P2 Report - Visualization and Dimensionality Reduction

1. Introduction

2. Chosen Datasets

[One Paragraph for each dataset]

Vote dataset is composed by 435 samples and 17 features. All the features are categorical and binary. As all of its variable are categorical, this analysis is performed using One Hot Encoding. This is done to avoid the problem of ordinality.

3. Analyze PCA implementation

In this section we will explore the Vote dataset with the PCA algortihm.

3.1. PCA Developed in this work

In this section, we will explore the results obtained with the implmenetation of our PCA.

Given that this transformations are linear and based on the covariance matrix, no real world understanding can be extracted, at first glance, from the new coordinates. An example can be found in Table 1.

	PC1	PC2	PC3	PC4	PC5
0	-2.538132	0.208806	-0.664641	-0.174588	0.450081
1	-2.566310	0.385575	0.730238	0.070440	0.545899
2	-1.181888	1.863804	-0.193548	-0.056694	-0.517372

Table 1: First 3 samples for vote dataset in the new PC space using our implementation of PCA

Given that the data is implicitly centered but not standardized, the mean of the data is zero but its variance is not one. It doesn't affect the results but it is important to take into account when analyzing the results. As can be observed in the following table, the mean of the data is zero but the variance is not one.

	PC1	PC2	PC3	PC4	PC5
mean	1.429253e-16	-1.250596e-16	6.061562e-17	2.261760e-16	-2.779386e-16
std	1.899175e+00	8.106390e-01	7.187142e-01	6.343247e-01	5.914352e-01
min	-2.674723e+00	-1.830662e+00	-1.672052e+00	-1.657765e+00	-2.113420e+00
max	2.520503e+00	2.093992e+00	1.432702e+00	1.671510e+00	1.636564e+00

Table 2: Transformed vote dataset description

It's also interesting to analyze the loadings of the transformation. The loadings are the eigenvectors of the covariance matrix. They represent the direction of the maximum variance. In Table 3 an example is shown.

Feature	PC1	PC2	PC3	PC4	PC5
handicapped-infants_n	0.256762	-0.093263	0.088130	0.325191	-0.187823
handicapped-infants_y	-0.247978	0.092984	-0.092867	-0.325534	0.188177
water-project-cost-sharing_n	-0.060097	-0.332781	-0.013825	0.126901	0.282068

Table 3: Loadings for vote dataset

The results for the handmade PCA projection on the three datasets are shown in the following figures. The first two components are plotted in order to visualize the data in a 2D space. The first component is the one that separates the classes the most. The second component separates the classes but not as much as the first one.

