Work 2: Dimensionality reduction

Introduction to Machine Learning, 2021-2022

Augusto Moran Calderon Santiago del Rey Juárez

Yazmina Zurita Martel

November 2021

1 Validation metrics

To evaluate the quality of the clusterings obtained from the application of the different dimensionality reduction algorithms used in the project, we have used a total of four validation metrics.

1.1 Internal validation

To validate our findings independently of the ground truth labels, we use two validation metrics which are:

Calinski-Harabasz index (CH) This index is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters, where a high score relates to a model with better-defined clusters [1].

Davies-Bouldin index (DBI) This index signifies the average 'similarity' between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves. Zero is its lowest possible score, and values closer to it indicate a better partition [1].

1.2 External validation

To validate our findings compared to the ground truth labels, we use two validation metrics which are:

Adjusted Mutual Information score (AMI) The AMI score is a normalized version of the Mutual Information function that measures the agreement of the two assignments, ignoring permutations. It ranges from 0 to 1. Perfect labelling is scored 1 [1].

Fowlkes-Mallows score (FMI) The FMI score is defined as the geometric mean of the pairwise precision and recall. The score ranges from 0 to 1. A high value indicates a good similarity between two clusters [1].

2 PCA

In this section, we briefly describe our implementation of the PCA algorithm [2]. Then, we describe the results obtained from applying our implementation of the PCA algorithm over the three datasets used in the previous project. Also, we compare these results with the ones obtained from using the PCA and Incremental PCA implementations of the sklearn library [3].

2.1 Implementation

For the implementation of the Principal Component Analysis algorithm, we will heavily rely on matrix multiplication and methods from the numpy library. First, we split this process into 5 steps with some additional methods that provide useful information for the user such as retrieving the covariance matrix, eigenvalues, etc.

The first step is to center the data by subtracting its mean and calculate its covariance matrix, which corresponds to the extent to which two dimensions moving in the same direction. This

means we calculate the covariance for each pair of dimensions/attributes in the dataset. When the covariance of the same variable is computed we are actually retrieving the variance which measures how spread is the dataset with respect to that dimension.

The second step is to compute the eigenvalues and eigenvectors. Next, we order the eigenvectors with respect to the eigenvalues in descending order. Now we have our most significant dimension which accounts for most of the variability in the first column, then the second in the second column and so on. After that we pick the first k eigenvectors, this will be our new dimensions for the dataset. Finally, we transformed the n x m dataset into k x n dataset where k should be lower than n using the matrix of k x m eigenvectors.

Once the PCA is concluded, we reconstruct the original dataset from the transformed data by inverting the operations applied till now. That is, by multiplying the transformed data by the transpose of the eigenvectors matrix and adding the mean previously calculated. In the case of the categorical dataset, we rounded the result to the closest integer and calculated the number of mismatches with respect to the original dataset.

In addition, we implemented a method to compute how much of the original variance is captured by each of the components (explained variance ratio). It can be used as a criterion to choose the number of components later on.

2.2 Results

Heart-C

In order to select the most relevant features in the Heart-C dataset [4], we used three different implementations of the PCA algorithm.

First, we used the implementation of the sklearn python library. The main reason to start with this implementation was the possibility of choosing automatically the number of components by using the method described by Minka in [5]. This resulted in a reduction of 7 dimensions, from 26 to 19. Once we found the number of components that kept almost all the variance in the data (99%) we proceeded to use the Incremental PCA and our PCA with the same number of components.

Our first thoughts were that with this number of components we would achieve the best results when using K-means. However, we noticed that the results of the validation were the same for the non-reduced dataset as for the reduced one. We guess that this happens because the new dataset is almost identical to the old one with respect to the distribution in the space. Thus, we may still have features that are not relevant enough in the data and that are inducing the algorithm to detect bad clusterings. This is also corroborated by the fact that all the eigenvalues obtained from our implementation have values smaller than one except for the first component.

To solve this problem we decided to reduce the number of components since augmenting it would not make sense. After several tries, we found that 7 components, which represent 72.5% of the original variance, achieved good enough results in the K-means.

