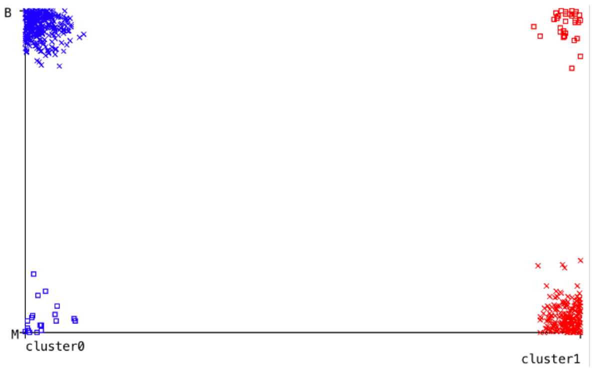


Figure 6. Cluster-target plot of dataset 2.

Expected Maximization (EM)

Expected Maximization (EM) clustering algorithm is a soft clustering algorithm that allows some points to be shared between multiple clusters. Unlike hard clustering such as k-means, where a point is either in a cluster or not, EM assigns to each instance probabilities for that point to be in some cluster.

For this algorithm, first I ran both datasets on the EM clusterer in WEKA with a default setting. In such setting, WEKA finds the optimal number of clusters through cross validation. The k value for the Heart Disease dataset turned out to be 5 with a log likelihood of 8.17947. The percentage of incorrectly clustered instances was 56.43%. I compared this with EM run on $k = 2$ which was found with the k-means algorithm. EM with $k = 2$ had a log likelihood of 0.994 with 23.1023% of incorrectly clustered instances. While it seems logical to pick $k = 5$ as the optimal k value due to its higher log likelihood, it is highly likely that $k = 5$ achieved a higher score by overfitting. This is supported by that fact that despite a lower log likelihood, EM with $k = 2$ has a lower percentage of incorrectly clustered instances. Thus, we can say that $k = 2$ is the optimal value.

This was a similar situation with the Breast Cancer dataset. EM run on default parameters returns a k of 14 with a log likelihood of 39.8508 and 76% of incorrectly clustered instances. EM run with $k = 2$, found using k-means algorithm returns a log likelihood of 27.297 and only 8% of incorrectly clustered instances. As in the other dataset, although $k = 14$ has the higher log likelihood, it makes one hard to believe that this is the optimal k value seen that $k = 2$ has a significantly less percentage of falsely clustered instances. Again, we can conclude that $k = 2$ is the optimal k value for this dataset, and $k = 14$ overfitted.

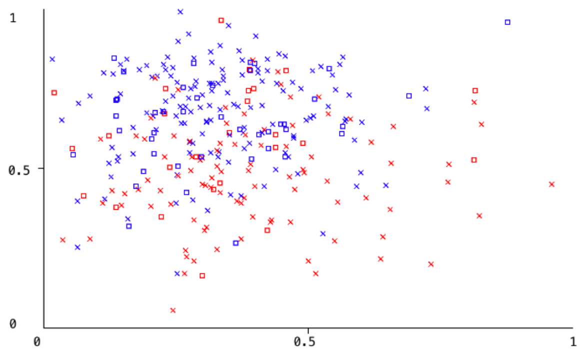


Figure 7. Clusters of EM algorithm for Heart Disease dataset. Optimal k value is 2.

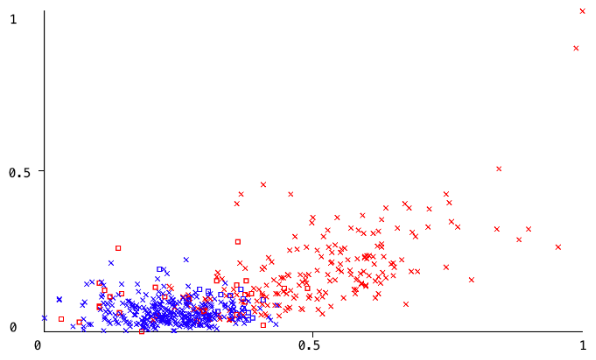


Figure 8. Clusters of EM algorithm for Breast Cancer dataset. Optimal k value is 2.