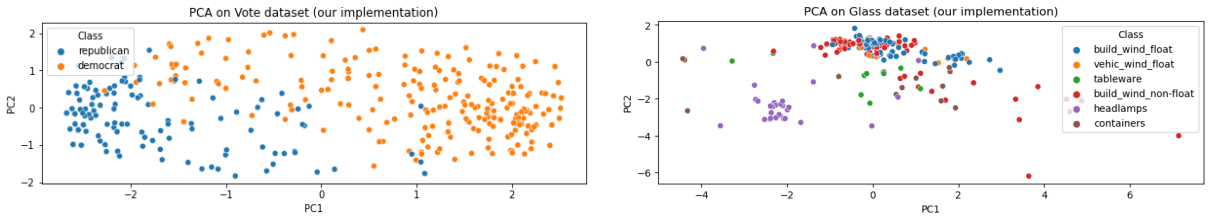


Figure 1: Projection results handmade PCA for vote and glass datasets

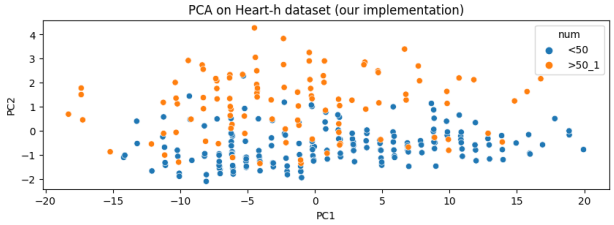


Figure 3: Projection results handmade PCA for heart-h dataset

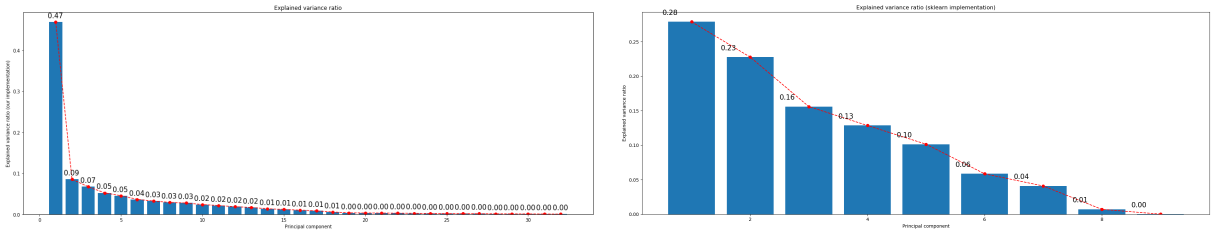


Figure 2: Relative Variance Explained handmade PCA for vote and glass datasets

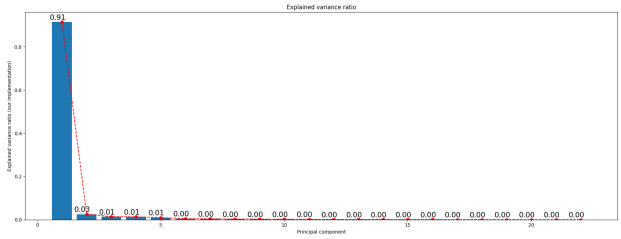


Figure 3: Projection results handmade PCA for heart-h dataset

It's noticeable that, in vote dataset, the half of the Principal Components don't explain anything. The reason is that the data is categorical and binary and in our One Hot Encoding, there is a column for Yes and another one for No. This causes that each pair of features are linearly dependent and hence, they don't provide any information, which is represented as a zero variance in the PC space. In Table 3 can be observed that the "No" class loadings are the opposite of the "Yes" class loadings.

In the case of glass data, we can see that the the variance is slowly reduced until it reaches 0, where the ninth principal component does not explain any of the variance in the graph. The eight component corresponds to around 4% **no seria 7??** of the variance that is explained in the first component.

And in the case of heart-h dataset, we can see that the first principal component is the one that explains the major part of the variance, with a proportion of the 0.91 of the variance. This is similar to what happens in the vote dataset, altough now the difference between the first and the second component is higher. This results could be attributed to several factors in the preprocessing pipeline when dealing with different types of variables, such as standardizing numerical features, imputing missing values with the median, and encoding categorical variables.

In the following sections, we can, therefore, see the contrast between variance in a categorical dataset (such as vote), a continuous-variables dataset (glass) and a dataset containing both categorical and continuous-variables (heart-h).

3.2. Sklearn PCA

When Scikit-Learn's implementation is applied to our dataset, the results are the same respect to the coordinates in our new PC space. This happens independently of the datasets.

	PC1	PC2	PC3	PC4	PC5
0	2.538132	0.208806	0.664641	-0.174588	-0.450081
1	2.566310	0.385575	-0.730238	0.070440	-0.545899
2	1.181888	1.863804	0.193548	-0.056694	0.517372

Table 4: First 3 samples for vote dataset in the new PC space using sklearn PCA

The only difference found, is that the signs are not the same in some of the components. This is because the only restrictions are in the orthogonality and unitarity of the components. But there is freedom in the choice of the sign of the components.

The explained variance results are also exactly the same in all the cases, given that the same one-shot algorithm is used.

3.3. Sklearn IncrementalPCA

The PCA object proves to be beneficial but exhibits limitations when dealing with large datasets. Its primary drawback is its exclusive support for batch processing, necessitating that all data fit into main memory. In contrast, the IncrementalPCA object offers an alternative processing approach, enabling partial computations that closely align with PCA results while handling data in a minibatch manner.

	PC1	PC2	PC3	PC4	PC5
0	2.538132	0.208806	0.664641	0.174588	-0.450081
1	2.566310	0.385575	-0.730238	-0.070440	-0.545899
2	1.181888	1.863804	0.193548	0.056694	0.517372

Table 5: First 3 samples for vote dataset in the new PC space using IncrementalPCA

The main difference is performance in time and memory. The Incremental PCA is much faster for big datasets than the PCA and it uses less memory. However, the results are not exactly the same. The explained variance ratio is the same but the components are not. This is because the IncrementalPCA is an approximation of the PCA.

What can be observed from the following comparison is that, when batch size is increased, IPCA is faster and uses less memory. However, in this case, approximation error from IPCA is negligible. This is observed from the difference in two first explained variance ratio and the average difference in components. The approximation error is not noticeable.