To compare the three different implementations of the PCA, we used the same number of components. In Figure 1 we can see the results obtained from the different implementations. In blue we observe the samples belonging to the true class '<50' and in orange the ones of the true class '>50_1'. As seen in the figure the three obtain rather similar results. If we compare them with the original data, we see how they acquired a more rounded and spread distribution in the space. Also, we noticed that the `sklearn` PCA results got a slightly compact dataset, which might be due to small differences in the implementation, e.g. computation optimizations. Surprisingly, if we carefully look at the results we observe how our implementation returns almost the same results as the `sklearn` PCA with the difference that it is inverted in the Y-axis. Also, we can see that the

same happens with the Incremental PCA, but this time the inversion occurs on the X-axis.

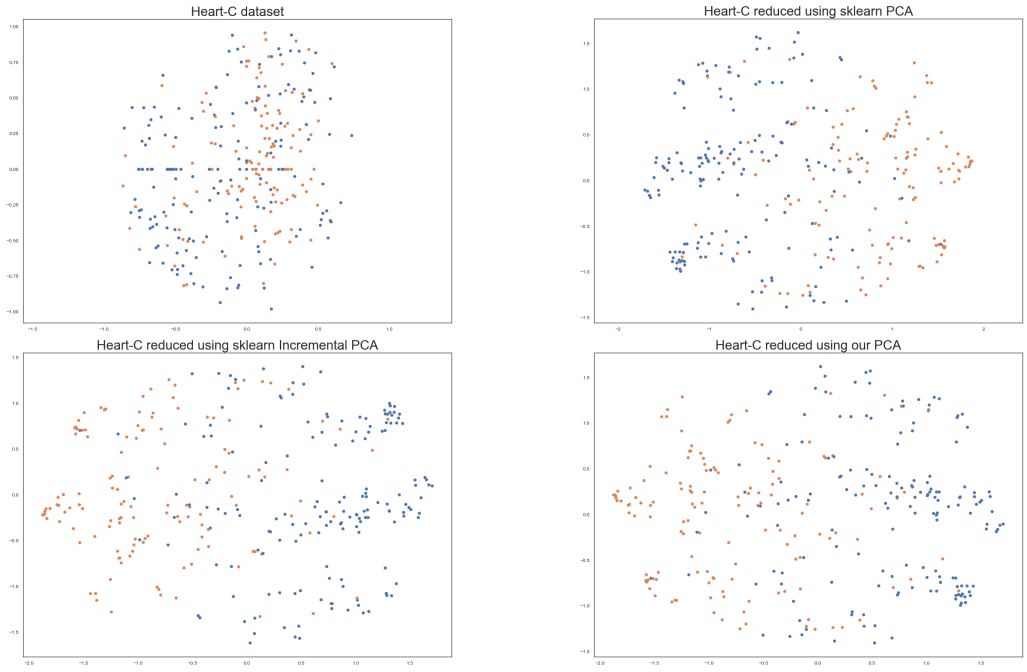

Figure 1: Results of applying different PCA implementations to Heart-C.

Connect-4

The objective of applying PCA to our datasets was twofold. On the one hand, we wanted to know if visualising the data under a new representation allowed us to extract information about its structure. On the other hand, we tested whether this projection of the encoded Connect-4 dataset [6] would improve previous results of its K-Means clustering or at least reduce the computation time needed.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Variance ratio	0.1242	0.1121	0.1004	0.0904	0.0793	0.0632	0.0410	0.0372
Cumulative variance ratio	0.1242	0.2362	0.3366	0.427	0.5064	0.5696	0.6106	0.6478
	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16
Variance ratio	0.0359	0.0324	0.0300	0.0279	0.0242	0.0224	0.0214	0.0209
Cumulative variance ratio	0.6837	0.7161	0.7461	0.7740	0.7982	0.8207	0.8421	0.8630

Table 1: Variance of Connect-4 captured by each principal component.