Figures 7 and 8 display the EM clusters using the optimal k values determined from above. The clusters for the first dataset are not clearly distinguishable, but it is better than the clusters for k-means on Figure 3. I think this is because the EM is better at clustering noise and instances that belong in multiple clusters by giving them probabilities. The Breast cancer dataset show a clearer distinction between clusters, but it is slightly more noisy that that of k-means. The running time of EM to determine the optimal k value was 0.75 seconds for the first dataset, and 55.56 seconds for the second dataset. We can assume the running time of EM increases with more attributes. The target-cluster plot for the datasets turns out very similar to Figure 5 and 6. Still, the Breast Cancer dataset clusters line up better with the labels.

Table 1. Optimal k-values for two clustering algorithms on two datasets.

k-value	Heart Disease	Breast Cancer
k-means	k=2	k=2
EM	k=5	k=14
EM-optimal	k=2	k=2

Table 2. Percentage of incorrectly clustered instances for each algorithm in two datasets.

%	Heart Disease	Breast Cancer
k-means	38.6139	7.2056
EM	56.4356	75.9227
EM-optimal	23.1023	8.7873

The tables above outline the k-values and percentage of incorrectly clustered instances for each clustering algorithms performed on the two datasets. We can see that the predicted optimal k values from EM algorithms were not optimal, and following the k values found from the k-means algorithms turned out to be more accurately representative of the clusters. Also, we can see that the Heart Disease had a lower percentage of error for the EM algorithm compared to k-means, while for the Breast Cancer database k-means had a slightly lower error. We can deduce from this that the first algorithm had more suited domains for the expectation maximization algorithm.

Overall, for these two datasets, I would trust k-means algorithm to find the optimal k value. Although expectation maximization may have less error in some cases (for the first dataset), k-means took much less computing time and its performance evaluation turned up optimal k values that were close to the true number of clusters.

2. Dimensionality Reduction

Dimensionality reduction attempts to increase performance by reducing the number of features for learning algorithms. The two approaches to dimensionality reduction are feature selection and feature transformation.

Since this project is ultimately on unsupervised learning, I attempted to find the optimal k value like I did for the two clustering algorithms in part 1. However, I thought it would be much easier to set k to 2 in order to compare the effects dimensionality reduction has on clustering. Thus, in this part I will set the k value to 2 for all clustering algorithms.

Principal Components Analysis (PCA)

Principal Components Analysis (PCA) is a feature transformation algorithm that performs dimensionality reduction by applying orthogonal transformation. The WEKA Principal Components filter combines possibly linearly correlated attributes into features and ranks the features according to its eigenvalue. I used a variance of 0.95 for the PC filter and applied the features to both clustering algorithms.

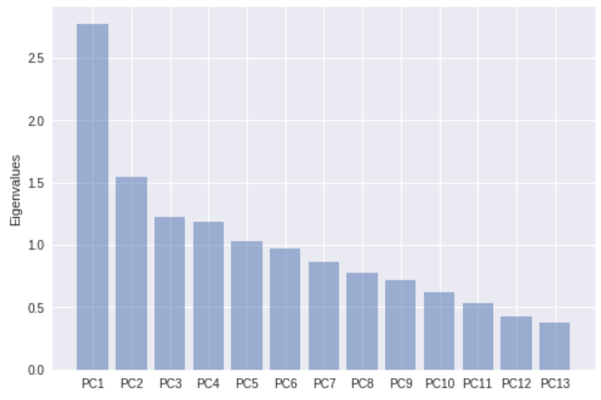


Figure 9. Eigenvalue distribution of attributes in Heart Disease dataset.

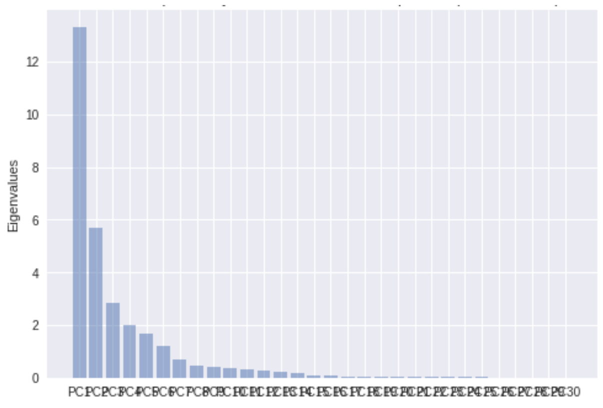


Figure 10. Eigenvalue distribution of attributes in Breast Cancer dataset.