In the other hand, it doesn't either speed up the process. This is because the dataset is not big enough to notice the difference. For glass dataset, some erratic performance is observed for certain batch sizes. The reasons are, probably, numerical inestability and the heterogeneity of the batches (i.e. two batches can contain very different instances).

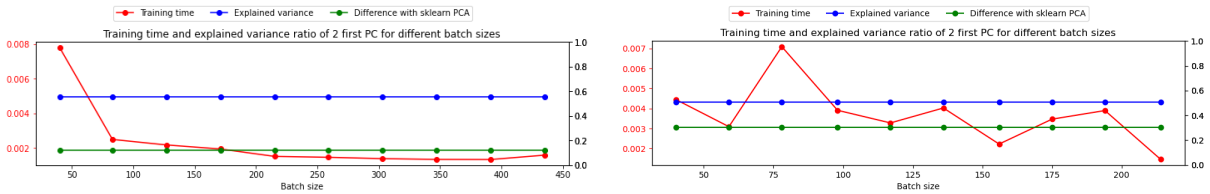


Figure 3: Training time (left axis) for different batch sizes together with explained variance (blue) and average difference between components wrt PCA (green) for vote and glass datasets

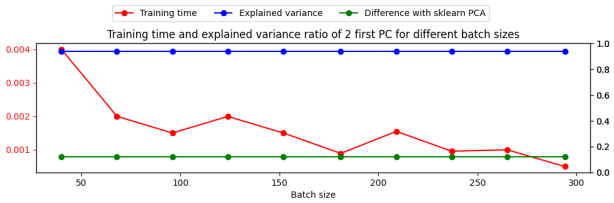


Figure 3: Projection results handmade PCA for heart-h dataset

For a `batch_size` equal to the number of samples, the results are the same as the PCA. This is because the algorithm is the same. The visible difference in the plots can be due to external factors such as the random initialization of the algorithm.

As an illustrative example, a comparison in time performance has been done with vote dataset. Our implementation of PCA is faster than the incremental one until `batch_size` of 260. However, when `batch_size` is increased, the IPCA is faster as it is expected given that sklearn PCA is faster than ours.

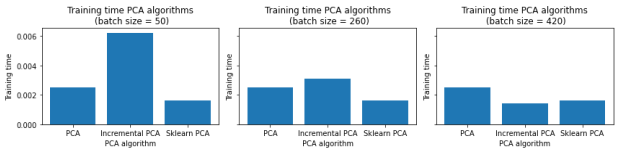


Figure 4: Comparison in training time for vote dataset

From these figures we can conclude that there isn't a significant change in the current variance explained of each component. However, we know that the IncrementalPCA is designed to be a more efficient form of PCA: losing some accuracy to in change have better memory and temporal usage. This is specifically useful for larger datasets, we can see that the training time in the incremental PCA starts to be noticable very early and around the 420 batch we can see a large difference between our PCA, the incremental PCA and the Sklearn PCA. Indicating, that the larger we make the batch size the greater difference we will see between the computational times in the different versions of the PCA.

4. Use PCA with k-Means and BIRCH to compare performances

ESTA BIEN AKI?? In this section, our own implementation of PCA and k-Means, together with sklearn's BIRCH, are used in order to cluster the data.

As shown in Figure 1, for vote dataset after projecting the data into the 2 first Principal Components, the classes are quite well separated. A lot of variables weren't providing almost any information in terms of variance. This causes the well known dimensionality curse, in which when the number of features is increased, the performance of the algorithm decreases. This is why PCA is used, to reduce the number of features and hence, the dimensionality of the data.

In the case of glass dataset, only using the first 2 components it's not enough to make the classes separable given the complexity of the problem. This can be infered as

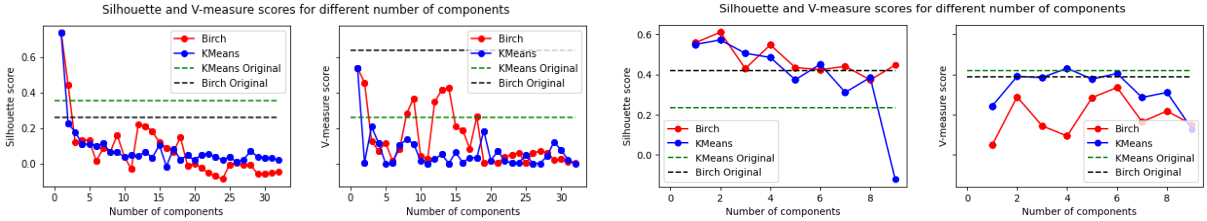


Figure 5: Performance comparison for clustering using different number of components for vote and glass datasets