First, we need to decide the number of components to keep, so we compared the results of the clustering using the 2, 5, 10 and 16 first principal components and visualised different pairs. We retrieved a slightly better result in the clustering using 16 PCs, although the difference was not significant. Visualising higher order components did not help us interpret the dataset better either. We finally followed the standard practice, which is to keep the lowest number of components needed to explain between the 85% and 95% variance in the data. Under this criteria, we kept the 16 first components which explain around 86% of the variance (Table 1). This implies a reduction of 26

dimensions.

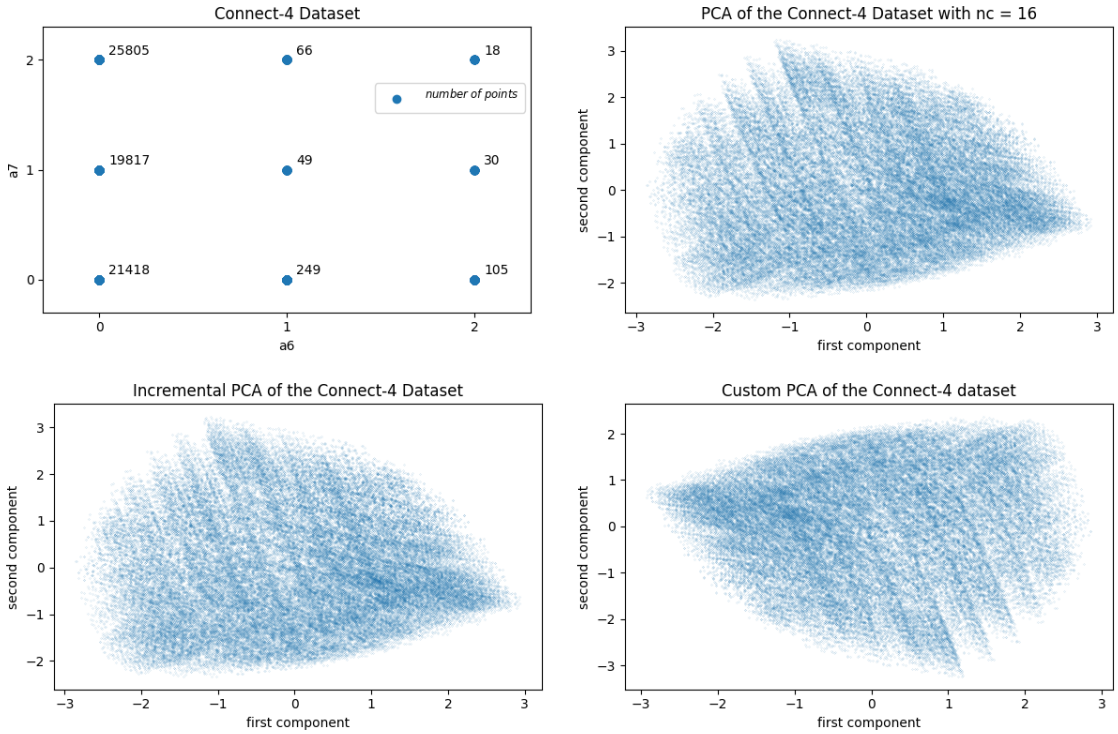

Figure 2: Results of applying different PCA implementations to Connect-4.

We also used different implementations of PCA: `sklearn`'s PCA and Incremental PCA and our own. They all returned very similar projections as can be seen in Figure 2. The main difference is that our custom implementation retrieves the same structure but inverted in the Y-axis.

Comparing with the original dataset, the new representation confirms what we already suspected rather than provide new information. Since Connect-4 is a categorical dataset, we expected a cube-like structure as the one shown using PCA. In addition, we can see that some areas are denser than others in the original Connect-4. This can be seen in the reduction too, where a corner of this cube-like form shows a higher number of points than in other areas.

Pen-Based

First, it is important to select a particular number for dimensionality reduction. Sadly, there is no best method to obtain it. We select the Scree method which finds the variance explained by each component by dividing each component's eigenvalue by the sum of all eigenvalues. It states that the number of dimensions to be selected finishes when the curve starts to level off. In this case 6, which accounts for more than 95% of the variance of the Pen-Based dataset [7].

