Characterization of the k-means algorithm for spectral profiles

Trabajo de Fin de Grado
Grado en Física

ALUMNA

Eva Sola Viladesau

TUTOR

Dr. Daniel Elías Nóbrega Siverio
Instituto de Astrofísica de Canarias

COTUTOR

Dr. Fernando Moreno Insertis
Universidad de La Laguna - Instituto de Astrofísica de Canarias

Mayo 2023

Abstract

The k-means algorithm is a Machine Learning clustering method that has gained popularity both for its scalability and its simplicity. The output of this method contains a distribution of the input data in k groups as well as k representative examples.

The aim of this Bachelor's Thesis is to test k-means clustering results under controlled conditions by means of an artificial dataset. The data mimic solar observations from the *Interface Region Imaging Spectrograph* (IRIS) in the Mg II h&k lines. The situation is made incrementally more complex and the impact on the clustering is studied on a case by case basis. The goal is to consistently obtain a distribution that accurately separates the different profiles in the dataset. Furthermore, the results are compared to those of hierarchical clustering methods and the effect of two common preprocessing schemes is analyzed.

The k-means final results are considered satisfactory, given that the main goal of discerning between spectral behavior patterns is achieved with very low error rates, even when the data are purposefully contaminated with defective profiles and noise. Nevertheless, when these impediments become too widespread, masking becomes necessary, allowing for the previous statistics to be recovered. The hierarchical methods are deemed equal or inferior to k-means in terms of performance, depending on the specific criterion.

Keywords: Machine Learning, k-means algorithm, agglomerative hierarchical clustering, feature scaling, Principal Component Analysis

Resumen

El algoritmo k-means es un método de *clustering* de aprendizaje automático que ha ganado popularidad por su escalabilidad y su simplicidad. El *output* de dicho método es una distribución en k grupos de los datos introducidos, además de k ejemplos representativos.

El objetivo de este Trabajo de Fin de Grado es someter a escrutinio los resultados de k-means bajo condiciones controladas por medio de un conjunto de datos artificial. Los datos imitan observaciones solares obtenidas con el satélite *Interface Region Imaging Spectrograph* (IRIS) en las líneas de Mg II h&k. La complejidad de la situación se incrementa gradualmente, estudiando el impacto sobre el agrupamiento caso por caso. La finalidad es obtener consistentemente una distribución que separe los diferentes perfiles del conjunto de datos. Además, los resultados se comparan con los de algunos métodos de *clustering* jerárquico y se analiza el efecto de dos estrategias de preprocesado comunes.

Los resultados finales de k-means son satisfactorios, dado que el objetivo principal de discernir comportamientos espectrales se consigue con tasas de error muy bajas, incluso cuando los datos están intencionadamente contaminados con píxeles defectuosos y ruido. Sin embargo, cuando estos impedimentos se extienden demasiado, es necesario aplicar máscaras, que permiten volver a las estadísticas previas. Los métodos jerárquicos son declarados inferiores o iguales a k-means en términos de cuántos ejemplos agrupan correctamente, dependiendo del criterio concreto.

Palabras clave: Aprendizaje automático, algoritmo k-means, agrupamiento jerárquico aglomerativo, escalado, Análisis de Componentes Principales

Contents

1. Introduction

El *Machine Learning* o aprendizaje automático es una rama de la inteligencia artificial en la que se entrenan modelos matemáticos mediante algoritmos para efectuar predicciones. Las dos grandes ramas en las que se divide son el aprendizaje supervisado y el no supervisado. El aprendizaje no supervisado contiene al *clustering* o agrupamiento.

k-means es un algoritmo de *clustering* que separa un conjunto de datos en diferentes grupos, asignando un "centroide" a cada uno. Este centroide actúa como media representativa del *cluster*.

1.1. Machine Learning. Unsupervised learning and clustering.

The term Machine Learning (abbreviated ML) was first coined by Arthur Lee Samuel, who developed a trailblazing program capable of playing checkers and gradually learning how to become better at the game by analyzing each position and its possible ramifications, much like a human mind would (Samuel, 1959). This algorithm was said to reach an amateur playing level after years of development. Since these humble beginnings, Machine Learning and artificial intelligence in general have flourished and continue to grow exponentially, the latest examples being some recently controversial Artificial Intelligence chat bots, which are profoundly impacting modern society (Dwivedi et al., 2023).

Machine Learning, at its core, is the process of training a program (called a *model*) to make predictions or decisions, with the capability of *learning* from previous results. Mathematically, the outline is simple: at first, the model has a set of initial coefficients that allow it to take the input data and calculate a result, which is translated onto a specific answer if the desired output is not in the form of a number. For example, some predictions require yes or no answers, which could be manually assigned to certain result ranges. The act of training the model changes these coefficients recursively, slowly making the model more accurate in its predictions. Given a sufficient amount of high-quality data, an ML algorithm is capable of finding relationships between data points or variables that a human being would be incapable of.

Applications of Machine Learning are as wide as one can imagine. Even the development of seemingly inconsequential algorithms such as a checkers playing program can help develop new strategies that are then extrapolated for other uses. ML has found great success in social media engagement algorithms, text pattern recognition for various end goals, information compression for scientific purposes, climate and weather predictions, the early stages of self-driving vehicles, text and image generation, assistance in medical diagnoses, and many more (Alzubi et al., 2018).

Like most areas in science, Machine Learning has its own jargon. Here, we define some terms that will be used later on:

- **Example**: each of the singular data points in the dataset. Each example has a defined value for all the features, and they may have a label in the case of supervised learning, which we define further below. When treating spectral data, for instance, each spectral profile is an example.

- **Feature**: each of the properties that characterize an example. Not all features are equally useful for prediction purposes. The choice of features will depend on the available computational power, the desired precision, and the nature of the prediction one wishes to make. In a spectral profile, the intensity at each certain wavelength is a feature.

- **Label**: the empirical, true answer of the prediction for an example. These are only sometimes available, and if they are, the goal of the algorithm is to match these labels when producing results. A label could be manually given to a spectral profile if desired, identifying it as the outcome of a certain phenomenon or the result of a specific atomic transition, for example.

- **Dataset**: the entirety of the data available for training and testing the algorithm, which includes all examples with their respective feature values and, optionally, labels.

- **Model**: the set of mathematical relationships that the algorithm finds and uses to make a prediction.

- **Convergence**: When the algorithm stops iterating because a given condition has been fulfilled, it is said to have converged, and the situation at the last iteration is taken as that run's final results.

A classic example of ML is that of automatic spam management. In this case, the terms would be used as follows: we may have a *dataset* that contains all the e-mails managed by a company on a certain day, and we may try to predict whether or not the e-mail contains spam. The *examples* would be the individual e-mails, and some *features* could be: the hour when it was received, the presence of certain keywords or the number of characters in the e-mail. Each e-mail has an assigned *label* that says if it truly contains spam. Upon *convergence*, the *model* may find that the presence of keywords is a much more reliable way to tell if an e-mail is spam or not compared to the other features.

As a side note, in order for an ML algorithm to work, all the features must be numerical. If they are not (like the presence of keywords), then they must be encoded as numbers somehow. Such encoding is outside of the scope of this work, since the features we use are inherently numerical.

ML can be divided into two main branches: supervised learning, which is used with datasets that contain labels, and unsupervised learning, for those datasets that have none. Clustering methods belong to the latter.

Supervised learning comprises both regression and classification. These are intended to predict quantifiable (in the case of regression) or classifiable (in the case of classification) phenomena that have a true outcome, and therefore are labeled. Real life examples would be a model that predicts the amount of precipitation on a city (regression) or an algorithm that identifies handwritten digits (classification). Classification may be binary or multivalued.

Unsupervised learning, however, is not designed to find a known pattern. Instead, it aims to find relationships between examples in the dataset that programmers cannot find on their own. This is mainly used for data reduction and to find links between data that can then provide some insight into the system of study. In this kind of problem, the programmer must make use of mathematical, logical and statistical methods to analyze the performance of the model.

Clustering methods attempt to form groups such that the examples in the same group or *cluster* are as similar as possible while the groups themselves are as different from each other as possible. Specifically, in the case of this Bachelor's Thesis, the aim is to get an algorithm to identify and cluster regions with similar spectral behavior, in order to reduce the amount of data that will require human analysis or undergo lengthy operations. This will be discussed later in more detail.

1.2. The *k*-means algorithm.

The k-means algorithm is a widely used clustering method, and it will be the focus of this work. Say we have a dataset with a number of examples n, each with a number of features f and numerical values for each feature, and we would like them to be distributed in k clusters.

The model's modus operandi is the following: the number of clusters k is input by the programmer. Then, the first step is to choose k *centroids*. These centroids are generated by copying k random examples in the dataset. The creation of the centroids is called *initialization*[1].

Then, each example in the dataset is assigned to the centroid which is closest to it distance-wise. Most commonly, the Euclidean metric is used, identifying each feature with a dimension and using the definition of Euclidean distance for a space with f dimensions. At this point, the centroids are redefined as the mean of all the examples assigned to it. This is the working definition of a centroid and it is the one that will be used from now on, unless otherwise stated. The centroids do not necessarily coincide with examples present in the dataset anymore. After the redefinition of the centroids, the examples are reassigned to their closest centroid, which is, in general, not the same one as before. This marks the end of the first iteration.

Every subsequent iteration involves (a) the recalculation of the centroids according to their definition and (b) the reassignment of the examples. Note that after step b is carried out, the centroids do not necessarily match with the mean of the examples assigned to them because, in general, some examples have switched groups. This continues on until k-means declares convergence, which happens once the difference in the centroids' position between two consecutive iterations is lower than a certain threshold value called the *tolerance*[2].

Figure 1 is an example of a fully converged k-means clustering result on a two-dimensional distribution of points. These types of datasets are ideal for illustrating the algorithm since the concept of Euclidean distance on a plane is completely intuitive. In this case, the features are the x and $\cdot y$ values. The centroids are represented by the white crosses. Datasets with more than 2 or 3 features are hard to visualize, but the idea remains the same —the clusters will contain examples that are close to each other (as per the Euclidean metric) in this

[1]Due to abuse of language, oftentimes the word "initialization" will be used to refer to the full convergence process. Therefore, "performing N initializations" means "to run the data through the algorithm N times, each with different centroid seeds".

[2]More specifically, the algorithm declares convergence when the Frobenius norm of the matrix given by the difference between the centroids' array in two consecutive iterations is lower than the tolerance value.

phase space, the multidimensional space made up by all the features.

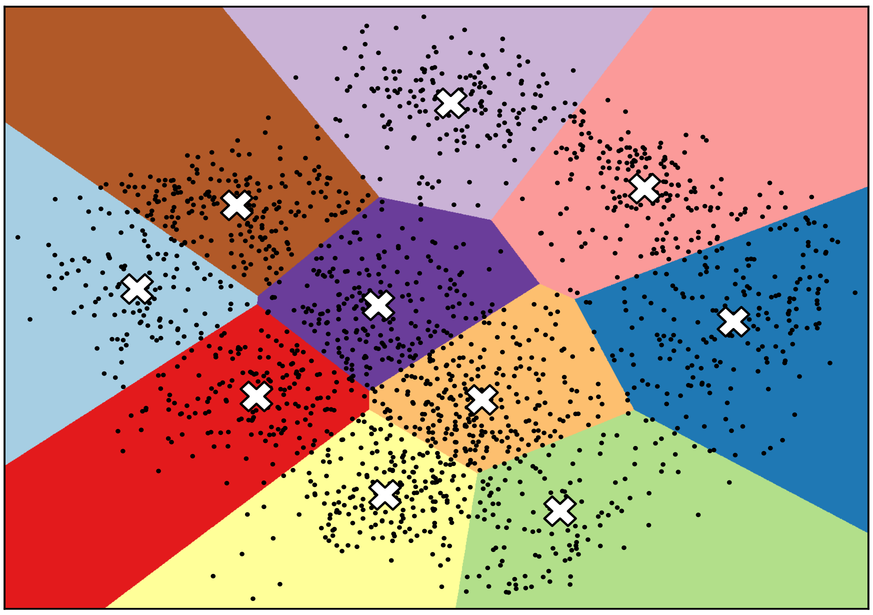


Figure 1: Clustering by k-means on a 2-dimensional distribution (the features are the x and y values). Image adapted from an example code in the k-means user guide (scikit-learn, 2023b).

The way k-means operates is designed to minimize the second-order central momentum or *inertia*, both for each cluster and globally. This quantity can be defined as follows. A single cluster's inertia is:

$$\sum_{i=0}^{n}\sum_{m=0}^{f}(||x_m^i - \mu_m||^2) \tag{1}$$

where n is the number of examples, f is the number of features, x is an example and μ is the mean of the cluster's examples. As we can see, the difference is computed for every feature.

The total inertia is therefore:

$$\sum_{j=0}^{k}\left[\sum_{i=0}^{n_j}\sum_{m=0}^{f}(||x_{mj}^i - \mu_{mj}||^2)\right] \tag{2}$$

where we have simply added the inertia for all clusters, j being the index that identifies a cluster and n_j being the number of examples in cluster j.

The way inertia is computed, namely, as the total distance to the local cluster mean summed over all clusters, may cause k-means to be inaccurate when it comes to anisotropic clusters, where one feature or combination of features exhibit a wider range of values. This is one of the reasons why *scaling* the data is sometimes useful. The viability of feature scaling will be discussed in Sections 7.1 and 7.3.

Although it is a useful success metric, it can be hard to judge a model's performance solely on the inertia. For example, though inertia values close to 0 are desirable, a clustering with a number of clusters equal to the number of examples would yield 0 inertia. One would be hard-pressed to consider this a useful situation, since the clustered data are identical to the original data and no new information has been gained. Additionally, because of the so-called curse of dimensionality, inertia tends to inflate in high-dimensional datasets (in other words, datasets with many features). For these reasons, the most useful way to use inertia in the case that concerns us will be by comparing its values for various choices for the number of clusters to use, k, under the same conditions. The details on this will be given later on as this metric is introduced.

Due to the nature of the algorithm, a number of problems may arise. Firstly, the random positioning of the centroids upon initialization might cause convergence to happen to a local minimum rather than the optimal clustering distribution. As a means to avoid this, the algorithm allows for a number of initializations to be done, each with different centroid seeds, keeping the results with the lowest inertia. Another way to avoid local minima is to use the k-means++ initialization scheme (Arthur and Vassilvitskii, 2007), which ensures that the centroid seeds are somewhat distant from each other rather than completely random. We will use both k-means++ and several initializations. Secondly, the tolerance threshold may never be reached if the number of clusters or the data do not allow for a proper convergence. k-means also accounts for this, letting the user set a maximum number of iterations. If this maximum is reached, the algorithm will stop and the results from the last iteration will be kept. In the event of this termination, the centroids may not coincide with the average of all the profiles assigned to it, since the iterative process stops before the centroids can be recalculated. This can be spotted and it was evaluated for every case to make sure convergence was reached naturally.