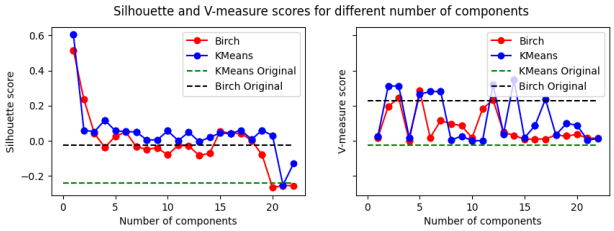


Figure 5: Performance comparison for clustering using different number of components for heart-h datasets

Vote dataset

As can be infered from the previous plot, in every case, incrementing the number of components even adding more information, leads to equal or worse results. This is because of the dimensionality curse. The more features, the worse the performance of the algorithm. In the case of silhouette score, a model trained only with the first component has far better value than in the case of the model trained without applying PCA. In the other hand, in terms of V-Measure, KMeans increases its performance when trained with just one component. However, Birch's performance slightly decreases. As it is an informed external metric, it is more sensitive to the information loss.

Another interesting fact is that both algorithms have exactly the same performance when trained with the first component.

	BIRCH	KMeans	Birch (PC1)	KMeans (PC1)
Silhouette	0.26	0.35	0.74	0.74
V-Measure	0.64	0.54	0.54	0.54

Table 6: Vote dataset. Performance using PCA and not using it.

Glass dataset

In the case of silhouette score, a model trained only with the first component has far better silhouette score than in the case of the model trained without applying PCA. On the other hand, in terms of V-Measure, both Birch and K-Means' performance slightly increase with 2 and 6 components, but decrease in other cases. KMeans obtains a maximum with 4 components whereas Birch obtains the maximum V-Measure score with 4 components.

	BIRCH	KMeans	Birch (PC1)	KMeans (PC1)
Silhouette	0.42	0.23	0.56	0.55
V-Measure	0.39	0.16	0.05	0.24

Table 7: Glass dataset. Performance using PCA and not using it.

Heart-h dataset

In Heart-h dataset, we can observe that the silohouette results are better when training a model with the first component after applying PCA for both KMeans and Birch. Getting optimal results with 1 component for both algorithms means that the information loss is not significant, as PC1 is able to capture most of the variance. However, Birch performance in terms of V-Measure gets worse with this technique in comparison to the model trained without applying PCA, while KMeans performance does not improve either. Birch maximises its V-Measure score with 5 components whereas KMeans maximises its V-Measure score with 14 components.

	BIRCH	KMeans	Birch (PC1)	KMeans (PC1)

	BIRCH	KMeans	Birch (PC1)	KMeans (PC1)
Silhouette	-0.02	-0.24	0.51	0.55
V-Measure	0.23	0.01	0.02	0.01

Table 6: Vote dataset. Performance using PCA and not using it.

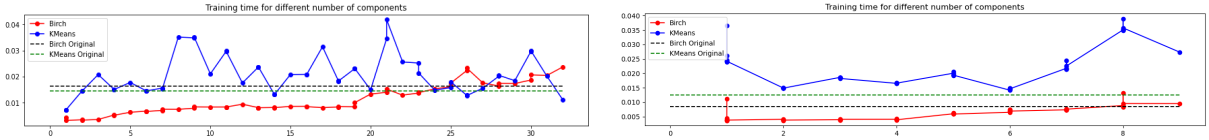


Figure 6: Performance in time comparison for clustering using different number of components for vote and glass datasets

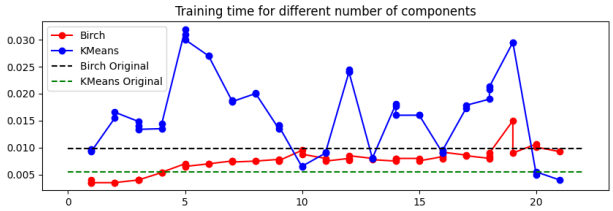


Figure 7: Performance comparison for clustering using different number of components for heart-h datasets

We can see in Figure 6 and 7 a slight improvement in the temporal capacity of the algorithms as well. Again we see that due to the size of the chosen datasets the temporal difference is not as significant as it would be with a larger dataset. Either way, we see that Birch shows promising results both in the training scores and the efficiency of the datasets.