After running the PCA and Incremental PCA from the `sklearn` library and our implementation of PCA we got results shown in Figure 3. Looking at the results, we conclude that PCA algorithms provide a better representation of the dataset where we can actually see the insights and have a better understanding of the distribution of the data as we can appreciate in the image above. Finally as discussed before for the other databases our implementation, Incremental PCA and PCA outputs almost similar representations but somehow inverted.

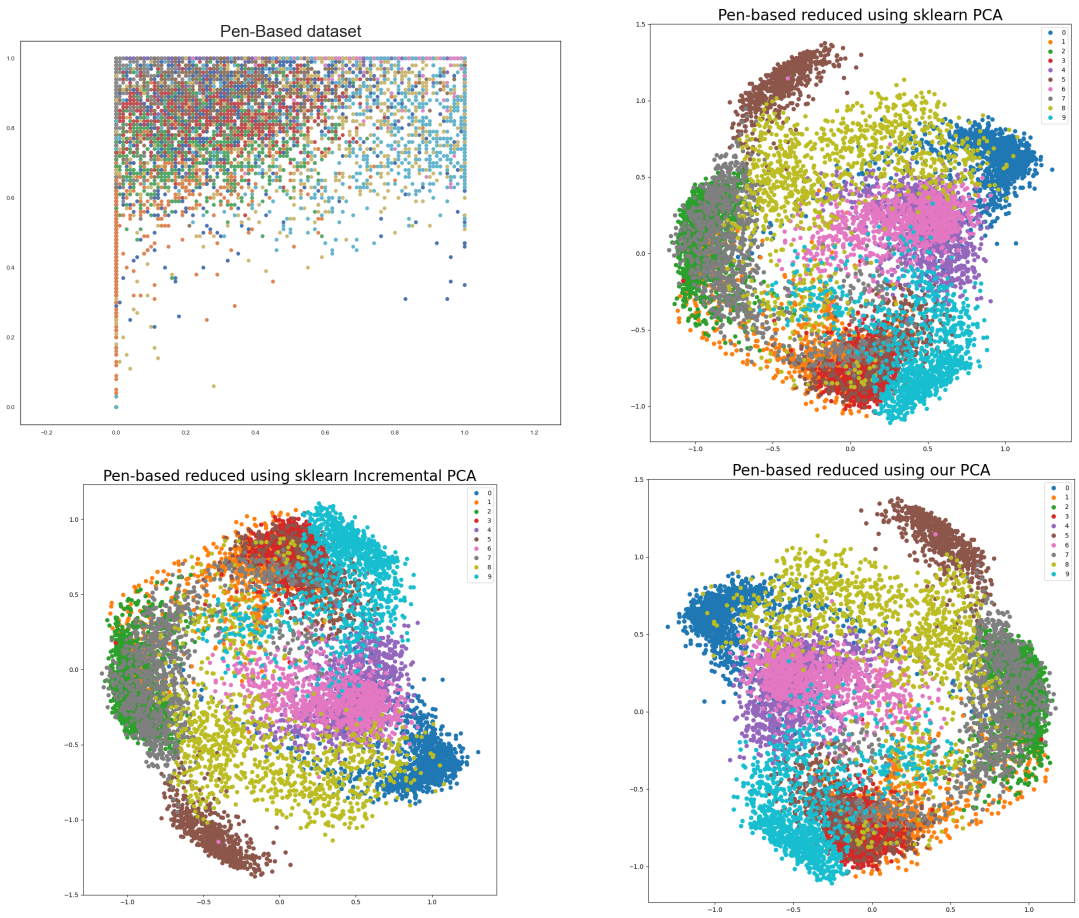

Figure 3: Results of applying different PCA implementations to Pen-Based.

3 UMAP

Uniform Manifold Approximation and Projection (UMAP) is an unsupervised non-linear dimensionality reduction and data visualization technique similar to t-SNE proposed in 2020 by McInnes *et al.*

In this section, we describe the configurations used for each one of the datasets and our results after applying this algorithm.