2. System under study

La cromosfera y la región de transición son dos zonas de la atmósfera solar de gran interés. El satélite IRIS observa rutinariamente estas capas mediante espectroscopía e imagen (De Pontieu et al., 2014). Entre las líneas espectrales que es capaz de observar se encuentran las de Mg II h&k, que han sido ampliamente utilizadas como herramientas de diagnóstico en la cromosfera (por ejemplo, Leenaarts et al., 2013b; de la Cruz Rodríguez et al., 2016; Kriginsky et al., 2023), y además también han sido objeto de *clustering* con k-means (Panos et al., 2018; Bose et al., 2019; Nóbrega-Siverio et al., 2021, entre otros).

Se construyeron dos conjuntos de datos artificiales simulando las líneas espectrales Mg II h&k. Para ello se sumaron varias funciones gaussianas centradas en diferentes longitudes de onda y con distintas desviaciones estándar, emulando la forma de los perfiles reales. Se crearon rásteres tridimensionales cuyas primeras dos dimensiones pueden considerarse los ejes x e y de una imagen y cuya tercera dimensión es el eje de longitudes de onda, distribuyendo ordenadamente los perfiles para su identificación inmediata tras el proceso de *clustering*.

2.1. The Mg II h&k lines. The Interface Region Imaging Spectrograph.

The chromosphere is a dynamic, complex layer of the Sun's atmosphere that lies between the photosphere and the corona. This region is important both for the many physical processes that occur within its confines, such as magnetic reconnection, wave propagation, ionization/recombination out of equilibrium, and so on; as well as the different associated phenomena, namely, surges, spicules, prominences, among others. In addition, anything that gets to the corona must pass through the chromosphere first. Hence, the characterization of the chromosphere may very well be key in understanding fundamental solar open questions, for instance, the solar wind, the solar coronal heating, and the coronal magnetic field (see the Introduction on de la Cruz Rodríguez et al., 2016).

The Mg II h&k lines are high-quality diagnostics for many processes in the chromosphere (e.g., Leenaarts et al., 2013b; Kriginsky et al., 2023). For this reason, inversion codes are often used on these lines when focusing on said region (e.g., the STiC code by de la Cruz Rodríguez et al., 2016; de la Cruz Rodríguez, J. et al., 2019). Inversion can be computationally costly, especially when undertaking considerations of Non Local Thermodynamical Equilibrium (NLTE), which is common in the chromosphere and particularly in the Mg II h&k lines (Leenaarts et al., 2013a). This is why spectral datasets could benefit from a clustering method: instead of inverting every single spectral profile, the inversion code is run on a representative subset of the original data, potentially saving a significant amount of time and, if done right, neglecting a very minimal amount of information.

The Interface Region Imaging Spectrograph (IRIS) is a satellite launched by the National Aeronautics and Space Administration (NASA) in 2013 as part of the Small Explorer program. It is designed to provide images and spectroscopic data focusing on the chromosphere and the transition region (De Pontieu et al., 2014), and it is often used to retrieve Mg II h&k data. In light of this, the chosen dataset of this Bachelor's Thesis will consist of artificial profiles mimicking the Mg II h&k lines, and they will be collected in a three-dimensional raster, much like IRIS data.

It is worth mentioning that the use of Machine Learning and the k-means algorithm to study these lines is not unheard of. Clustering methods are beginning to be exploited as data reduction mechanisms to assist and influence subsequent human analysis (Panos et al., 2018; Bose et al., 2019; Nóbrega-Siverio et al., 2021, among many others), further backing the previous argument. The present work intends to be but an introduction to such strategies.

2.2. Creation of artificial data.

A three-dimensional raster is built, the first two dimensions being the x and y axes of a hypothetical IRIS image and the third dimension being the wavelength axis. The dimensions for the Mg II h&k raster are (108, 64, 327). We will refer to the 108x64 (x, y) positions in the raster as pixels, following the analogy with a digital image. Eight different spectral profiles, shown in the left panels of Figure 2, are created by adding Gaussian functions centered at specific wavelengths and with different standard deviations. In ML terms, each spectral profile is an example and the intensity at each wavelength is a feature. Therefore, there are $64 \cdot 108 = 6912$ examples and 327 features. The distribution of the profiles in the raster is as follows: the first profile is assigned to all pixels in the first 8 rows; the second profile to all pixels in rows 9 through 16; and so on, until filling the whole matrix.

An illustration of this pattern is provided in the right panel of Figure 2.

This spatial distribution, although unrealistic, will allow for an immediate estimate of performance upon seeing the clustering results in a color map. No issues arise because of this ordering since k-means is insensitive to example order.

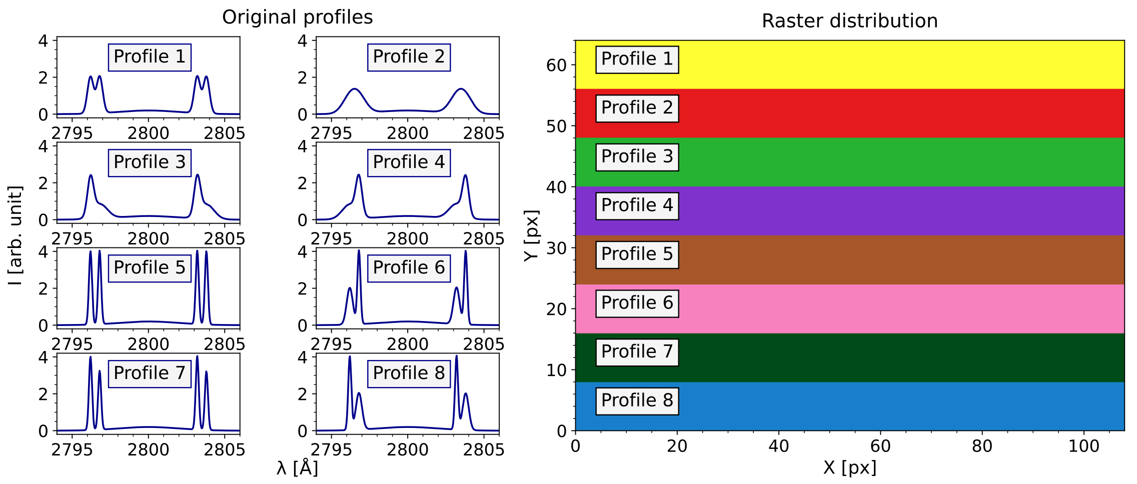


Figure 2: Different profiles included in the dataset (left) and distribution of each profile in the raster (right). Each profile fills 8 consecutive rows, as indicated by the different colors.

3. Control case

Se realizó un primer *clustering* con los datos originales. Los resultados tuvieron una inercia total de cero (salvo precisión computacional), y los grupos finales estaban todos compuestos por 864 ejemplos, cada grupo conteniendo solo un tipo de perfil. Por tanto, el agrupamiento fue tal como se esperaba obtener, sin mezclar distintos comportamientos espectrales. Se comprobaron los resultados mediante la inercia total y el comportamiento de *plateau* de la inercia.

3.1. Raw data. Preliminary results and first success metrics.

The k-means method was implemented via the scikit-learn module for Python (Pedregosa et al., 2011), which is a Machine Learning oriented package. A first clustering was performed on the aforementioned raster. This will serve as a benchmark with which other cases will be compared. The parameters input in k-means were the following: the number of initializations was 100, the maximum number of iterations for each initialization was 450, the tolerance was 10^{-5} (same units as the intensity) and we chose a certain random state seed so that the first centroid chosen would remain the same upon running the clustering several times and the results were somewhat reproducible.[3] Having input 8 different profiles in the raster, k-means was always run with the instruction to find 8 clusters, unless otherwise stated.

After retrieving the results, we show in an x, y frame the cluster assigned to the spectrum in each pixel: this is achieved by using different colors for the different clusters. The resulting color map, or 'image' is expected to look like the one in Figure 2, with eight clean stripes occupying eight rows each, though the ordering of the colors may change. This result is indeed obtained and it is shown in Figure 3.

The reason for the color arrangement changing is the following: the colors are

[3]By using k-means++ and a random state together, we ensure that the first centroid seed is always the same, but the other centroids are chosen via the k-means++ probability function and can vary. Hence, it still makes sense to do several initializations.

assigned sequentially, always in the same order (the one that appears on the color bar on top of the left panel in Figure 3), to the clusters. However, we sorted the clusters by their number of examples, and when this number is the same for several clusters, their order is arbitrary. This is due to the fact that k-means does not follow a spatial order when assigning labels.

The matrix on the right in said Figure is analogous to a *confusion matrix*. These are used in supervised methods to quantify the amount of errors made, since the label of each example is known and can be compared to the label assigned by the ML algorithm. In this case, label numbers were not always associated with the same profile. However, it is possible to count the number of instances of each label at every 8-row layer of the raster, and subsequently build a confusion matrix that identifies how many profiles were mislabeled with respect to the layer's general trend. Since this method relies on comparison between label frequencies, it will only be applicable when the vast majority of pixels in the same layer share a label.

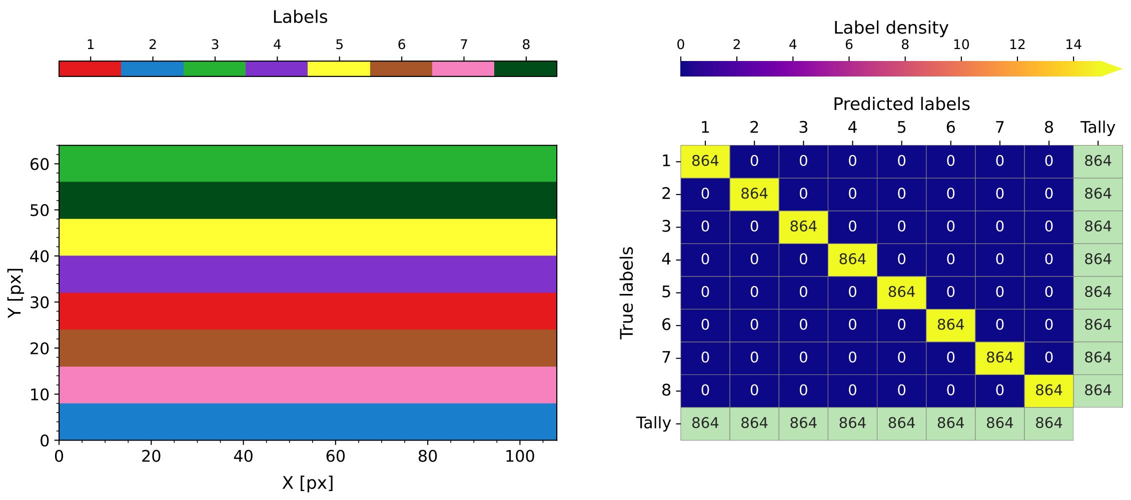


Figure 3: Left: the different clusters for this control case are assigned a color and a label each. In the panel, the pixels show the color of the cluster they have been associated with by the k-means processing. Right: confusion matrix for this case. The results show no errors.

The tallies on the bottom row of the confusion matrix in Figure 3 show that all clusters contain an equal amount of examples: 864 each. This amount is often called the *cardinality* of the cluster. This quantity holds little meaning in general cases more complicated than the very simple test we are doing in this section. In those cases, the relative rate of occurrence of each behavior is unknown, and therefore we have no guidelines on how many examples each cluster should have. However, in our toy dataset, it is clear that equal cardinalities (exactly as they are in Figure 3) are the ideal result.

In this case, the inertia per cluster (Eq. 1) is practically zero for all clusters (see Figure 4), and so is the total inertia (Eq. 2). The fact that they are not exactly zero is due to computational precision limitations. These will be taken as baseline values to interpret the other results, as they are as close to perfection as possible using this algorithm and the described dataset.

The last success metric that will be introduced for now is the behavior of the total inertia as the number of clusters changes. This technique is referred to as a *plateau* or *elbow* alluding to the distinctive shape of the produced graph, and it has been used in research (see Appendix in Bose et al., 2019 and in Nóbrega-Siverio et al., 2021).

It works as follows: the k-means algorithm is requested to fully converge (under the same conditions of regular runs: 450 iterations maximum, 100 initializations, 10^{-5} tolerance in the same units as intensity) to a varying number of clusters. First, it is asked to find 2 clusters, then 3, and so on all the way to 15.

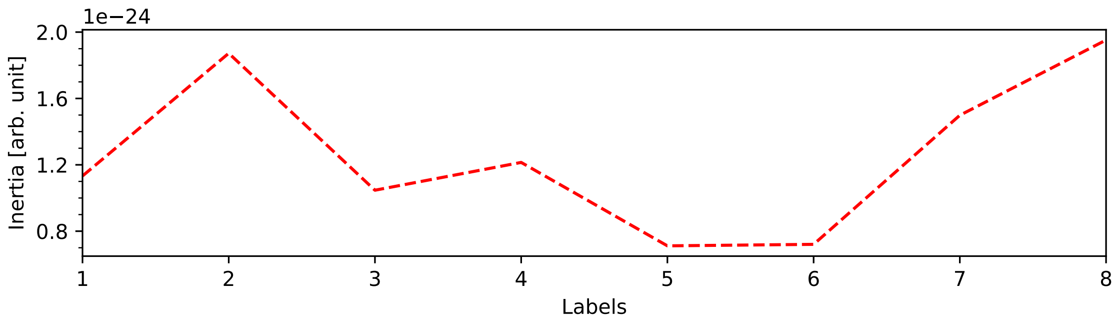


Figure 4: Inertia per cluster (Eq. 1) for the control case. Each cluster is assigned to a certain label, which need not necessarily correspond to those shown in Figure 2.

For a number of clusters (k) lower than the number of distinct profiles included in the dataset (8 in our case), the inertia is very high, since very different examples are being clustered together. Increasing the number of clusters steadily decreases the inertia at first, but as k approaches the number of distinct profiles, the values of the inertia stabilize. This change in the rate of descent happens because, as k surpasses the number of profiles, similar examples will be separated, but each cluster will still contain very similar examples. Once we reach this point, the inertia will not change much regardless of the exact value of k.