First I looked at the distribution of eigenvalues after applying PCA to both datasets. PCA calculates the variance for all attributes. Attributes with higher variance larger impact on classification. PCA in WEKA combines the attributes to create new features that are ranked on their eigenvalues. In order to cut running time and increase performance, we use only part of the features based on their eigenvalues. I used the Kaiser criterion which is to drop all features with eigenvalues under 1. As a result the first dataset was left with 5 features and the other 6. This is a significant reduction from the 15 and 30 attributes in the original datasets.

Table 3. Percentage of incorrectly clustered instances for datasets feature reduced with PCA.

%	Heart Disease	Breast Cancer
k-means	19.1419	8.4359
EM	34.3234	18.1019

Table 3 outline the error rates of the clustering algorithms run on the datasets with reduced features. The error rate decreased noticeably for the Heart Disease dataset, while slightly increased for the Breast Cancer dataset. This shows that the attributes in the prior dataset has more dependency with each other compared to the latter. The Breast Cancer dataset performed worse because its attributes are more independent than correlated. PCA combines attributes to create new features, which worked well for the first dataset but did not for the other.

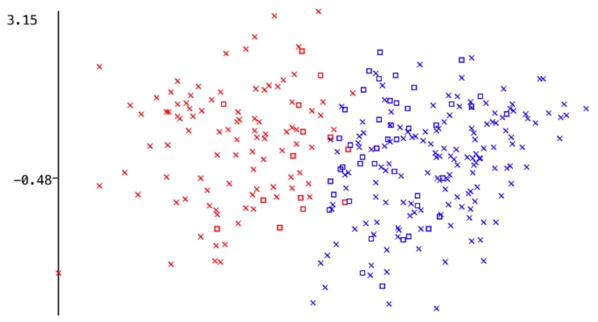


Figure 11. PCA-Kmeans cluster for Heart Disease dataset.

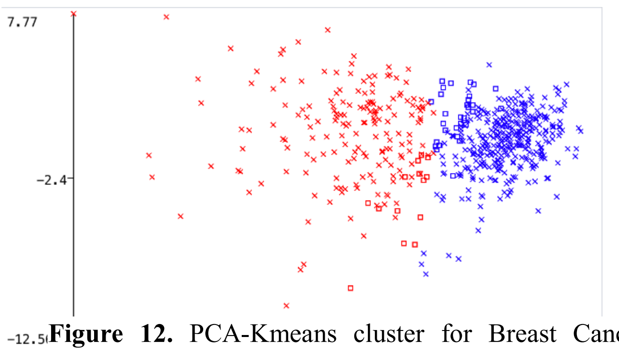


Figure 12. PCA-Kmeans cluster for Breast Cancer dataset.

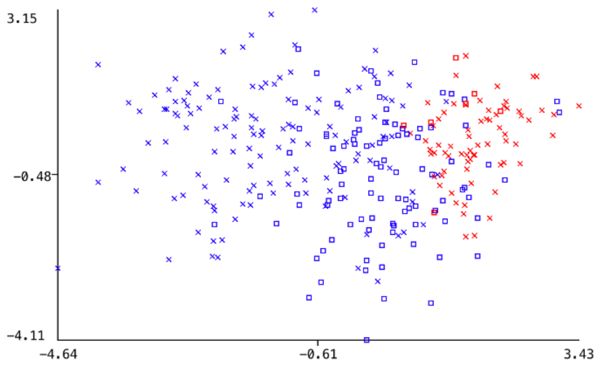


Figure 13. PCA-EM cluster for Heart Disease dataset.

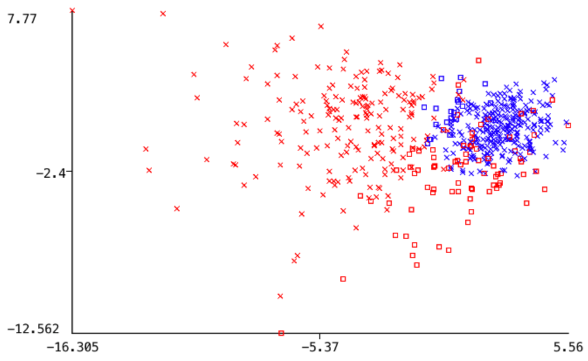


Figure 14. PCA-EM cluster for Breast Cancer dataset.