3.1 Heart-C

Depending on the goal of the dimensionality reduction we may choose a different configuration for the hyperparameters of the UMAP algorithm. Since our objective was to cluster the data after the reduction we followed the guidelines introduced in UMAP's documentation [9]. Thus, we set the minimum distance to 0 since we want to get cleaner separation across clusters, and tried different values for the number of components and the number of neighbours.

To select the final configuration we validated the results after applying the K-means and observing how good our clusters were. First, we fixed the number of components to the same number used with the PCA, i.e. seven, since we wanted to compare these results with the ones obtained using PCA. Second, we tried different numbers of neighbours, specifically: 8, 15, 20, 30, and 40. Although the results we obtained did not change greatly from one another, the one that gave us

better results was 40. Third, we tried to change the number of components to see if the results improved. However, after testing with 7, 10, 12, and 19 components we observed that the quality of the cluster did not vary significantly. Thus, we decided to keep them to 7 in order to match the number of components extracted from PCA, as previously mentioned.

3.2 Connect-4

As we did with PCA, we tested several hyperparameters of UMAP in order to obtain the best results in clustering and visualisation. We fixed the minimum distance to 0 so points can be packed as tightly as possible and thus create better clusters. Then, we tried combinations of 30, 60 and 90 neighbours and 2, 10, 16, 20 and 30 number of components. The reason behind testing much higher numbers of neighbours than the default value is to focus on the global structure of the data rather than the local.

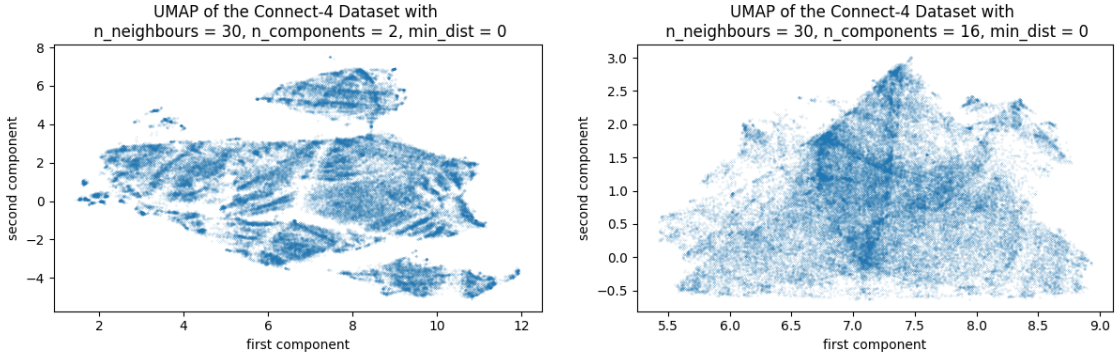

Figure 4: Results of applying UMAP on the Connect-4 dataset with different configurations.

For clustering purposes, the best choice of number of neighbours was 30. Since the number of components did not play an important role in the results, we decided to keep 16 as we did in PCA so we can compare between both methods later.

We tuned the hyperparameters again and tried to visualise partitions in the data. This was achieved setting the number of neighbours and components to 30 and 2 respectively as we can see in Figure 4. We observe different clouds of points in this representation so it could confirm that the data can be described by different classes. The number of classes is not clear since we can group the different partitions in several ways. Nonetheless, we should not jump to conclusions with this method because different configurations can yield very different maps that either confirm or contradict what we already know or suspect.

3.3 Pen-Based

Because we pursue the same objectives with the Pen-Based dataset than with the others, we apply the same execution steps established in the UMAP's documentation. Yielding to some fixed parameters such as: (min distance = 0).

After some iterations we conclude that the configuration that outputs the best results according to our metric of clustering representation were the one with number of components equal to 6 (matching with the one of PCA), and number of neighbours equal 40. It is important to note that for lower values of neighbours such as 20 and 30 the algorithm generates worse results. This final consideration was very clear because as the documentation states, it "will focus more on very local

structure and are more prone to producing fine grained cluster structure that may be more a result of patterns of noise in the data than actual clusters".