When real data are employed, the "ideal" number of clusters is unknown. In this scenario, the researcher must find a number of clusters that sufficiently reduces the data volume in order to manually analyze the minimum number of behaviors while taking care that very different examples are not being put into the same group. One possibility is to approximate this number by looking for an inertia plateau, and then adjusting if necessary.

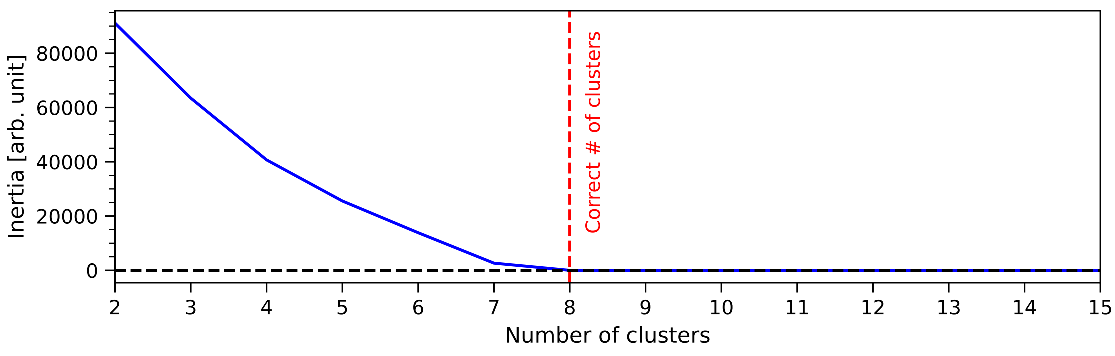


Figure 5: Total inertia plateau behavior for the control case. The dashed black line shows that after 8 clusters, the inertia becomes zero (save for machine precision).

The final results for this inertia plateau check are displayed in Figure 5. In this first case, when we have 8 clusters, the profiles in each cluster are identical to each other and therefore the point of inflexion is very clear. The values, which are as close to zero as computationally possible after 8 clusters, indicate that having fewer than 8 is highly inadequate and more than 8 is unnecessary separation. Thus, the conclusion is that 8 clusters is an ideal number in this situation. This behavior will be sought after in the following cases as well.

4. Added noise

Tomando el caso de control como referencia para los posteriores resultados, se introdujo un ruido aleatorio con una distribución normal (ruido blanco Gaussiano) que desvió los perfiles de sus valores originales. Ante este cambio, los ejemplos de un mismo perfil dejaron de ser idénticos. Los resultados del *clustering* siguieron siendo perfectos, separando adecuadamente los ocho comportamientos espectrales. A pesar de que la inercia aumentó en muchos órdenes de magnitud, afectada por el ruido y por la alta dimensionalidad, se siguió observando el comportamiento de *plateau*, lo que justificó el número de *clusters* escogido.

4.1. Addition of noise to the profiles.

The original profiles serve well as a control case, however, in order to realistically test the algorithm, some modifications were implemented to make the data look more like a true observation.

As a first strategy to achieve this, White Gaussian noise (random noise with a normal distribution which is equal for all wavelengths) was added, with a standard deviation of 0.2, which lies between 5 and 10% of the signal amplitude, depending on the profile. This made the profiles look jagged and irregular, which is a common feature of observational data. An example of these new profiles can be seen in Figure 6.

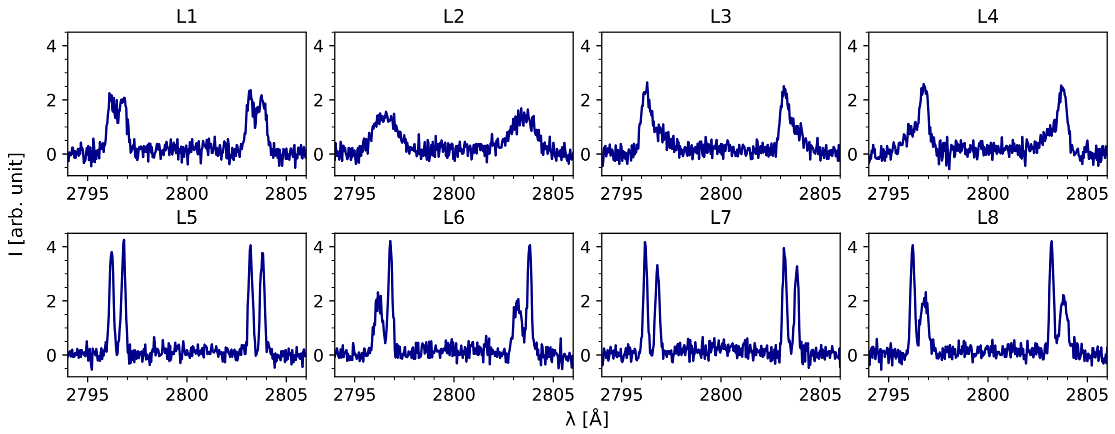


Figure 6: Profiles with an added normal noise distribution. The amplitude of said distribution is the same for all examples. While there is contamination in their shape, the general look of each profile is maintained.

A very important remark is in order: in the previous section, the profiles in each "layer" of the raster were completely identical. Now, while the base profile is shared within a layer, each pixel holds a unique example, since the exact sequence of noise values is different for every (x, y) position. The examples in Figure 6 are meant to be a visual guideline of what the data look like at this point.

Though the examples look quite noisy, their general shape is still discernible, to the extent where if a profile were to be put into the wrong cluster, one would likely be able to tell, if it was superimposed with the other profiles. Following this idea, we introduce a new type of graph: the centroid and the "most different profile" in a cluster, plotted in the same figure. The profile with the largest cumulative standard deviation from the centroid is chosen as the "most different one". This graph might allow us to identify clusters that contain at least one stray profile that clearly belonged in another group, and would be the principal metric to do so in general cases that do not have an ordered distribution like the present dataset.

4.2. Results. New success metrics.

After adding noise, the algorithm held up and continued to yield the same results as for the control case, correctly segregating the different spectral patterns as intended. This is a good first sign, as it proves that k-means can withstand some alterations to the examples and accurately recognize the patterns that are most interesting to us, that is, the shape of the peaks of the lines.

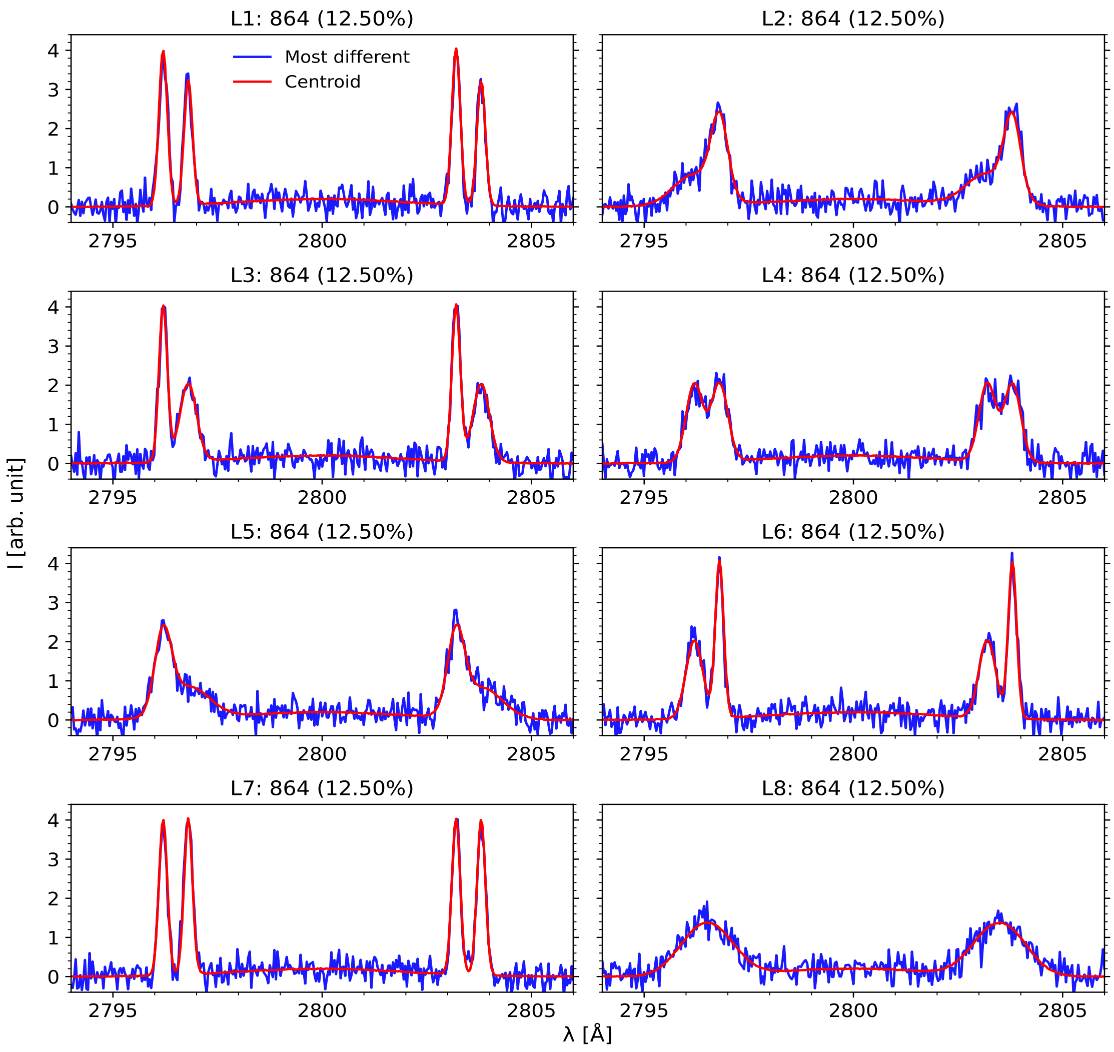


Figure 7: Most different profile of each cluster (blue) together with the corresponding centroids (red) for the case with noise. On top of each panel is each cluster's cardinality in absolute value and percentage.

Figure 7 shows the aforementioned most different profile (in blue) in contrast to the centroid of each cluster (in red). This graph alone serves as an appreciation of the general performance of the clustering: firstly, we can once again look at the number of examples in each cluster. The aim is to have equal cardinalities of 864. In this case, no incidents seem to have happened. However, it is possible that switching of an equal amount of examples might have happened, leaving no trace on the cardinalities. This would be easily seen on the confusion matrix and the color map. Nonetheless, detection could happen here as well since a wrongly clustered example is likely to be the most different one. By seeing this Figure only, one could conclude that the clustering has been successful, since no most different profiles look out of place.

On another note, the inertia shows significant changes. In this case, the effect of arbitrary deviations from the profiles caused by the random noise, combined with the large dimensionality in the dataset, drive the inertia per cluster up to around $1.1 \cdot 10^4$ (Figure 8).

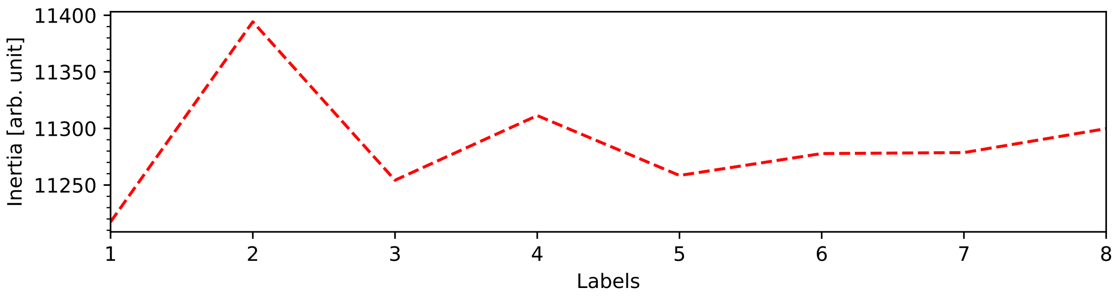


Figure 8: Inertia per cluster for the case with noise. Given the high dimensionality and the added noise, values are much higher than the ones for the control case.

In a realistic case, where the data is not as tidy as our raster and therefore we lack a good guideline like the color map to see if we have made mistakes or not, these high values of the inertia could make one wonder how good the results truly are. In that situation, it is still possible to fall back on the inertia plateau to justify the clustering (see Figure 9). This time, the difference between having 6, 7 or 8 clusters is not as striking.

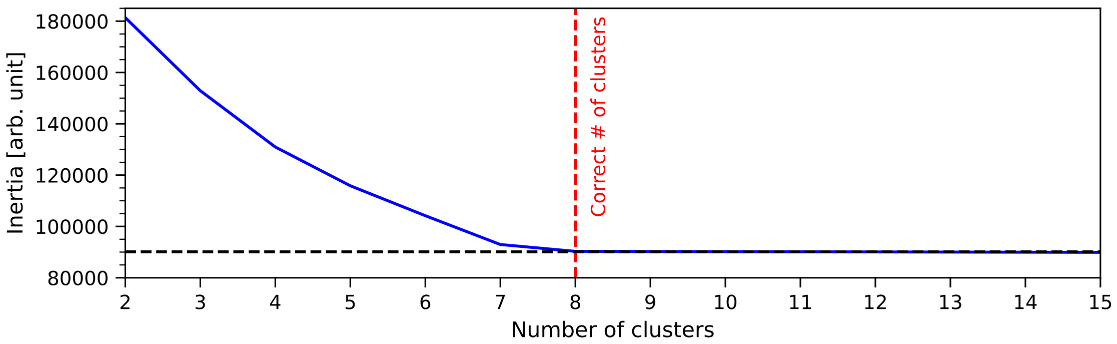


Figure 9: Plateau test for the case with noise.

However, the plateau behavior is clearly still there and therefore 8 still seems to be an appropriate number of clusters. Nevertheless, this supports the chosen number of clusters, but not the distribution of the examples. To analyze that, we may turn to Figure 7 or the color map (not shown for this case).

Another useful verification is to plot the relative difference between each cluster's centroid and the mean of all profiles assigned to it as a function of wavelength (see Eq. 3).

$$\delta(\lambda) = \frac{\mu(\lambda) - \bar{x}(\lambda)}{\mu(\lambda)} \cdot 100 \quad [\%] \tag{3}$$

with δ being the relative difference in percentage, μ the centroid of the cluster and $\bar{x}$ the mean of the intensity at that wavelength for all the examples in the cluster.

Since, theoretically, μ and $\bar{x}$ should be the same, a result that greatly differs from 0 in most wavelengths will mean the algorithm declared convergence without fulfilling the tolerance condition, and consequently, that the number of iterations was fully exhausted before k-means had the chance to reach the best distribution. Should this issue happen, the maximum number of iterations must be increased until this phenomenon disappears (ideally).