Shown above are the clusters for the PCA features run in k-means and EM clustering algorithms. The clusters for the Breast Cancer dataset look similar for both algorithms. However, some of the red points overlap with the other cluster on the EM clusters. (Figure 14) I suspected this is because EM is a soft clustering algorithm, this allowing some instances to be in both clusters at the same time. Because of this, applying EM clustering to the second PCA dataset decreased accuracy.

The first dataset showed a much clear distinction of clusters when clustered with the PCA features. Unlike Figure 3 and 7, the new clusters Figure 11 and 13 can be split easily if we draw a line in the boundary. However, compared with k-means, EM doesn't seem to divide the instances properly; the data is biased to the blue cluster. Again I can suspect that the reason k-means and EM has different clustering results is due to the probabilistic nature of EM clustering.

The cluster-target plots for PCA look very similar to Figure 5 and 6. For the Heart Disease dataset, (Figure 5), PCA cluster displays a better line up with the labels with the points in the upper left and lower right moved to the lower left and upper right corners.

Independent Component Analysis (ICA)

Independent component analysis (ICA) is a dimensionality reduction algorithm that aims to maximize independence between features. It performs well for independent attributes with less correlation between each other. For this analysis the fastICA package in WEKA was used. To measure the fit of ICA in each dataset, I measured the kurtosis of all the features in the datasets. Kurtosis measures a distribution based on the closeness to it to the normal distribution. The closer the kurtosis value is to 3, the similar the distribution to the normal (Gaussian) distribution. Generally, ICA is considered to have performed better when the kurtosis of the features are far from 3. Figure 15 and 16 are bar graphs measuring the kurtosis of ICA generated features from the datasets. The x axis is the number of components (features) used. Starting from the original number of features,

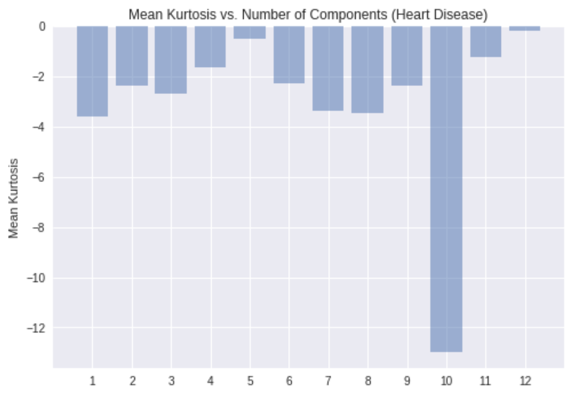


Figure 15. Mean kurtosis of features for Heart Disease dataset when x number of components are used.

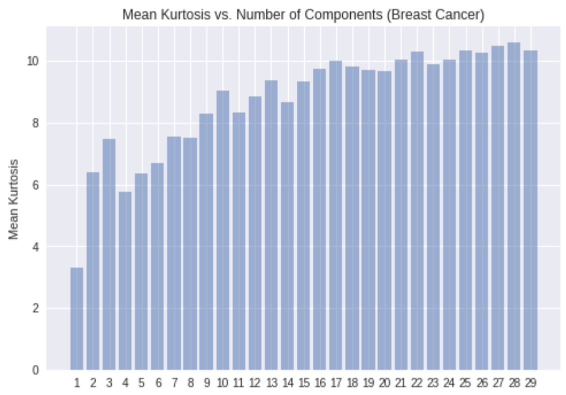


Figure 16. Mean kurtosis of features for Breast Cancer dataset when x number of components are used.

a feature is deleted every iteration and the mean kurtosis of the features used is plotted. Seen that both plots mostly display kurtosis far from 3, we can say that ICA successfully pulled out independent components. The Breast Cancer dataset had overall higher kurtosis, it is safe to say that the dataset has mutually independent features. The kurtosis for Heart Disease dataset is highest when 10 features are used. Figure 16 show an increasing trend when the number of components increase. The kurtosis is highest when 28 components are used. I used these data as the target dimension for ICA. Below are the results of ICA run on the two clustering algorithms.

Table 4. Percentage of incorrectly clustered instances for datasets feature reduced with ICA.