5. Cluster the transformed Data (SVD) using K-Means and Birch

The Singular Value Decomposition (SVD) is a fundamental numerical linear algebra technique that decomposes a matrix into three other matrices. Given a matrix A of dimensions $m \times n$, the SVD can be represented as:

$$A = U\Sigma V^T$$

where:

- U is an $m \times m$ orthogonal matrix (meaning $(U^T U = I)$, where I is the identity matrix).
- Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal, known as the singular values. The singular values are arranged in descending order.
- V is an $n \times n$ orthogonal matrix.

The SVD is widely used in various applications, such as dimensionality reduction, noise reduction, and solving linear equations. In the context of machine learning, SVD is often used in techniques like Principal Component Analysis (PCA) for dimensionality reduction and collaborative filtering in recommendation systems.

If SVD is directly applied to the data, the behavior is strange. The first singular value is the lowest one and the rest are sorted from higher to lower magnitude.

	SV1	SV2	SV3	SV4	SV5
0	2.706905	-2.462044	0.216076	0.663765	-0.177440
1	2.929801	-2.476804	0.409878	-0.735057	0.086129
2	2.464877	-1.118496	1.868968	0.186572	-0.044793

Table 8: Vote dataset. First 5 Singular Components.

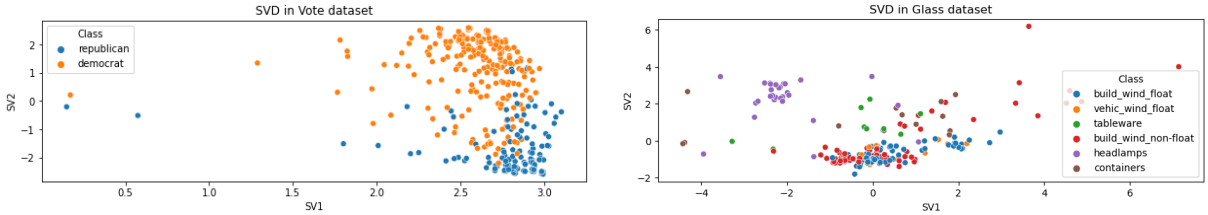


Figure 7: Performance in time comparison for clustering using different number of components for vote and glass datasets

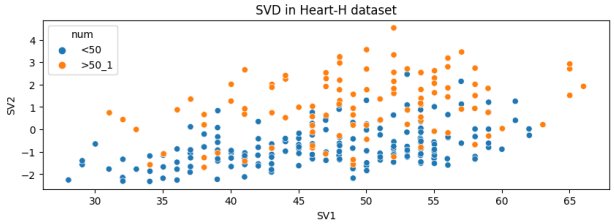


Figure 5: Performance comparison for clustering using different number of components for heart-h datasets

The results in case of vote and heart-h datasets are quite different from the ones obtained from PCA. After running some other experiments, it's been noticed that this effect is due to the scale of the features. In fact, when Standard Scaler is applied, the results are virtually the same as in PCA (except sign). For glass dataset, the results are the same as in sklearn's PCA (the same as in Figure 1 except change in signs).

a que imagen se refiere The distribution of the explained variance ratio by singular values is similar to the one we had for PCA but in this case, the first singular value is the lowest one, being the first one 0.01, the second one 0.47, the third one 0.08 and the rest are decreasing in magnitude. This can be due to this scaling issue. In the case of glass dataaset, the distribution is similar to the one obtained in PCA for explained variance ratio.

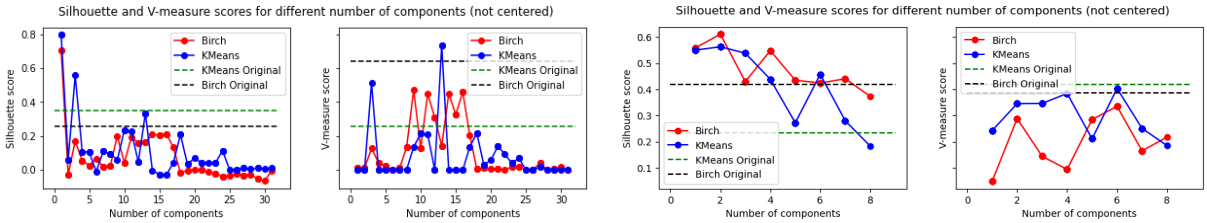


Figure 8: Performance in clustering using different number of components for vote and glass datasets and SVM