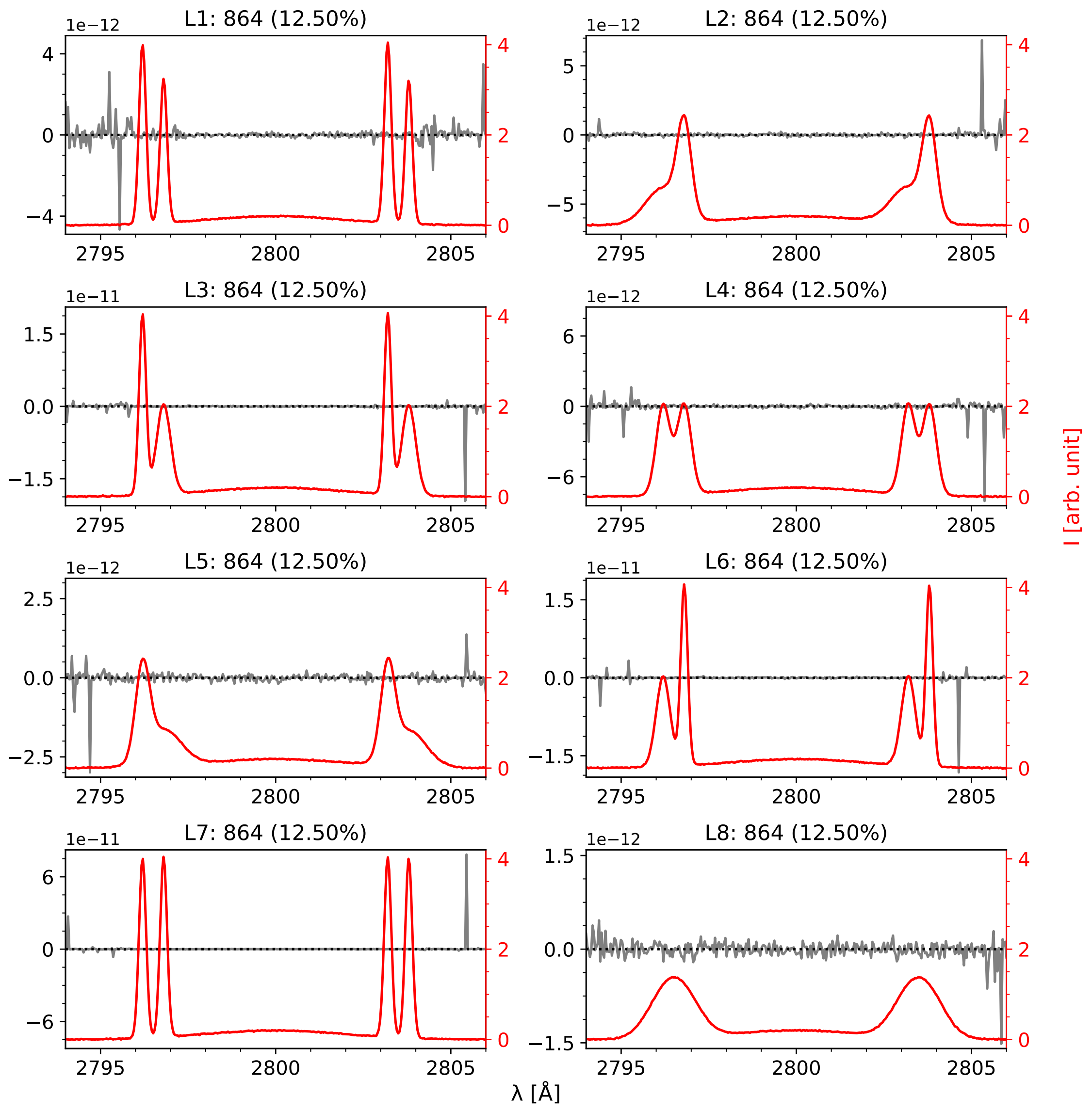


Figure 10: Relative differences between the mean of each cluster and the centroid (as per Eq. 3) for the case with noise.

Figure 10 shows low values for this difference, suggesting that the algorithm reached convergence before reaching the maximum number of iterations. The profiles have also been plotted in red, for ease of showing the wavelength placements of the largest relative differences. There are visible peaks at the edges of the x axis; however, they are not a major issue: Eq. 3 shows that we are dividing by the centroid's value at each wavelength, therefore it is expected that this quantity could become exceptionally large in places where the centroid is close to zero. Moreover, the order of magnitude is, at most, 10^{-10} in this Figure, so the highest values are still not a concern.

All in all, the clustering is very successful for a reasonable amount of normalized noise. As an interesting comment, we notice that the centroids (Figs. 7 and 10, in red) look extremely similar to the original profiles. This is due to having averaged out over 800 profiles for each cluster with normalized, white noise. A non-normalized noise distribution might not yield such a good replica of the original data.

We have also tested the algorithm's resilience when adding noise of a higher amplitude. For instance, for a standard deviation of 0.4 (10-20% of the signal), the average error rate (pixels that were put in a group that did not correspond to its original shape) in 500 runs was 1.87 errors (0.03% of the total number of pixels). This is still a very positive result.

5. Defective pixels

Para poner el algoritmo frente a situaciones más realistas, se introdujeron dos tipos de píxel defectuoso: impactos de rayo cósmico, en los que se añadía al perfil una gaussiana muy estrecha (visualmente, un pico muy pronunciado) en una longitud de onda aleatoria; y píxeles muertos, que consistían en un perfil plano con un valor negativo constante para todas las longitudes de onda.

Al introducir pocos rayos cósmicos, los resultados decayeron ligeramente en calidad, asignando algunos ejemplos a un grupo que no le correspondía según su forma original. Al aumentar el número de rayos se comenzaban a cometer errores más graves, incluso mezclando comportamientos espectrales.

A fin de remediar esto, se introdujeron máscaras, que sustituyeron este tipo de píxeles por perfiles planos en cero, y sabiendo que esto se había aplicado, se realizó el *clustering* con un grupo adicional. Los nuevos resultados recuperaron el comportamiento previo, y los píxeles enmascarados se agruparon correctamente, separados del resto.

5.1. Addition of defective pixels to the raster.

As the final step in the introduction of realistic features, the presence of defective pixels was accounted for in the form of cosmic ray impacts and pixels that malfunctioned. In the same vein as the profile distribution, we added these defects in a predictable pattern, as follows: if the total number of defective profiles for a specific case is less than the number of pixels in a column, we distribute them uniformly in that column. If larger, then they are distributed in as many adjacent columns as necessary to accommodate all of them. The cosmic rays always began in column 10 and the dead pixels, in column 90. Concerning the actual spectral profiles with defects, Figure 11 shows two examples. All dead pixels look identical and contain no information, but profiles with cosmic rays still contain a lot of information, save for the wavelengths that the ray renders unusable, so it is feasible that k-means can still cluster them correctly.

This uniform distribution was chosen so that a mistake caused by a defective pixel could be easily identified as such, as opposed to a mistake that incorrectly clustered an example with no cosmic ray. Once again, no clustering bias emerged, given that k-means does not account for spatial correlation between the examples, and the position of the profile in the raster was not a feature in the dataset.

Cosmic rays were approximated by another Gaussian, albeit much sharper than the ones used for the lines, with a smaller width. The exact width and the wavelength it fell on were both randomized. Any wavelength was a possible impact point, but the full width at half maximum was limited to the 0.07-0.14 Å range. Dead pixels were simply flat profiles with large negative values.

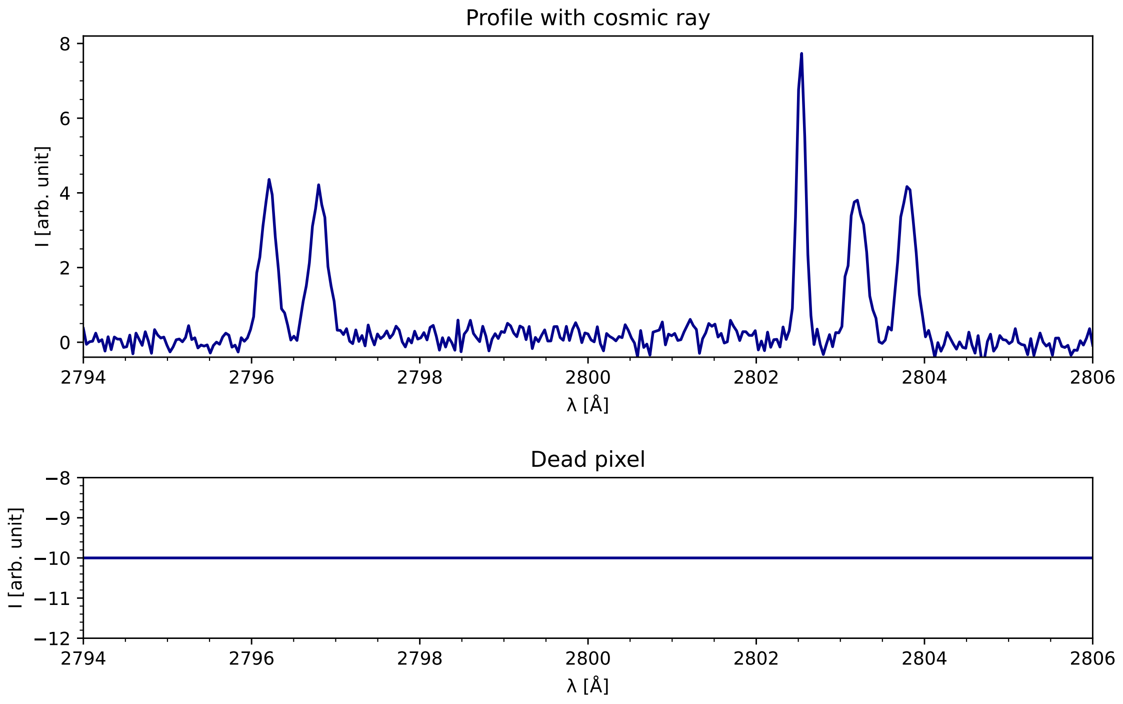
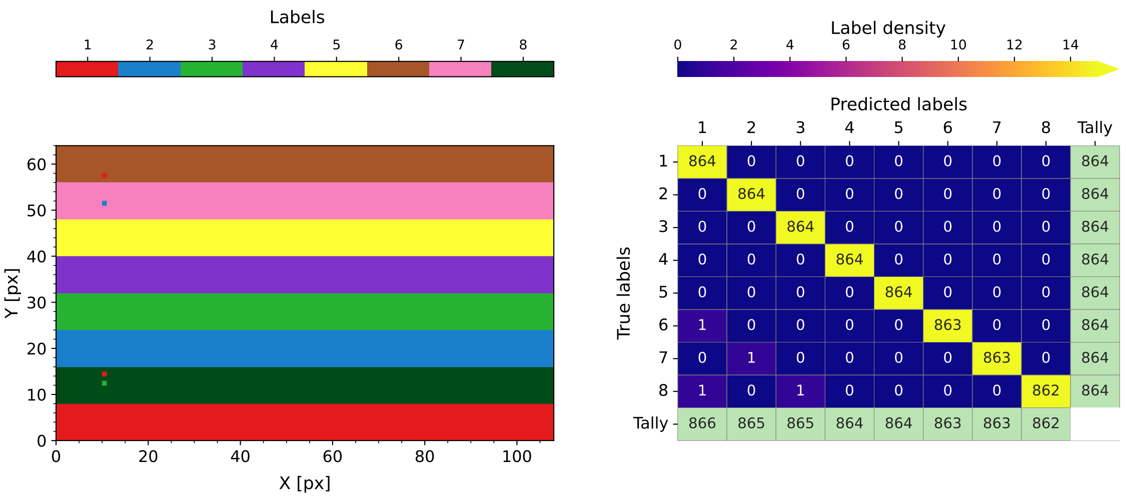


Figure 11: Example of a profile with an added cosmic ray and a dead pixel flat profile.

5.2. Results.

For a first test, we added no dead pixels and 70 cosmic rays, which is around 1% of the total number of pixels (6912). Figure 12 displays the results for this scenario. The color map on the left shows the first indications of clustering mistakes, with four pixels having a different color than the rest of their layer. The confusion matrix on the right shows this as well, where four cells off the main diagonal are nonzero.

Though the results are not perfect, 4 mistakes among 6912 examples, traded for a reduction to only 8 profiles (a factor of 864), are completely negligible. The mistakes will not be noticeable after the centroids are retrieved, since hundreds of similar profiles will be averaged and the dominant behavior will be that of all the correctly assigned examples, and more specifically, the ones with no cosmic ray impacts. Therefore, the clustering is still very adequate, and the output representative profiles, still useful.



True labels \ Predicted labels	1	2	3	4	5	6	7	8	Tally
1	864	0	0	0	0	0	0	0	864
2	0	864	0	0	0	0	0	0	864
3	0	0	864	0	0	0	0	0	864
4	0	0	0	864	0	0	0	0	864
5	0	0	0	0	864	0	0	0	864
6	1	0	0	0	0	863	0	0	864
7	0	1	0	0	0	0	863	0	864
8	1	0	1	0	0	0	0	862	864
Tally	866	865	865	864	864	863	863	862	

Figure 12: Color map (left) and confusion matrix (right) for the case with 70 cosmic rays. The errors can be clearly identified both on the color map and the matrix.