%	Heart Disease	Breast Cancer
k-means	32.3432	37.4341
EM	45.8746	28.471

Interestingly, ICA returned worse error rates for all cases for the two datasets. One reason for this outcome could be the low k value for clustering. Since ICA looks for independence between features and not correlation, a k value of 2 might not have been enough to distinguish the features, thus the low accuracy.

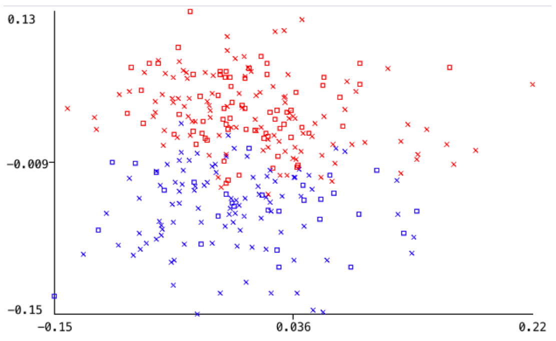


Figure 17. ICA-Kmeans cluster for Heart Disease dataset.

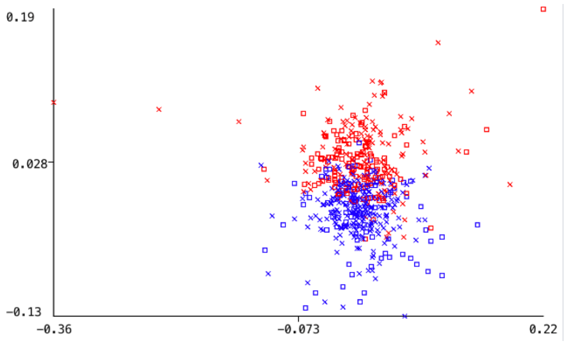


Figure 18. ICA-Kmeans cluster for Breast Cancer dataset.

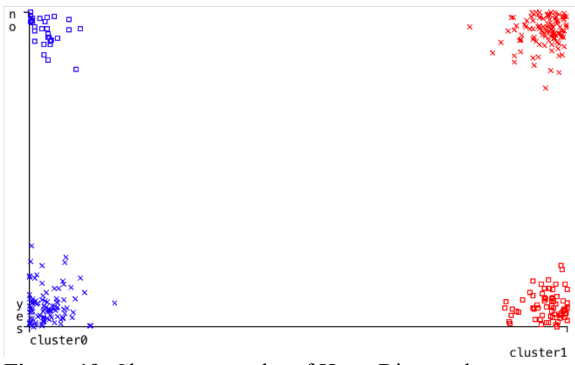


Figure 19. Cluster-target plot of Heart Disease dataset.

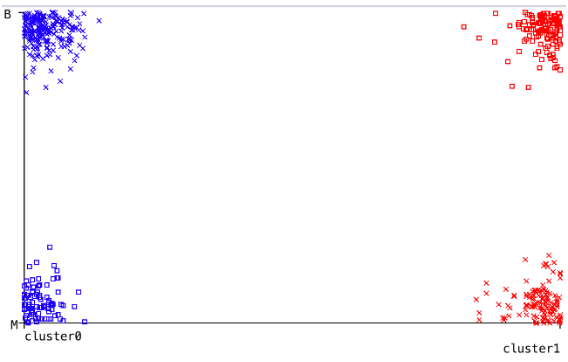


Figure 20. Cluster-target plot of Breast Cancer dataset.

Another explanation is that the clusters generated from the features made by ICA do not correspond to the target clusters. K-means clustering for the Breast Cancer dataset shows rather distinguishable clusters (Figure 18). However, the clusters and target labels do not match well (Figure 22). This is interesting because this dataset showed strong independence between features in the kurtosis analysis. This explains that although features are mutually independent, it does not necessarily lead to the right clustering.

Randomized Projections (RP)

Randomized projection generates a random matrix in order to project the number of attributes to a lower dimension. While the results of RP are not as good as other dimensionality reduction algorithms such as PCA and ICA, it returns relatively similar results without the complexity of the other algorithms.

First, I ran the default setting of RP on WEKA for the two datasets. The default setting has the number of target dimensions at 10. Below are the results.

Table 5. Percentage of incorrectly clustered instances for datasets feature reduced with RP

%	Heart Disease	Breast Cancer
k-means	36.3036	11.4236
EM	36.3036	10.7206

As seen in the results table, the error rates of RP were slightly worse or similar for the Heart Disease dataset, and similar or better for the Breast Cancer dataset. What was interesting was that RP returned these results within a fraction of time of PCA and ICA. For example, when ICA for the second dataset took 0.06 seconds for EM clustering, RP took 0.02 seconds with a higher accuracy.