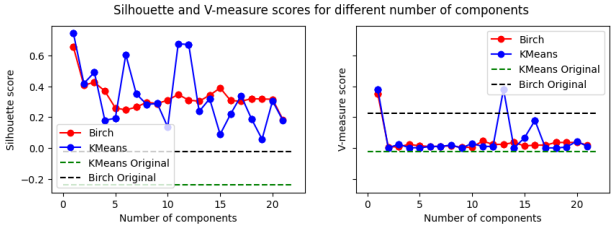


Figure 5: Performance comparison for clustering using different number of components for heart-h datasets

6. Visualize in low-dimensional space

In this subsection, original data sets are visualized together with the result of the k-Means and BIRCH algorithms without the dimensionality reduction, and the result of the k-Means and BIRCH algorithms with the dimensionality reduction. To visualize in a lowdimensional space (2D or 3D) PCA and ISOMAP approaches are used.

Isomap clustering is a dimensionality reduction and clustering algorithm that can be used to visualize and cluster high-dimensional data. It works by constructing a graph of the data points, where the edges of the graph represent the distances between the data points. The algorithm then finds a lower-dimensional embedding of the data points, while preserving the distances between the data points as much as possible.

Vote dataset

Given that the data is categorical, the visualization of its 2 first components is not very useful. Until now, 2 first linear projections have been used in order to visualize the data, such as SVD and PCA. The result is that they both provide similar information.

In this case, Self Organized Maps are used in order to have a nonlinear point of view, i.e. to use a different approach.

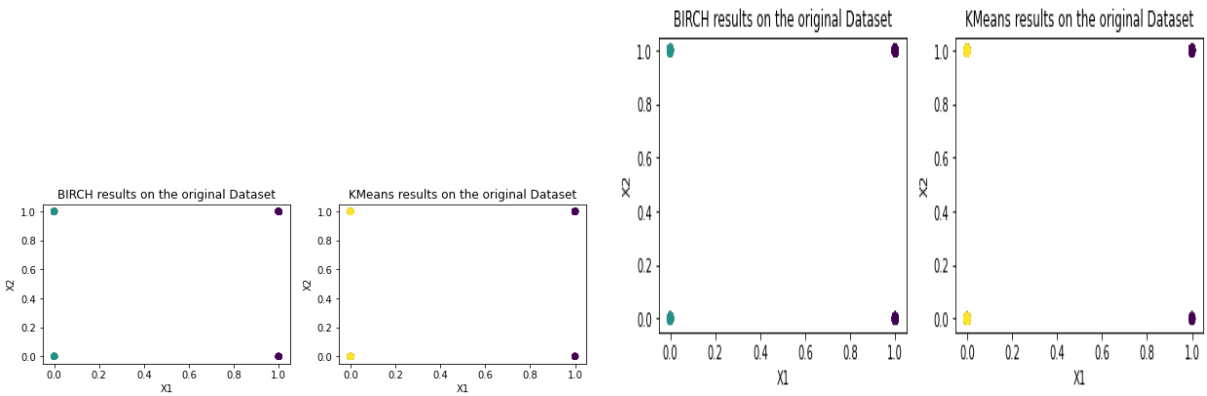
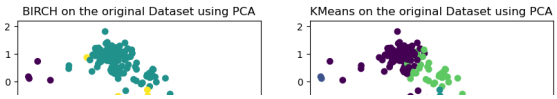


Figure 9: Vote dataset. Visualization of projection using SOM (Isomap) and PCA

In this case, both algorithms provide similar information. This can be due to the fact that the data is not very complex and hence, the linear projection is enough to visualize the data.