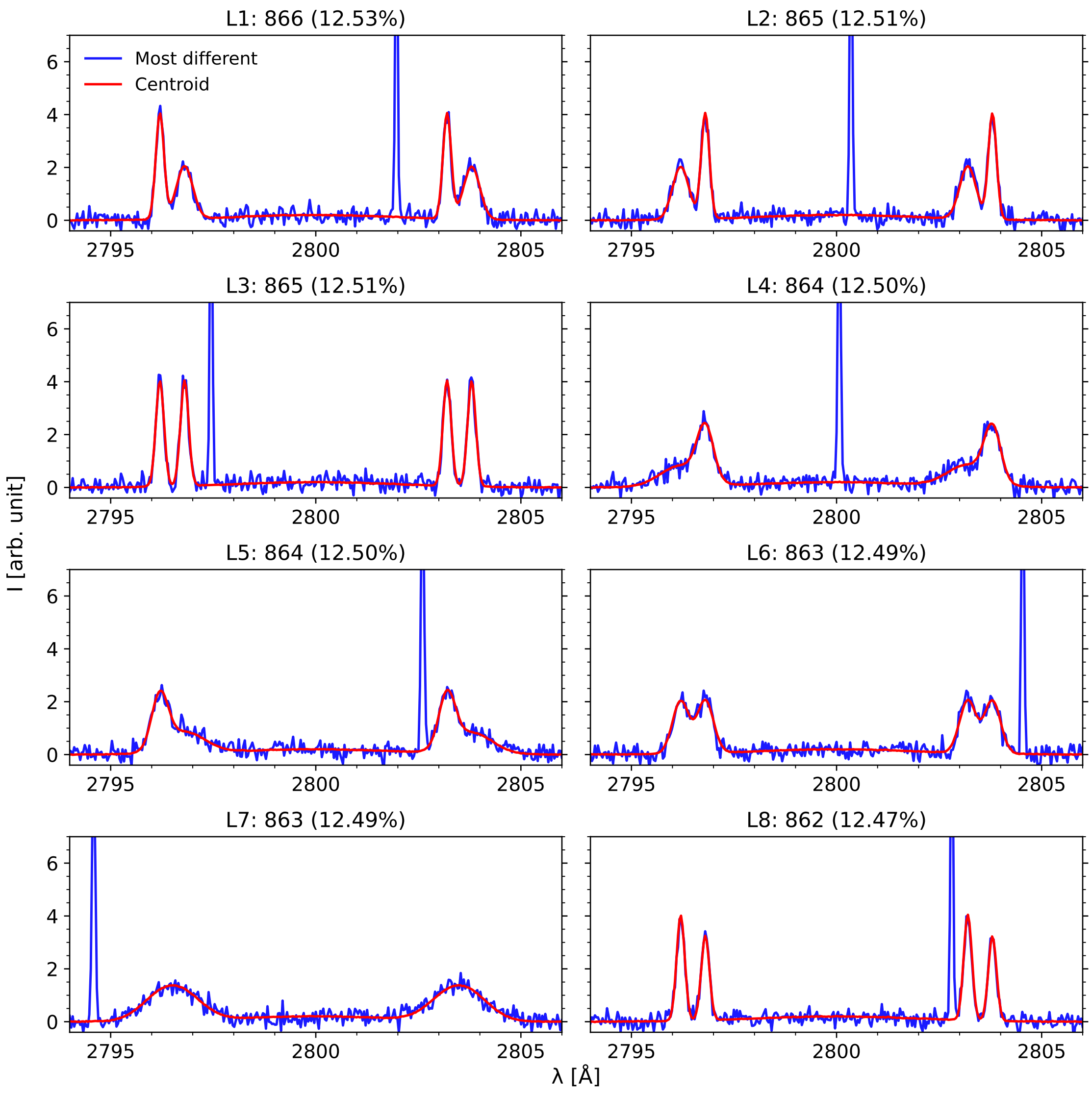
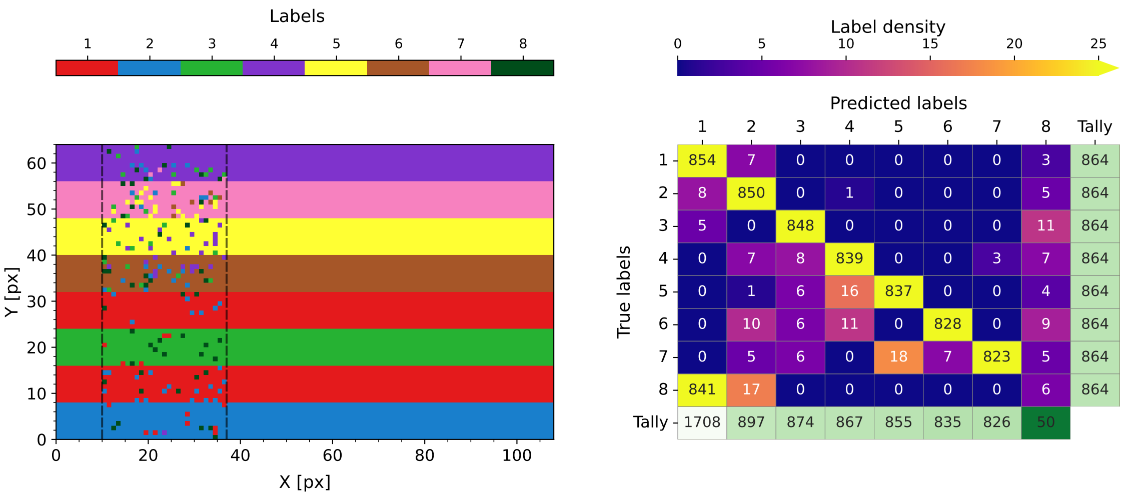


Figure 13: Most different profiles (blue) along with their centroids (red) for the case with 70 cosmic rays. The cosmic rays are clearly visible on the blue profiles.

The viability of the clustering can also be seen in Figure 13: the centroids clearly still look like the original profiles. In this graph, the most different profile is always one with a cosmic ray. However, we see no evidence of the wrongly clustered examples being detectable via the most different graphs. This is curious, since the wrongly clustered examples, which contain cosmic rays, differ from the centroid both in the cosmic ray and in the general shape of the profile, so logically they should be the most different ones. To further explore what is going on, let us add a very significant amount of cosmic rays, specifically, 25% of the total number of profiles.[4]

Figure 14 shows the great decline in clustering quality. 1728 cosmic rays have been added. The edges of the area in which we contaminated the profiles with cosmic rays are indicated on the color map by the dashed black lines. Every pixel inside them contains exactly one cosmic ray.

We see that all of the individual mistakes are caused by the presence of rays, though many examples inside this area are still clustered correctly. Additionally, there is a big systematic error as two full stripes of the color map have been assigned to the first cluster (shown in red), mixing the information of two of the original profiles and almost doubling this cluster's expected cardinality. Further, the last cluster contains 50 examples (shown in dark green) that are exclusively located within the confines of the ray region. This is a step back in the clustering, as valuable information has been lost in the merging of two different profiles, not to mention the fact that the last centroid will not be representative of any real behavior either.



True \ Predicted	1	2	3	4	5	6	7	8	Tally
1	854	7	0	0	0	0	0	3	864
2	8	850	0	1	0	0	0	5	864
3	5	0	848	0	0	0	0	11	864
4	0	7	8	839	0	0	3	7	864
5	0	1	6	16	837	0	0	4	864
6	0	10	6	11	0	828	0	9	864
7	0	5	6	0	18	7	823	5	864
8	841	17	0	0	0	0	0	6	864
Tally	1708	897	874	867	855	835	826	50	

Figure 14: Color map (left) and confusion matrix (right) for 1728 cosmic rays (25% of the total). The dashed black lines on the color map represent the limits of the cosmic ray affected area.

Figure 15 shows the six most different profiles for each cluster in various colors. The centroids of the clusters are shown in black. Comparing it with Figure 14, an inevitable question appears: if there are so many wrongly clustered examples, why are they not visible as the most different profiles (other than in the last cluster)? If each cluster contains: 1) a bulk of profiles with no cosmic rays, which define the "correct" spectral behavior for each cluster and the shape of the centroid; 2) a significant amount of the same type of profile, but with a cosmic ray, and 3)

[4]Though this might seem excessively pessimistic, it is not unprecedented, because "IRIS passes through the South Atlantic Anomaly (SAA) on a regular basis." (cited from IRIS data notes, LMSAL, 2022, included in the references). The SAA is a region in Earth's magnetosphere that has an increased flux of energetic particles due to the solar wind and cosmic rays. Observations with such large quantities of impacts are usually discarded, though.

some completely different profiles that, moreover, also have a cosmic ray; we should expect the most different examples to come from the third category instead of the second.

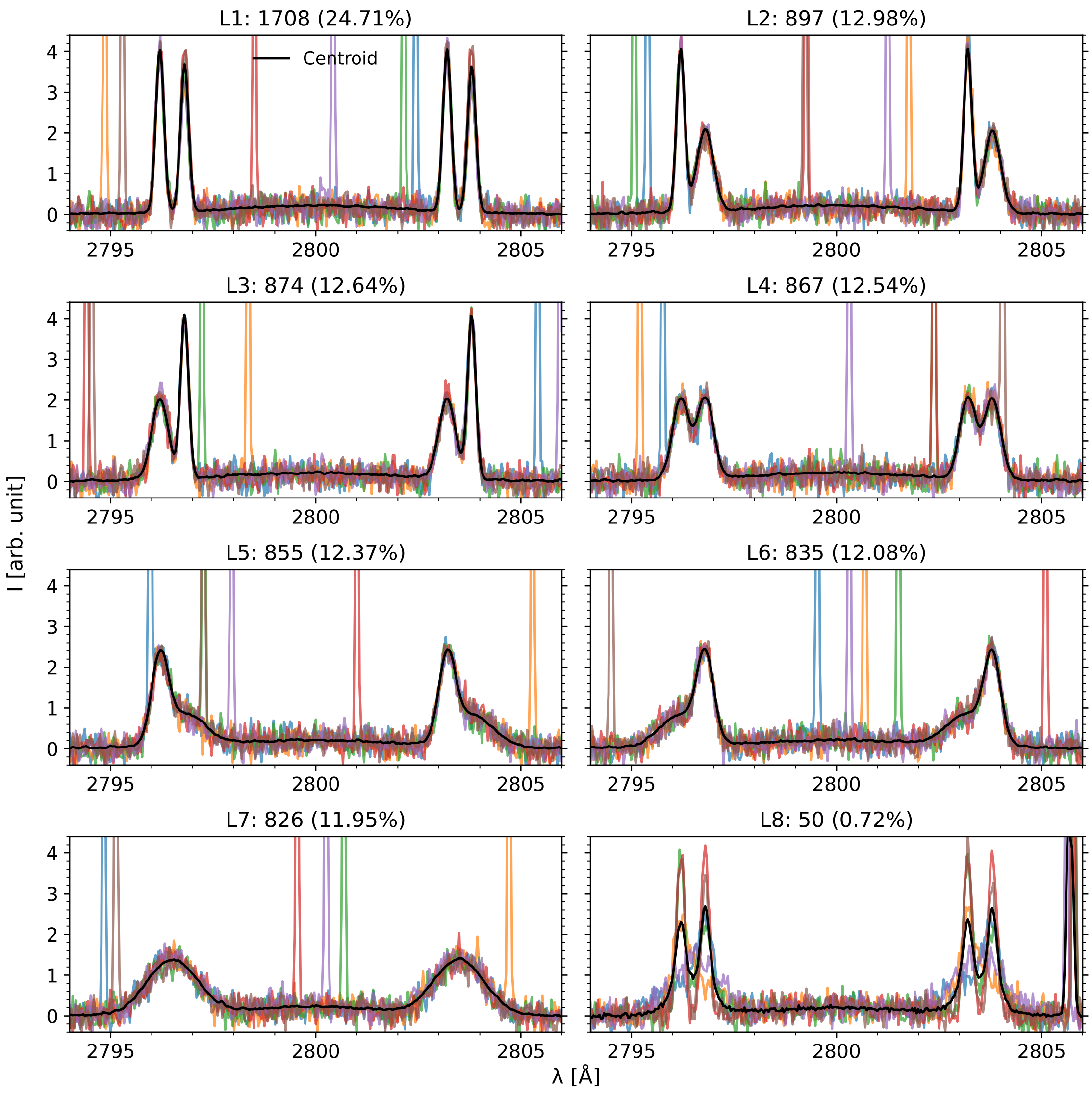


Figure 15: Six most different profiles (various colors) for each cluster, for the case with 1728 cosmic rays. Centroids are shown in black.

Figure 16 shows a sample of the errors in the clustering that sheds some light on this issue. The solid black line represents the centroid of one of the clusters. The colored lines are some examples that were originally a different profile, but have been incorrectly assigned to said cluster. Finally, the dashed black line is one of the original profiles, clearly showing the common tendency among these errors in the absence of the cosmic rays.

This is why the most different profiles look unexpected: the errors are caused by the cosmic rays landing directly on the h&k lines and twisting the information contained in them by making them look like completely different profiles in the eyes of the algorithm. Thus, the standard deviation that the cosmic ray introduces in these cases is not being fully accounted for, because some of it is mistaken as part of the signal, while the standard deviation of a ray in the wings is completely taken into account, ending up as one of the most different. The reason why we have so many of these errors is because we have added a great amount of rays, and therefore it is statistically bound to happen for several pixels in the raster.

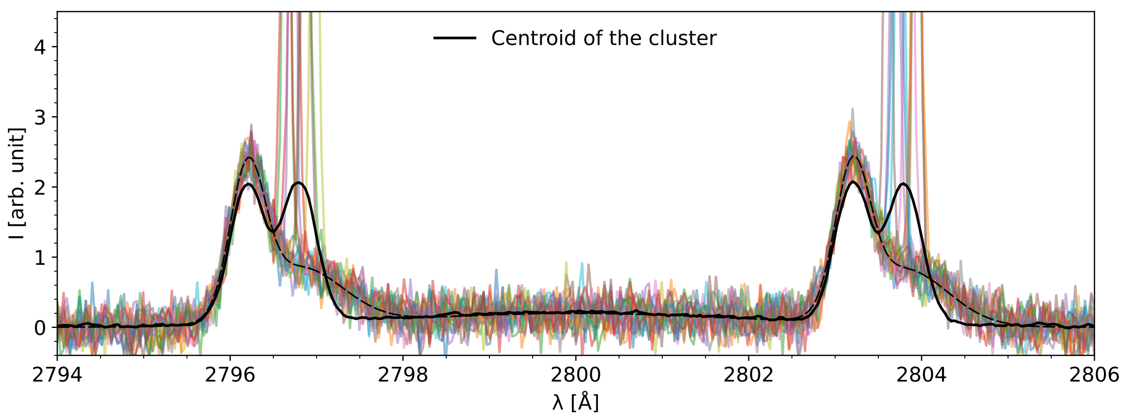
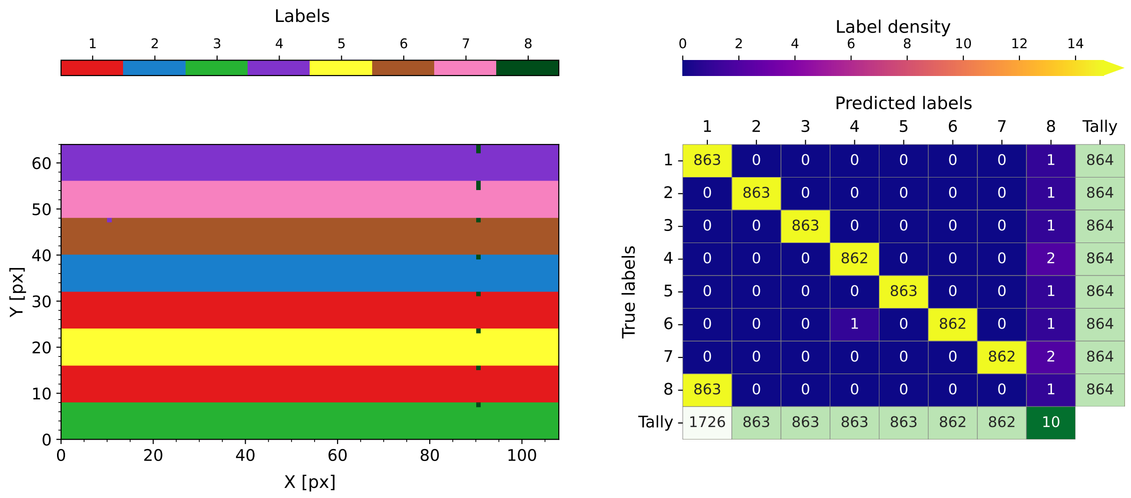


Figure 16: Wrongly clustered examples (various colors) plotted against their assigned cluster's centroid (solid black line) and the original profile that the errors actually had (dashed black line).