Below are the clusters for k-means clustering on the two datasets. EM clustering returned very similar results. RP run on the Breast Cancer dataset had similar accuracies with PCA, and better accuracies than the ICA

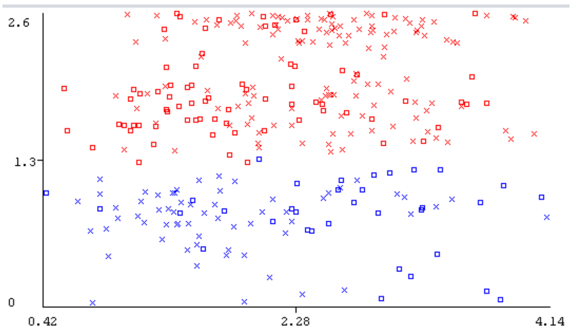


Figure 21. RP-Kmeans cluster for Heart Disease dataset.

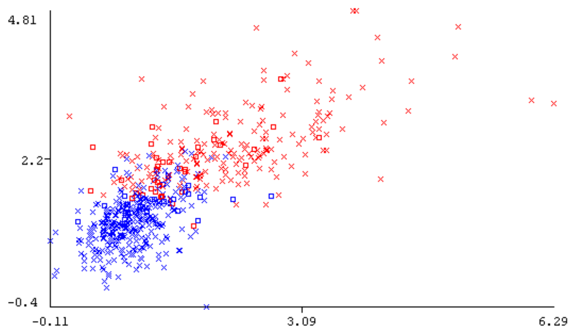


Figure 22. RP-Kmeans cluster for Breast Cancer dataset.

algorithm. This is explainable with the distinguishable clustering in Figure 22. However, RP struggles to cluster the Heart Disease dataset. Although in Figure 21 we can see a horizontal split between the clusters, but it is not strong enough. Furthermore, if we check the cluster-target plot of the dataset, we can see that the labels and clusters do not line up very well. As seen in Figure 23, the two clusters are almost equally spread in the two labels yes and no. I think this happened because randomized projection generates features by passing it through a random matrix. I suspect that creates features that do not necessarily cater to the targets.

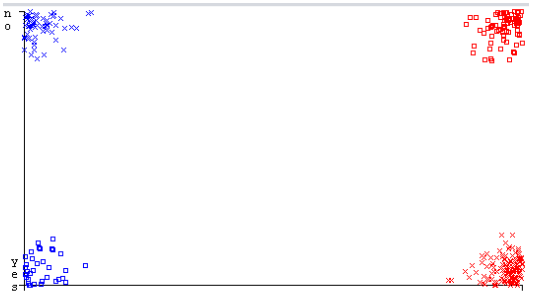


Figure 23. Cluster-target plot of Heart Disease dataset.

Another behavior I observed was that RP returns different features when run with different 'seeds'. Seeds are what determines the variables that end up in the random matrix. To test this, I varied the seed from 10 to 50 on the Heart Disease dataset and tested the error rate when clustered with k-means. Table 6 outlines the percentage of incorrectly clustered instances by different seeds.

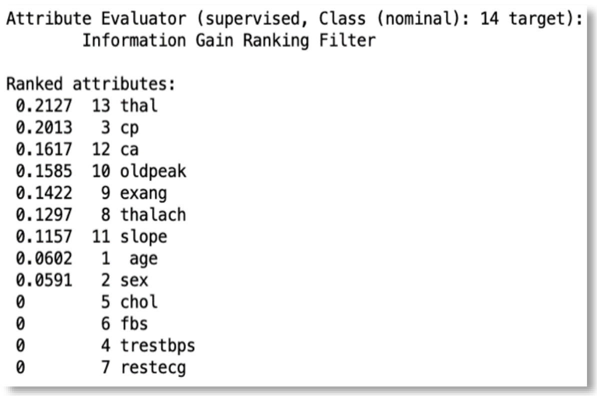
Table 6. Error rate by seed value in RP

Seed	10	20	30	40	50
Error (%)	27.7228	39.934	38.2838	38.9439	35.6436

Because of its random nature, RP returns different results based on what random seed it uses. This means that the result in Table 5 is not concrete, since it varies by the seed value.