Glass dataset



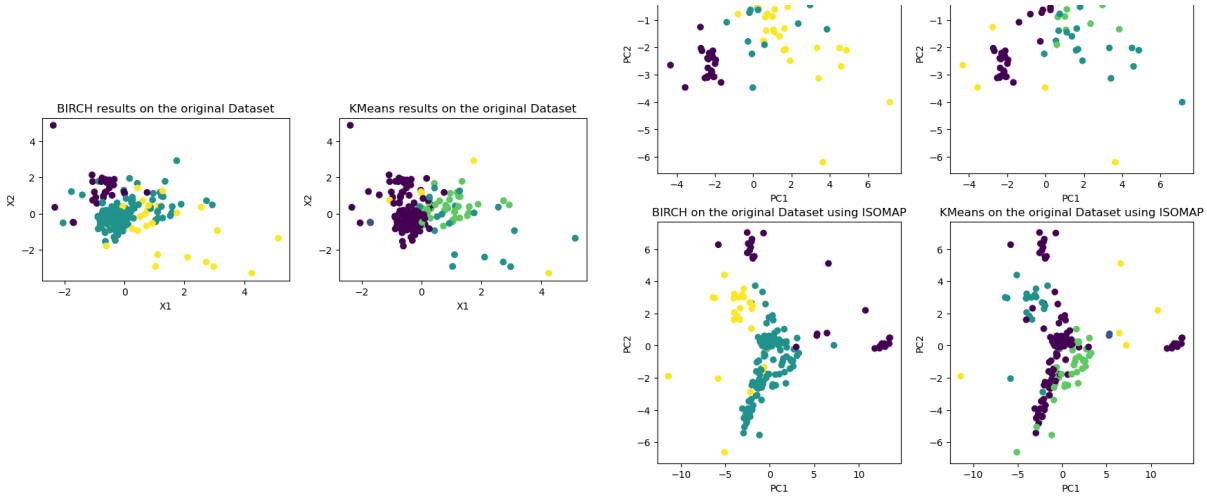


Figure 10: Glass dataset. Visualization of projection using SOM (Isomap) and PCA

The scatter plot in the image shows the lower-dimensional embedding of the Glass dataset using isomap. The data has been clustered using two different algorithms, BIRCH and KMeans. The results show that both algorithms are able to cluster the data into three distinct groups.

Heart-h dataset

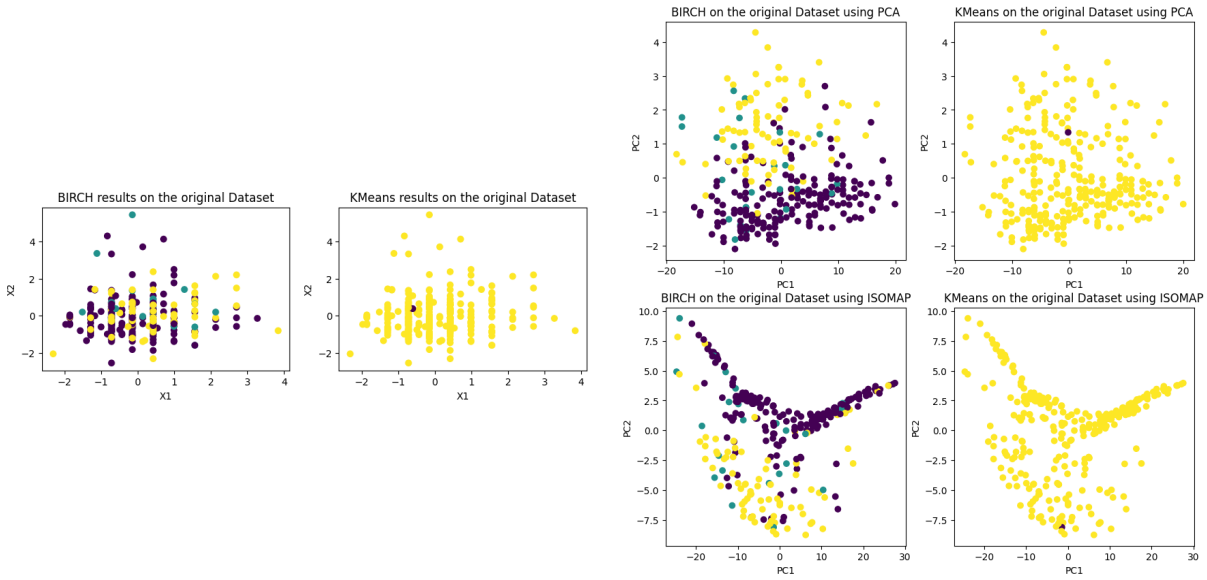


Figure 10: Glass dataset. Visualization of projection using SOM (Isomap) and PCA

The scatter plot in the image shows the lower-dimensional embedding of the Glass dataset using isomap. The data has been clustered using two different algorithms, BIRCH and KMeans. The results show that both algorithms are able to cluster the data into three distinct groups.