To sum up, the presence of cosmic rays severely hinders the proper functioning of k-means when they affect the spectral lines in our dataset. The errors that ensue are both problematic in terms of loss of information and hard to detect graphically (since the spatial distribution of profiles is unknown in a general case). It stands to reason that one must try to remove these outliers.

The idea stays the same when dealing with dead pixels. Because of the evident similarity between all dead pixels, k-means will surely cluster them together, inevitably joining two different profiles in the same cluster.

Figure 17 shows a case with only 10 cosmic rays and 10 dead pixels, which shows the aforementioned issue. Since the 10 dead pixels are extremely favorable as a cluster (the inertia in it is zero), two profiles are forced into the first cluster.



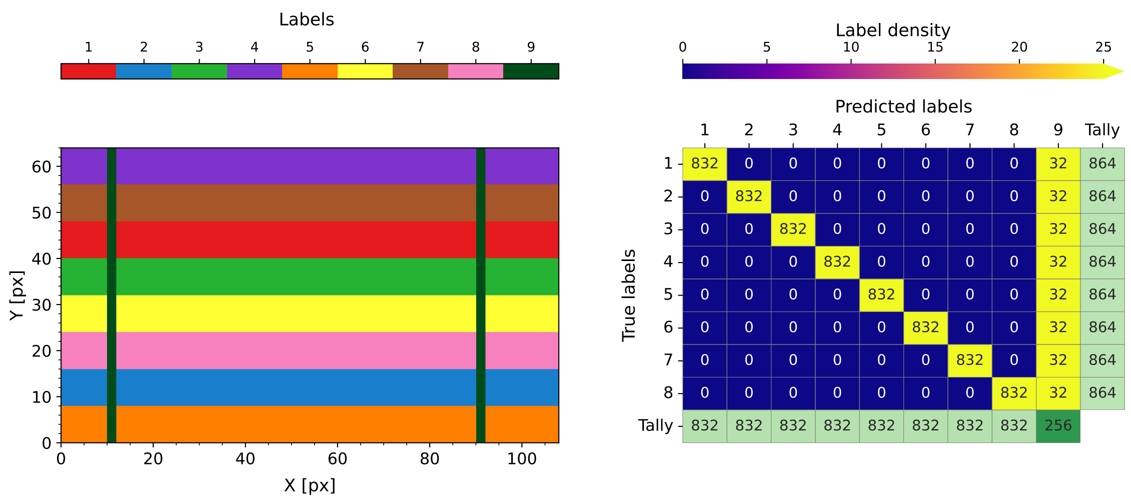
	1	2	3	4	5	6	7	8	Tally
1	863	0	0	0	0	0	0	1	864
2	0	863	0	0	0	0	0	1	864
3	0	0	863	0	0	0	0	1	864
4	0	0	0	862	0	0	0	2	864
5	0	0	0	0	863	0	0	1	864
6	0	0	0	1	0	862	0	1	864
7	0	0	0	0	0	0	862	2	864
8	863	0	0	0	0	0	0	1	864
Tally	1726	863	863	863	863	862	862	10	

Figure 17: Color map (left) and confusion matrix (right) for 10 cosmic rays and 10 dead pixels. The dead pixels are alone in cluster 8 and two different profiles have been fused.

This is further proof that once the outliers start piling up, efforts need to be put towards masking them so they do not affect the clustering of the other examples.

5.3. Masking. New results.

The previous section has proven that, if cosmic rays are not masked, the results can be misleading. It is clear why identifying and masking anomalous examples is paramount in any ML application. In this section, we will test the accuracy of k-means when defective pixels are added, but masked. For this last test, we added 128 cosmic rays (two entire columns) and 128 dead pixels. Figure 18 shows the clustering results for this run.



	1	2	3	4	5	6	7	8	9	Tally
1	832	0	0	0	0	0	0	0	32	864
2	0	832	0	0	0	0	0	0	32	864
3	0	0	832	0	0	0	0	0	32	864
4	0	0	0	832	0	0	0	0	32	864
5	0	0	0	0	832	0	0	0	32	864
6	0	0	0	0	0	832	0	0	32	864
7	0	0	0	0	0	0	832	0	32	864
8	0	0	0	0	0	0	0	832	32	864
Tally	832	832	832	832	832	832	832	832	256	

Figure 18: Color map (left) and confusion matrix (right) for the masked case. 9 clusters are now in play, the ninth one being the masked pixels only. The results have gone back to showing no errors.

Masking was performed with a very simple function that checked if the example's maximum was above a certain threshold, or if any of its features was a very large negative number, and substituted the entire example for a flat profile at 0. As stated before, flat profiles will undoubtedly be clustered together, so when masks are in play, k-means is asked to find 9 clusters. Given the nature of this ninth cluster, it is safe to say it does not impact the total inertia any more than a factor given by machine precision, as the centroid will be exactly equal to all of the examples in the cluster. This is, of course, negligible compared to the inertia of other clusters.

After adding the mask and performing the clustering again, the results were much more positive —the defective pixels all fall into the same cluster, as they have become identical, and the rest of the clustering is akin to that of the previous cases.

Figure 19 shows the most different profiles for each cluster, which once again look like the ones in the case for noise. The last cluster contains identical examples, so the most different one is perfectly superimposed with the centroid.

The mask defined in this work could be considered rather rudimentary, and rightfully so. Its construction was purely based on the need for analysis of the clustering and not aimed at nuanced detection of outliers. For this purpose, many procedures exist, but one can always assume some cosmic rays will be missed in the masking or removal process. Cases with 10 cosmic ray impacts ($\sim 0.1\%$ of the total) were also tested and, fortunately, no errors were detected in those runs.

In conclusion, the act of masking is both a necessary and a beneficial step in a dataset where remarkable outliers appear, as long as their characteristics are identifiable. This is not only the case for clustering, or k-means, but for all ML purposes.

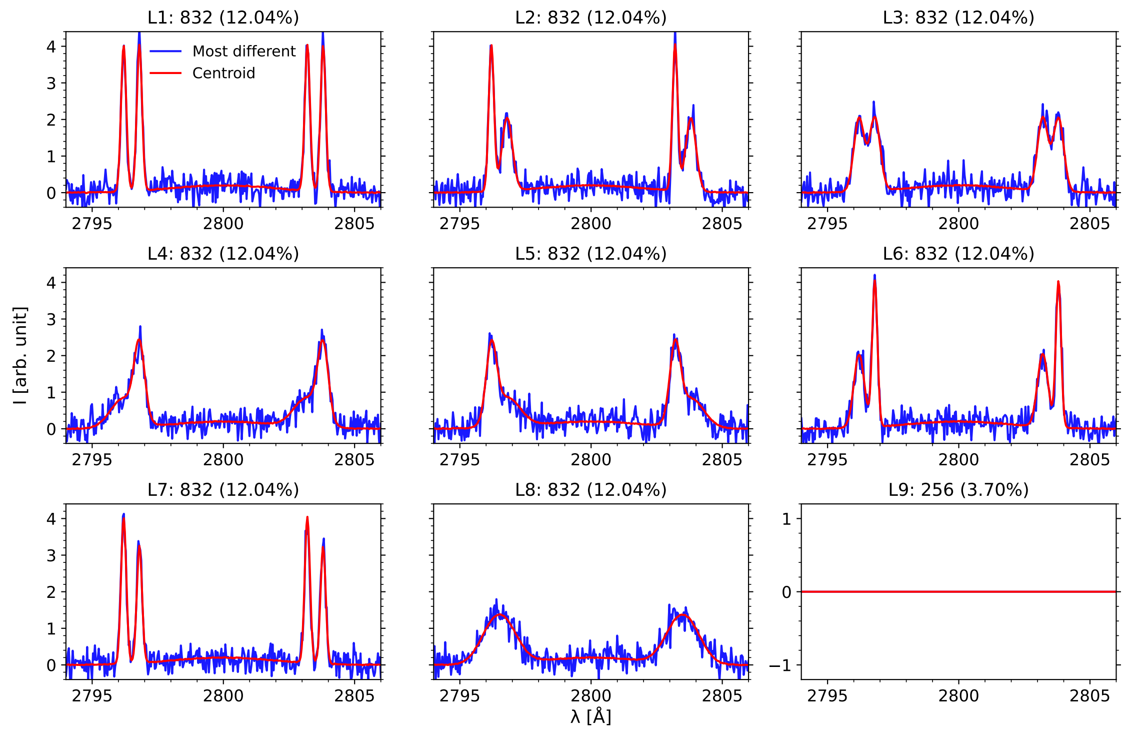


Figure 19: Most different profiles for the masked case. They look like those in the case with noise only, since cosmic rays have been completely filtered out and masked. Knowing how many defective pixels we have introduced of each type, we can tell how many examples of each profile were affected, given the method of distribution. It is confirmed that 256 pixels were affected in this run (128 cosmic rays and 128 dead pixels), equally affecting all profiles.

6. Hierarchical clustering methods

Los métodos de *clustering* jerárquico realizan la misma tarea que k-means pero de una forma distinta; concretamente, los métodos aglomerativos comienzan considerando todos los ejemplos como *clusters* y unen en cada paso los dos grupos más cercanos entre sí, hasta que todos los ejemplos están en un solo grupo. Para determinar qué dos *clusters* están más "cerca", se utilizan distintos criterios de enlace. A menudo se muestran las últimas uniones en un dendrograma, que es un tipo de gráfico que muestra las distancias a la que se realiza cada fusión entre grupos. Este entonces se emplea para determinar el número óptimo de *clusters* bajo la suposición de que este se encontrará donde haya la mayor separación entre uniones.

El criterio 'ward' resultó ser muy comparable a k-means, siendo ligeramente peor al añadir rayos cósmicos. Los resultados del resto de métodos fueron pobres, teniendo solo un desempeño equivalente en el caso de control.

6.1. General description and different methods.

Agglomerative hierarchical clustering methods begin with each example in its own cluster, called *singleton cluster*, and iteratively *merge* clusters until there is only one.[5] The first merging happens between the two clusters that are closest together,

[5] Agglomerative methods are also often called inductive methods. Deductive methods, in which all examples start in one cluster and slowly separate, also exist, but were not used in this work.

and each successive merging follows the same rule, therefore each consecutive merging will have a higher distance associated. Sometimes the distance at which a merging happens is called the *cophenetic distance* (Sokal and Rohlf, 1962). Once the whole merging scheme is finished, the user chooses an intermediate distance at which to extract the clustering results, called the *cut-off distance*.

The difference between individual hierarchical methods is the way they determine which clusters are to be merged next, that is, the *linkage criterion*. This should not be confused with the *distance metric*, which is the way the distance between two examples is calculated, and which, in this work, will always be Euclidean. Some linkage criteria are defined as follows: the distance between clusters equals the minimum of all the distances between all examples in both clusters ('single'), the maximum of them ('complete') or the average of them ('average'). These are sometimes biased towards merging some of the largest clusters, in a "rich get richer" manner (see the comment on scikit-learn, 2023a). The criterion that was found to be most useful was 'ward', which uses the Ward variance minimization algorithm (see SciPy documentation, for example: SciPy, 2023). We will revisit this later.

The cut-off distance is commonly chosen as the one in between the two consecutive cophenetic distances that are the furthest apart. This can be visualized in a graph called a *dendrogram*, a generic example of which is presented in Figure 20. The basic principle is that the cophenetic distance at which the mergings happen will be very low while the examples joined are very similar to each other, and will suddenly become very large as completely different examples are joined together. Since the clustering is hierarchical and the merged clusters are always the ones with the lowest distance out of all the possibilities, there should theoretically be a turning point where the distance increases greatly between two specific mergings.

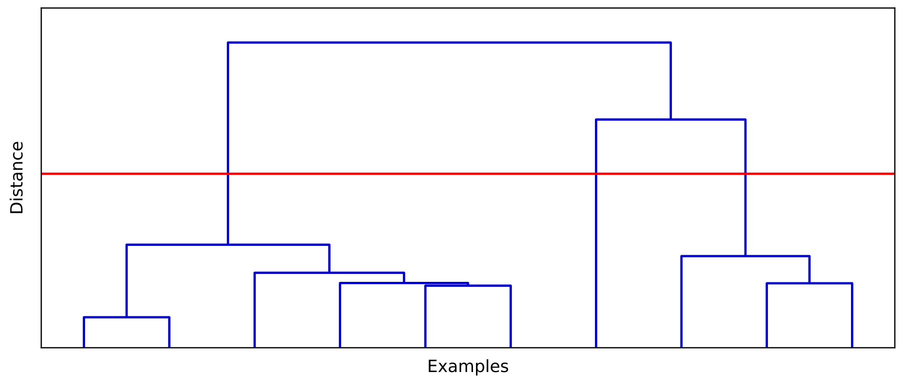


Figure 20: Example dendrogram. The horizontal red line shows a possible cut-off distance.

Figure 20 shows the general anatomy of a dendrogram: vertical lines represent clusters, and horizontal lines that join two vertical ones represent mergings. The height in the y axis indicates the distance at which the merging happens, and the horizontal red line shows a possible cut-off distance. Plotting any horizontal line can tell us how many clusters there are at the corresponding distance by counting how many vertical lines intersect it. In this example, the cut-off distance is chosen where there are 3 clusters left. We see that some of the mergings (horizontal blue lines) happen very close together, indicating that those clusters had a similar level of likeness, and some others are further apart. The cut-off distance will likely be chosen in one of these last instances, but the specifics of it depend on the dataset and this method is not always the most reliable.