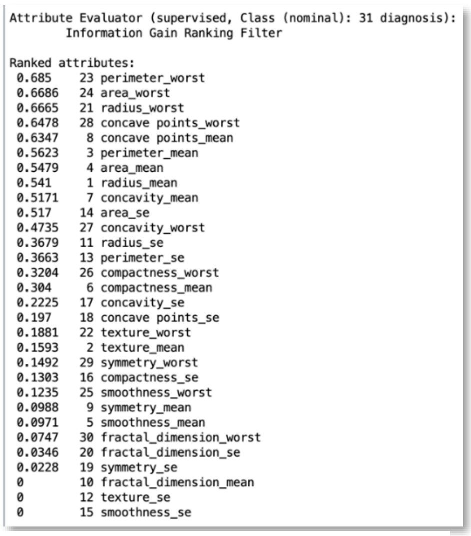
Information Gain (IG)

Unlike the preceding dimensionality reduction algorithms which are feature transformation algorithms, information gain is a feature selection algorithm. Feature selection algorithms strive to solve the curse of dimensionality by reducing features selected.



```
Attribute Evaluator (supervised, Class (nominal): 14 target):
        Information Gain Ranking Filter

Ranked attributes:
 0.2127  13 thal
 0.2013   3 cp
 0.1617  12 ca
 0.1585  10 oldpeak
 0.1422   9 exang
 0.1297   8 thalach
 0.1157  11 slope
 0.0602   1  age
 0.0591   2 sex
 0        5 chol
 0        6 fbs
 0        4 trestbps
 0        7 restecg
```



```
Attribute Evaluator (supervised, Class (nominal): 31 diagnosis):
        Information Gain Ranking Filter

Ranked attributes:
 0.685    23 perimeter_worst
 0.6686   24 area_worst
 0.6665   21 radius_worst
 0.6478   28 concave points_worst
 0.6347    8 concave points_mean
 0.5623    3 perimeter_mean
 0.5479    4 area_mean
 0.541     1 radius_mean
 0.5171    7 concavity_mean
 0.517    14 area_se
 0.4735   27 concavity_worst
 0.3679   11 radius_se
 0.3663   13 perimeter_se
 0.3204   26 compactness_worst
 0.304     6 compactness_mean
 0.2225   17 concavity_se
 0.197    18 concave points_se
 0.1881   22 texture_worst
 0.1593    2 texture_mean
 0.1492   29 symmetry_worst
 0.1303   16 compactness_se
 0.1235   25 smoothness_worst
 0.0988    9 symmetry_mean
 0.0971    5 smoothness_mean
 0.0747   30 fractal_dimension_worst
 0.0346   20 fractal_dimension_se
 0.0228   19 symmetry_se
 0        10 fractal_dimension_mean
 0        12 texture_se
 0        15 smoothness_se
```

Figure 24. From left to right: Output of Information Gain feature selection run on the Heart Disease and Breast Cancer dataset. Attributes are ranked based on information gain.

Running information gain attribute evaluation on WEKA returns a ranked list of attributes in order of information gain (entropy). Information gain represents how impactful each attribute is to determine the class. Figure 24 is the outputs after running the InfoGainAttributeEval attribute evaluator in WEKA on the datasets. The higher the information gain, the more that attribute contributes to class prediction. One can see that there are features with zero or very small information gain. For feature selection, I aimed to reduce the number of features to about half and set a threshold of the entropy based on that. For the Heart Disease, I picked features with information gain above 0.1, and above 0.3 for the other dataset.

Table 7. Percentage of incorrectly clustered instances for datasets feature reduced with IG

%	Heart Disease	Breast Cancer
k-means	28.0528	7.2056
EM	27.0627	6.6784

Table 7 shows the result of the reduced features run on the two clustering algorithms. Interestingly, using information gain gave mostly slightly better results. This was an expected outcome because information gain discard features with zero or low entropy, leading to better clustering. Below are the clustering visualizations for k-means clustering on both datasets. EM clustering returned very similar results.