4 K-means

In this section, we explore the results obtained from applying our implementation of the K-means on the original datasets and after their reduction using PCA and UMAP.

4.1 Heart-C

To compare the results obtained from the previous project, which used the full dataset, with the results after applying PCA and UMAP, we will use the same configuration of parameters as the one achieving the best results in the mentioned work. The parameters are: (i) K = 2, (ii) init_method = k-means++, (iii) metric = euclidean.

In Figure 5, we can see the results of the K-means algorithm on the different modifications of the Heart-C dataset. In blue, we observe the samples corresponding to class 0, and in orange the ones of class 1. One clearly observable thing is how after the use of both PCA and UMAP, the 2D representation of the clusters is much more clear since we can easily differentiate the two clusters. In contrast, in the original dataset, it is hard to distinguish them due to its high dimensionality. Also, when comparing the data obtained with PCA with the one obtained from UMAP, we can see how the former one better defines the clusters and in the latter, the two clusters seem to be a little intermingled. This is probably due to the difference in how both algorithms work since PCA rotates the vectors for preserving the variance, while UMAP does not preserve the variance but the distance by minimizing it between the points in the original graph and the low dimensional graph. However, we cannot compare the two by only observing the visualizations.

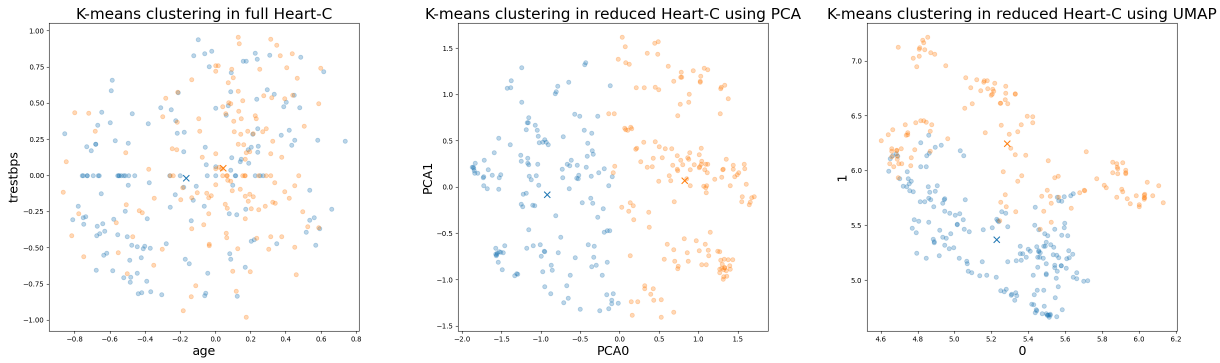

Figure 5: Results of applying K-means to different representations of the Heart-C dataset.

To assess the quality of the different clusterings and to better compare them, we have used the validation metrics presented in Section 1. As we expected, the results of the reduced datasets achieved better quality than the K-means on the full dataset (see Table 2). However, we have no clear winner on this occasion since the dataset with PCA reduction outperformed the one using UMAP with respect to the external validation. But, conversely, the dataset with UMAP reduction outperformed the one using PCA regarding the internal validation. We suspect that this is due to PCA trying to preserve the global structure of the data while UMAP tries to preserve the local structure.

	Full dataset	PCA reduction	UMAP reduction
Calinski-Harabasz index	65.0640	97.4639	437.2302
Davies-Bouldin index	2.1212	1.7214	0.8033
Adjusted Mutual Information score	0.3286	0.3356	0.2640
Fowlkes-Mallows score	0.7024	0.7111	0.6763

Table 2: Validation metrics for the different representations of Heart-C.

4.2 Connect-4

To compare the clustering of the different versions of Connect-4, we employed the same configuration of K-Means in the three cases: (i) K = 3, (ii) init_method = k-means++, (iii) metric = euclidean.