Though this is the usual way to get the number of clusters, in this work the distance will be chosen so that the algorithm finds 8 clusters (or 9 when adding an extra cluster that contains all the masked profiles). Thus, the results will be judged based on the accuracy of the clustering. For some cases, we will mention how different the cophenetic distances were between the correct number of clusters and the next merging, which will indicate how favorable it is to have that number of clusters.

Several linkage criteria are supported both by the SciPy library and scikit-learn. We found no discernible differences in performance between the same criterion in different libraries, so they will be used indistinctly.

6.2. Results in contrast with *k*-means.

For the sake of brevity, only the color maps and dendrograms will be discussed for the hierarchical methods.

Figure 21 shows two different dendrograms, both for the ‘ward’ criterion: the one on the left is obtained for the control case and the one on the right corresponds to the case with noise. Both dendrograms show the last 10 mergings before all the examples are fused together in a single cluster. We have indicated no units for the distance between clusters because the units depend on the criterion used and are not relevant for the analysis.

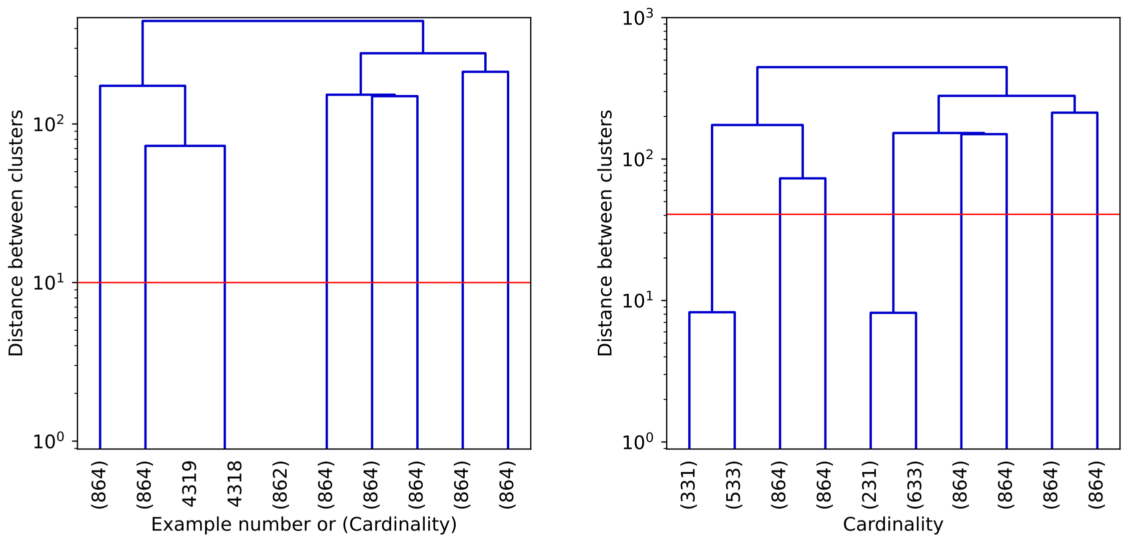


Figure 21: Dendrograms for the ‘ward’ criterion, for the control case (left) and the case with noise (right). The horizontal red line represents a distance at which there are exactly 8 clusters.

The dendrogram for the control case is peculiar in that the first two mergings, the lowest ones, occur for exactly 0 distance, and so do all the previous ones (in this Figure, the 0 value for distance does not lie on the x axis, but at $-\infty$, so the first two mergings are simply not visible). This it is the epitome of the sought after large difference in distance between two mergings. In this case, it is ideal to have 8 clusters, and the color map results (not shown here) proved that the hierarchical clustering was able to consistently cluster all 8 profiles separately with no confusion. As a final note, we see that after merging the two examples (shown as numbers without parentheses) with the cluster that contains 862 examples, all clusters contain 864 examples.

When noise was added, the dendrogram on the right of Figure 21 was obtained. This dendrogram shows a much tighter set of mergings (the previous ones are not below the 10^0 mark, they are just not represented here for ease of interpretation), which implies that 'ward' is having a harder time differentiating the profiles, but the results were still completely accurate for this criterion (see Figure 22). After adding up all the cardinalities for the two pairs of clusters that merge completely shortly before the red line threshold, each cluster's cardinality comes up to 864.

From now on, we will limit the analysis to comparing the 'single' and the 'ward' criteria. The 'single' criterion will act as a representative for all non-'ward' criteria, as their behavior is very similar.

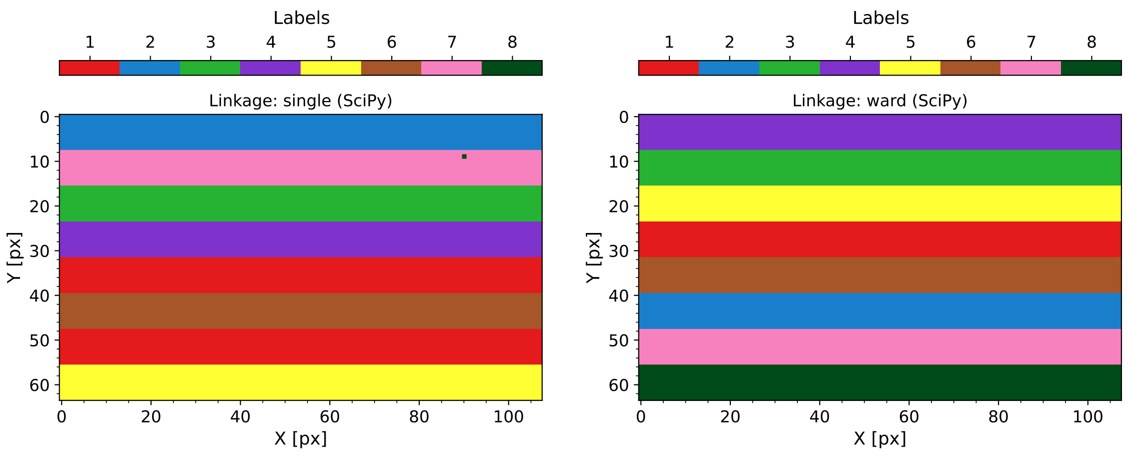


Figure 22: Color maps for the case with noise. The 'single' criterion (left) is all right other than the problematic mix-up of the two profiles in red and the dark green singleton cluster, while 'ward' (right) yields perfect results.

The clustering results for the run with noise are shown in Figure 22. On the left, we see that the 'single' criterion is incapable of isolating 2 of the 8 profiles and has a cluster with only one example (the dark green pixel on the pink stripe), in line with the aforementioned "large cluster" bias, while the 'ward' criterion holds up to the k-means standard.

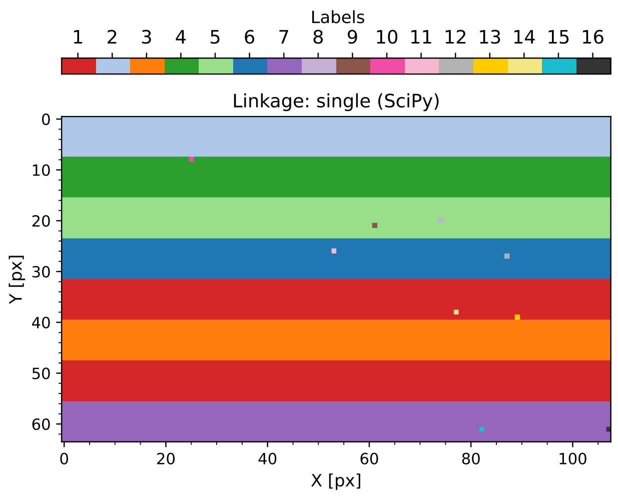


Figure 23: Color map for the 'single' criterion, with noise, at the point where 16 clusters were left.

It is not necessary to limit ourselves to a strict 8 clusters, though. Another attempt may be made with a larger number of clusters, hoping that, while the two clusters that were merged previously might be subdivided, there is a distinction between them and a successful grouping is still salvageable. Unfortunately, this was not the case as seen in Figure 23, where the two problematic profiles are already merged. The results in said figure also prove that the situation for 8 clusters will not be very promising, even if we lacked Figure 22, as the only possibility is to continue merging clusters.

Predictably, adding defective pixels did not help the non-'ward' criteria. For 10 cosmic ray impacts, methods such as single now exhibit a very high-cardinality cluster encompassing most of the pixels whereas the cosmic ray impacted examples are singleton clusters (Fig. 24). On the other hand, 'ward' begins to show a few mistakes. In a good portion of our tests, 'ward' still had a perfect clustering in this situation, but not always. This is an indication that it might not be as robust as k-means when outliers are present.

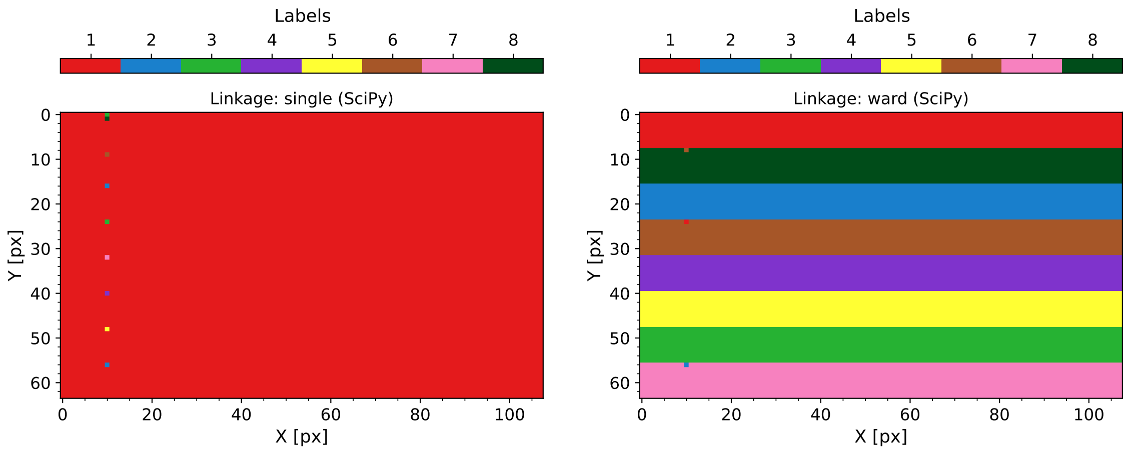


Figure 24: Color maps for noise and 10 cosmic rays. The 'single' criterion now fuses most of the profiles into one cluster, while 'ward' shows its first mistakes.

Figure 25 depicts the clustering results with 70 cosmic rays. In this scenario, 'ward' shows several mistakes, having incorrectly clustered many of the cosmic ray impacted pixels. These results now clearly put 'ward' a step behind k-means, because the latter made fewer mistakes than the former with this amount of anomalous pixels. The 'single' criterion, in contrast, shows exactly the same propensity to singleton clusters.

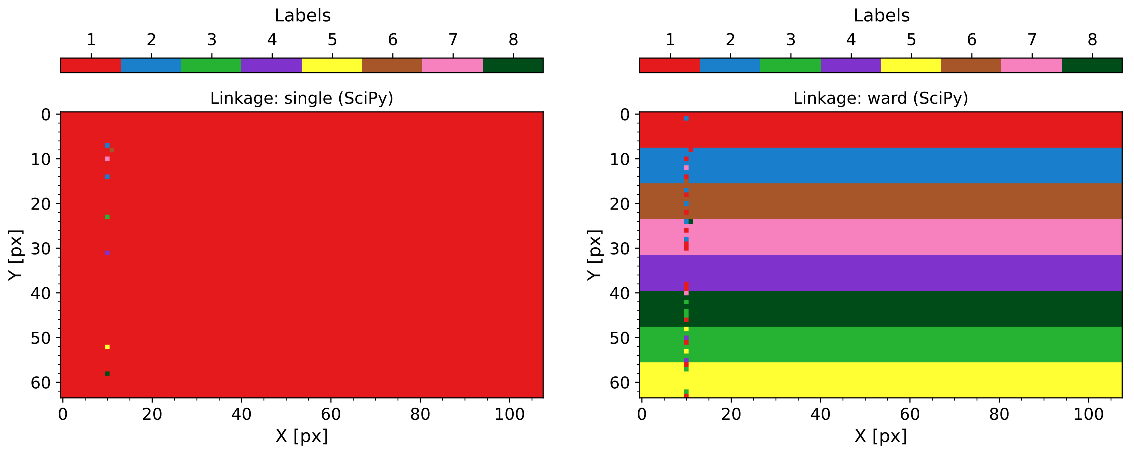


Figure 25: Color maps for noise and 70 cosmic rays. The 'single' criterion fuses most of the profiles into one cluster. 'Ward' shows several mistakes.

Figure 26 shows the results for 128 cosmic rays and 128 dead pixels, both of them masked. The trends that emerged when only noise was present are once again reproduced, except for the fact that masked examples are now separated into a different cluster. The 'ward' criterion continues to do a good job, but the issues of the 'single' criterion remain unsolved.

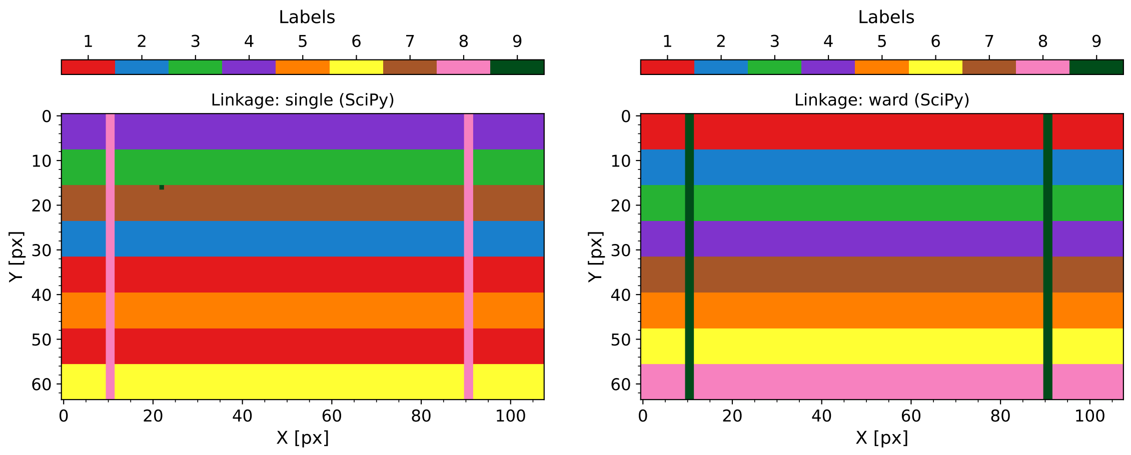


Figure 26: Color maps for 128 cosmic rays and 128 dead pixels, all masked. On the left, 'single' method, and on the right, 'ward' method, showing similar tendencies to the cases with noise.