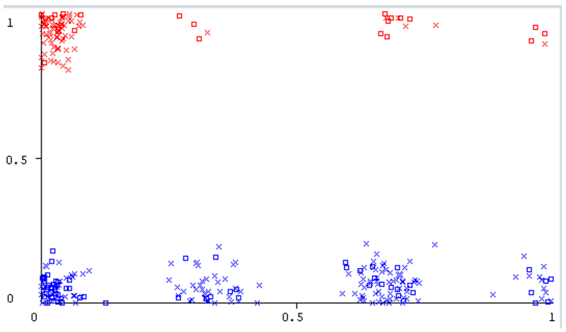


Figure 25. IG-Kmeans cluster for Heart Disease dataset.

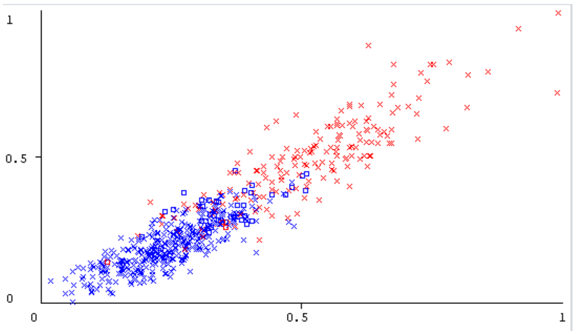


Figure 26. IG-Kmeans cluster for Breast Cancer dataset.

The clustering for the Breast Cancer dataset looks very similar to k-means clustering of the unmodified data. This makes sense because IG does not reconstruct features like the other three DR algorithms used. Since IG algorithm cherry picked features with the highest information gain, the clustering converged more and became even more distinguishable compared to the original clustering in Figure 4. The Heart Disease clustering turned out a horizontally clustered result. Although Figure 25 is not considered a 'good' clustering, we can still separate two clusters if we draw a horizontal line in the middle of the y axis. As expected, the target-cluster plot returned better results with the clusters more in line with the targets. Overall, using information gain gave more confidence to the selected features.

3. Neural Network

After applying dimensionality reduction algorithms in part 2, I applied the newly projected data on the neural network learner on the Heart Disease dataset. Then, I compared the accuracy of each algorithms to the NN accuracy from assignment 1.

Table 8. Results of neural networks run on newly created features from DR algorithms.

Dimensionality Reduction	NN Accuracy (%)	Running Time (s)
None	93.442	0.091
PCA	91.803	0.105
ICA	86.885	0.084
RP	85.245	0.077
IG	95.081	0.098

One can see that neural network run on four algorithms returned acceptable accuracies that were not far from the accuracy obtained from NN run on the unmodified dataset. One interesting observation is that information gain scored the highest accuracy, higher than that of the unmodified dataset. This is an expected result since IG only used attributes with high entropy, leading to an ease in classification.

Applying the three feature transformation algorithms reduced the overall accuracy, but it was within an acceptable range with the default accuracy. PCA and ICA had the longest running times. PCA performed best among the three. I think this is because PCA preserves the data compared to other algorithms. Another observation is that RP obtained similar accuracy as the other algorithms with a shorter running time. This pattern was also discovered in the next batch of NN results.

Next, I ran NN on the five dimensionality reduction algorithms, this time with the clusters added as an extra feature. Below are the results of NN. For a clear comparison, the number of clusters for k-means and EM were set to 2 for all cases.

Table 9,10. From left to right: results of neural networks for DR algorithms with k-means (left) and EM (right) clusters added as a feature.

Dimensionality Reduction	NN Accuracy (%)	Running Time (s)
None	91.803	0.092
PCA	93.442	0.164
ICA	93.442	0.105
RP	88.525	0.083
IG	91.803	0.084

Dimensionality Reduction	NN Accuracy (%)	Running Time (s)
None	93.442	0.097
PCA	90.164	0.091
ICA	90.164	0.104
RP	85.245	0.078
IG	86.885	0.075

Overall, adding clustering to the feature improved accuracy for most of the cases with k-means, but decreased for EM. I suspected this happened because EM created unclear clusters, thus decreasing the chance of accurate classification. For k-means, IG obtained the same accuracy as default with reduced running time. PCA and ICA had the same improved accuracies, with PCA having the longest running time. While EM clustering was not a good feature, adding k-means as a new feature led to a more accurate classification. One interesting point was that IG had similar or better results with a relatively fast runtime. Applying feature transformation algorithms did not fare well in classification accuracy and running time for this dataset.