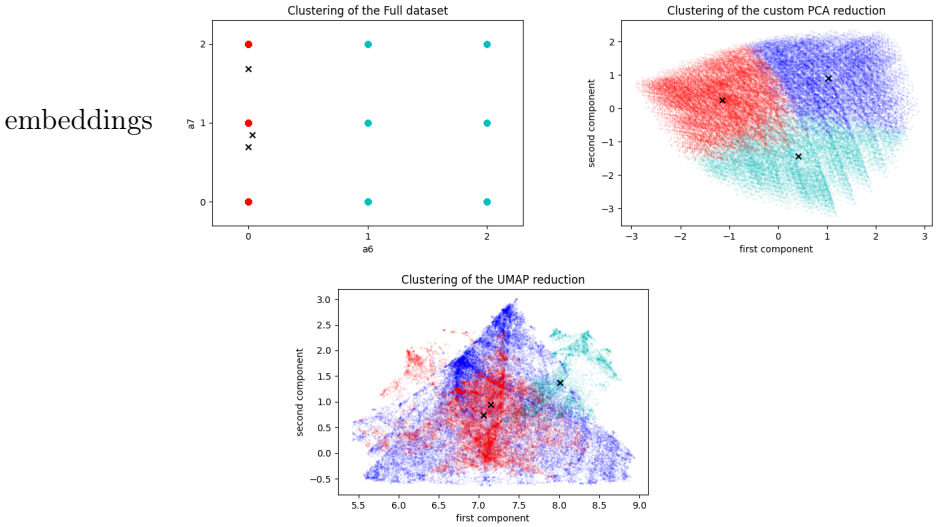

Figure 6: Results of applying K-means to different representations of the Connect-4 dataset.

We can visualise the results in Figure 6. It is impossible to see clusters when using a categorical dataset in its raw form since points overlap with each other in the small set of possible positions. This is where projections become very useful. We can locate the different clusters clearly in both the PCA and UMAP reduction. In the former, the less densely areas serve as boundaries of the partition. This is because PCA tries to preserve the global structure of the data so this becomes the only criteria to differentiate clusters. If we preserve the local structure instead as does UMAP, we lose the cube-like form of the dataset but retrieve a clustering that aligns better with our intentions. This can be further discussed if we look at the quantitative measures shown in Table 3.

We saw that the points are not as tightly packed in the UMAP reduction as in the other two cases, so the internal quality of the clusters is considered lower as the Calinski-Harabasz index shows. As a trade-off, the external quality is higher for the UMAP reduction. That is, the labels assigned are closer to the true ones than in the other cases. This can be seen in a slightly higher Fowlkes-Mallows score and a sample distribution between clusters closer to the true one, which was a 66% and 10% for the majority and minority class respectively.

As a final note, we observe very similar results using the full dataset and the PCA reduction, even in terms of execution time. This could be due to the ability of PCA of preserving the global structure of the dataset and the fact that we kept 86% of the original variance. But we did see a

	Full Dataset	PCA reduction	UMAP reduction
Clusters [% of samples over the total]	0: 40 1: 33 2: 27	0: 40 1: 33 2: 27	0: 54 1: 36 2: 10
Execution time	16min 49s	16min 26s	10min 15s
Calinski-Harabasz index	6303.9349	6303.8494	3708.5501
Davies-Bouldin index	2.7210	2.7208	3.3155
Adjusted Mutula Information score	0.0024	0.0024	0.0045
Fowlkes-Mallows score	0.4177	0.4177	0.4548

Table 3: Information and validation metrics for the different representations of Connect-4.

decrease in the computation time when using the UMAP reduction, saving up more than 6 minutes.

4.3 Pen-Based

As with the the other dataset, we use the same parameters of the previous exercises and we will compare the results obtained for the K-Means algorithm applied to original dataset and the reduction. The parameters used were: (i) K = 10, (ii) init_method = k-means++, (iii) metric = euclidean.

In Figure 7 we can see the results obtained for this dataset. As with the previous datasets, it is impossible to distinguish the clusters when looking at the original dataset. However, they are much more defined after applying any of the dimensionality reduction algorithms used in this work.

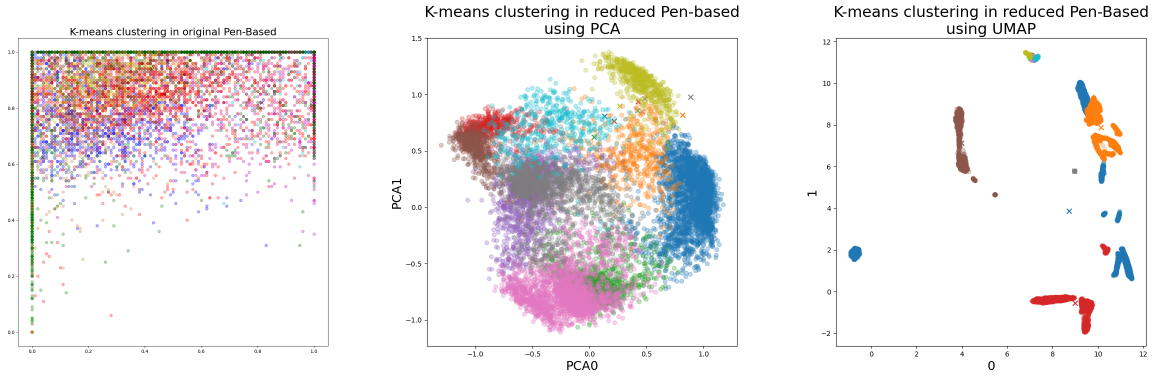

Figure 7: Results of applying K-means to different representations of the Pen-Based dataset.

When comparing the quality of the clusterings in the three configurations (see Table 4), we observe how the external quality does not vary greatly between them. This might indicate that both PCA and UMAP are preserving quite faithfully the global structure of the data. Nevertheless, when looking at the internal validation, we see how PCA is reaching better intra-cluster quality on average. As stated before, we suspect that this differences in cluster quality arise due to how PCA and UMAP transform the data.

	Original Data	PCA reduced	UMAP reduced
Calinski-Harabasz index	2731.2339	3490.3717	1540.9969
Davies-Bouldin index	1.1764	1.0500	0.9248
Adjusted Mutula Information score	0.6710	0.6710	0.6406
Fowlkes-Mallows score	0.5573	0.5573	0.5097

Table 4: Validation metrics for the different representations of Pen-based.

5 Conclusions

In this work, we have studied the implications of using a dimensionality reduction algorithm both for feature selection and visualisation. After analysing the results described in the previous sections, we have reached to the following conclusions.

First, we observe that when working with high-dimensional data it is very important to appropriately select the relevant features in order to reduce the data to a more manageable size. Even more important is the possibility this offers of eliminating irrelevant features, as well as the discovery of latent features.

Second, applying these techniques as a prepossessing step before clustering proved to be effective. Not only the execution time was lower in these cases but we also achieved a better performance in the K-means algorithm overall.

Third, the new representations revealed and facilitated the perception of different characteristics in the data. We could better appreciate the global structure of the datasets and the local relationships among points in the low dimensional embeddings.

References

[1] *2.3. Clustering*, en. [Online]. Available: `https://scikit-learn/stable/modules/clustering.html` (visited on 11/20/2021).

[2] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[3] *2.5. Decomposing signals in components (matrix factorization problems)*, en. [Online]. Available: `https://scikit-learn.org/stable/modules/decomposition.html` (visited on 11/21/2021).

[4] A. Janosi, W. Steinbrunn, M. Pfisterer, R. Detrano, and M. M.D., *Heart Disease*, UCI Machine Learning Repository, 1988.

[5] T. Minka, "Automatic choice of dimensionality for pca," *Advances in neural information processing systems*, vol. 13, pp. 598–604, 2000.

[6] J. Tromp, *Connect-4*, UCI Machine Learning Repository, 1995.

[7] E. Alpaydin and F. Alimoglu, *Pen-Based Recognition of Handwritten Digits*, UCI Machine Learning Repository, 1998.

[8] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[9] *Using UMAP for clustering*, en. [Online]. Available: `https://umap-learn.readthedocs.io/en/latest/clustering.html` (visited on 11/20/2021).