In summary, some of the hierarchical clustering methods have proven ineffective in dealing with the dataset at hand. The presence of noise alone is able to deteriorate the results, fusing the information of two profiles. The best one out of the considered criteria, however, is very comparable to k-means.

7. Effects of preprocessing

Una parte importante del *Machine Learning* es la preparación de los datos o preprocesado. El preprocesado consiste en modificar los datos originales (*raw data*) con un fin específico, a menudo el de reducir el coste computacional o el de evitar sesgos no deseados. Se analizaron los efectos de dos métodos de preprocesado, el escalado de *features* y el Análisis de Componentes Principales (PCA por sus siglas en inglés).

El escalado demuestra ser adecuado en ausencia de ruido, pero contraproducente en casos subsiguientes, dado que reduce la importancia de la señal e incrementa la del ruido y los rayos cósmicos. En cuanto al PCA, se muestra que es capaz de reducir el número de dimensiones del conjunto de datos sin perder gran parte de la información.

Finalmente, se discuten los aspectos estadísticos de cada situación en términos del número medio de errores y el tiempo de computación medio. Se demuestra que el PCA reduce el tiempo computacional sin causar grandes estragos en las tasas de error, mientras que el escalado aumenta el tiempo computacional y los errores, siendo inadecuado para este tipo de datos ruidosos. Ante estos resultados, se consolida la gran importancia del enmascarado.

Performing any kind of ML with raw data is generally discouraged, because it can lead to inaccurate results. Hence, data preparation is an integral part of ML. Some of the most common techniques include data scaling and dimension reduction. The present section is a discussion of the general effects that preprocessing had on the data, both in terms of loss of accuracy and reduction of computational time.

7.1. Feature scaling.

Feature scaling is the act of modifying the values in the dataset so that all features have a similar range; this is a useful method when some features span wider ranges than others. If they are left untouched, the linear nature of the algorithms would treat these characteristics as though they were much more important and representative than the rest. While this can be true sometimes, it is not always the case or at the very least it is unknown whether or not it applies. Feature scaling can help remove this bias.

Our original profiles are such that some features are practically zero for all examples and others range from 0 to 4, approximately. While it is not extremely significant, feature scaling could still be a good idea. A short analysis for the feasibility of scaling in the different cases considered in this work will now be provided.

In order to perform the scaling at hand, each feature was standardized, namely, its mean was subtracted and then it was divided by its variance. Figure 27 shows the rough effect of this process for the control case, the top panel being the original profiles and the bottom one the scaled ones. One of the important differences between both is that, where all 8 original profiles are hardly indistinguishable (in the middle and far ends of the wavelength spectrum), the scaled profiles split, making them that much more differentiable. For instance, the green and red scaled profiles show the opposite behavior (when one is up, the other one is down), which could aid k-means in telling them apart.

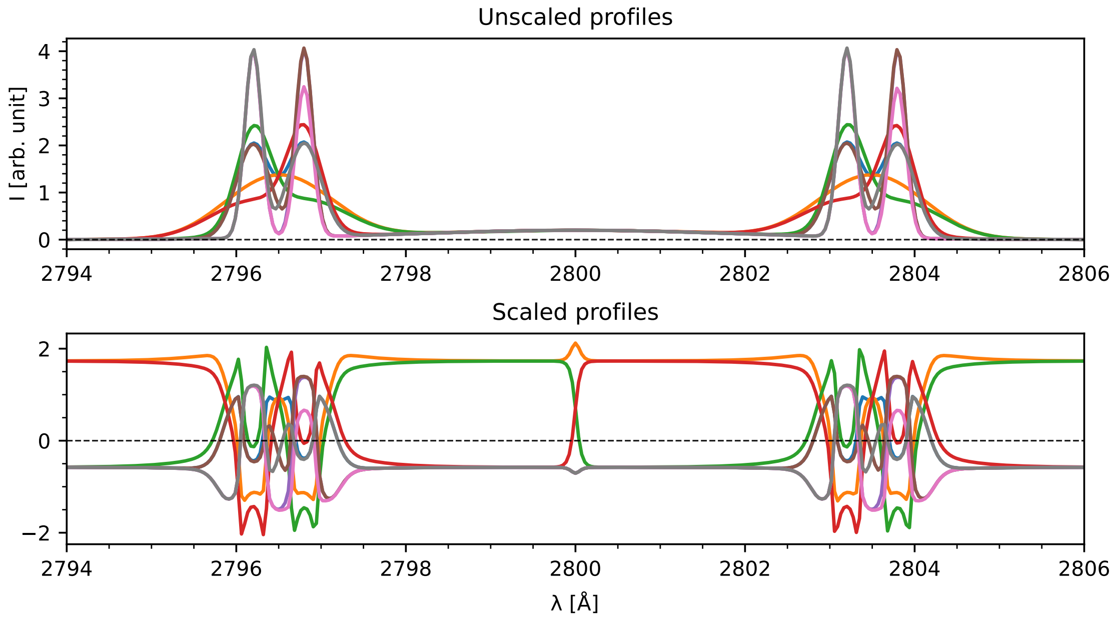


Figure 27: Effect of scaling on the profiles for the control case. We have omitted the y label on the bottom panel, as negative intensities hold no physical meaning.

Performing this type of scaling effectively removes all physical meaning from the data until they are returned to their original range. Therefore, we have omitted the y label on the bottom panel, since the values on the scaled profiles (which are sometimes negative) are not considered intensity.

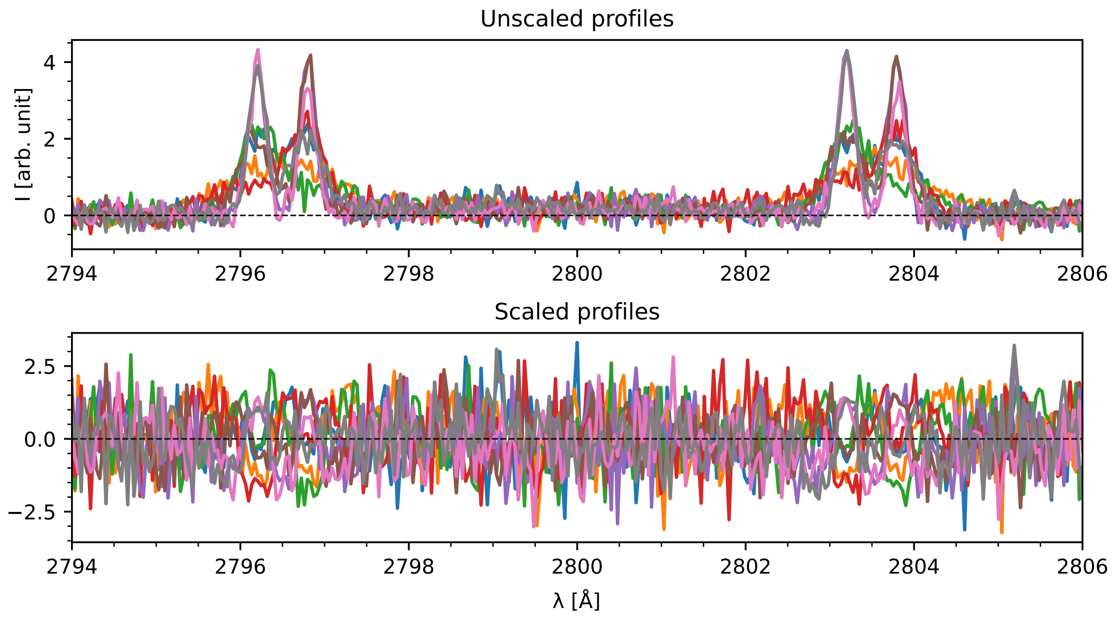


Figure 28: Effect of scaling on the profiles for the case with noise. We have omitted the y label on the bottom panel once again due to the negative values having no physical meaning.

The situation changes when noise is added, however. The outcome of feature scaling in this case can be seen in Figure 28. Upon forcing all features onto the same range, the noise-dominated features and the signal-dominated ones become equally important, and instead of becoming more distinguishable, the profiles become more easily confused.

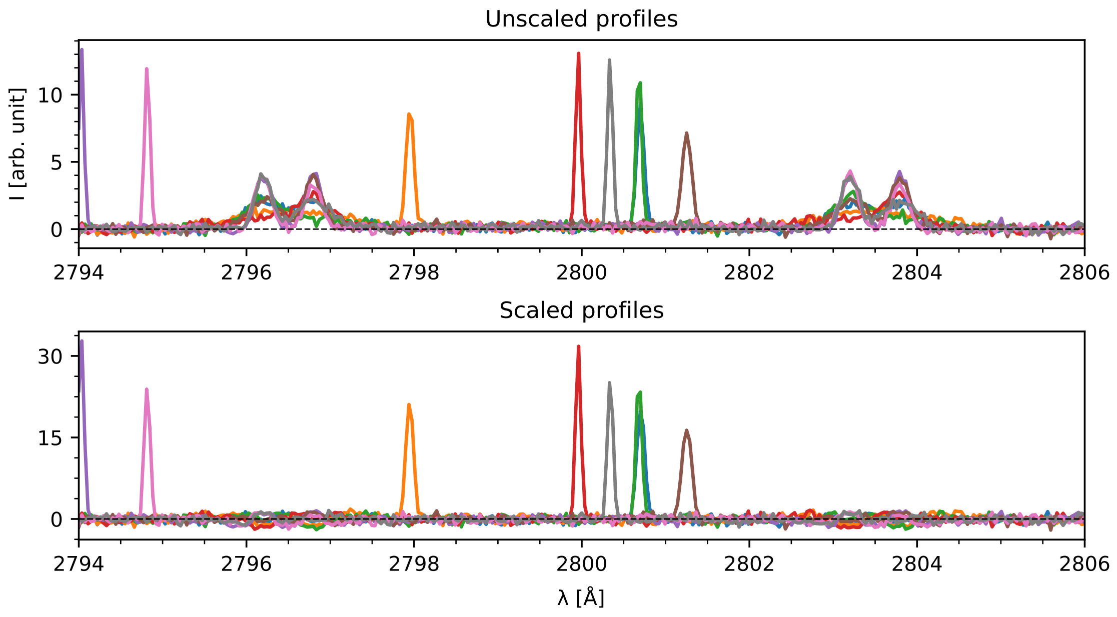


Figure 29: Effect of scaling on the profiles for the case with cosmic rays, showing that scaling the data as-is will enlarge the rays. The bottom panel has no y label for the same reason as Figs. 27 and 28.

Finally, Figure 29 shows the effect of scaling on the cosmic rays. The rays are complete outliers, affecting at most 1% of the profiles, and since this type of scaling is done feature by feature, each feature will contain at most 1 or 2 rays in the entire dataset. Hence, the noise-dominated features (that is, those on the wings of the h&k lines) will have a very low standard deviation despite the presence of a cosmic ray. This feature will then be scaled, dividing by a number smaller than 1, which enlarges the rays instead of reducing them. All in all, the meaningful signal in the line peaks has been scaled down to the same level as the noise and the least relevant characteristics have been amplified. This is obviously not a desirable outcome, and it reinforces the importance of masking the outliers in the data before attempting the clustering.

7.2. Principal Component Analysis.

Dimension reduction is used in very large datasets in order to greatly reduce the computational cost and time it takes for the algorithm to converge. A common method to do so is Principal Component Analysis, or PCA for short. PCA uses the concept of variance to transform the data to a new phase space, in which each direction is called a "Principal Component". Let us exemplify this idea:

Figure 30 shows a simple 2-dimensional distribution and its 2 principal components after performing PCA. The concept is easy to understand in this case: the original features are the x and y values of each point. However, there is a direction in this space (the Cartesian plane) that contains the widest spread of points, that is, the most variance, indicated by the larger black arrow. The other Principal Component, for a total of 2 (to account for all the dimensions of the phase space), is perpendicular to the first one. This is akin to a coordinate transformation, in this case, a simple rotation of the axes.

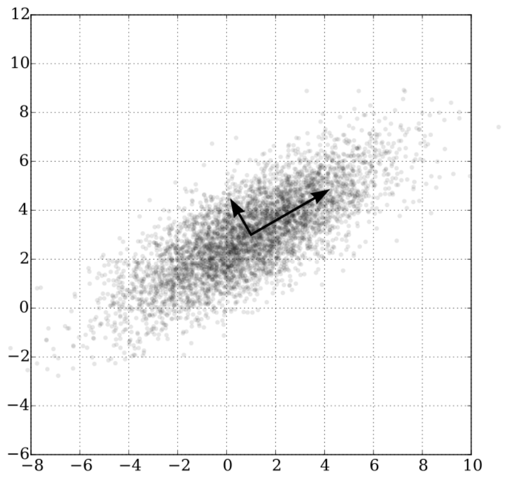


Figure 30: Diagram showing the eigenvectors for PCA in a Gaussian scatter in 2 dimensions.

Further, we can see that each component is a linear combination of both features. Finally, if we decided to eliminate the second component, the final data would be the *projection* of all these points along the line that the first Principal Component defines. While we have surely lost some information, the amount of data values has been cut in half. PCA can be extremely significant in large datasets that contain much redundancy.

In a more general case, with N features, the phase space is N-dimensional, but the concepts are the same: PCA transforms the data by linearly combining features. The principal components are chosen recursively, with the first one being the direction in the original phase space that explains the most variance, and each successive one being the one that maximizes the remaining variance while being orthogonal to all previous components. This creates a new system of orthogonal axes that have a specific order based on the amount of information that they provide, much like the previous 2-dimensional example.

Logically, following this order, every component explains less variance than each of the ones before. In light of this, one can choose to eliminate a number of components starting from the end, knowing that most of the information is still contained in the first ones. It is important to remember that the new components obtained through the PCA method are not features directly extracted from the dataset, but linear combinations of them. In the case of the spectra, we cannot say that each component corresponds to the intensity at a certain wavelength, but rather an indication of a certain set of values of intensity.

The number of features in our dataset is 327, which is the number of individual wavelengths for which we have values of intensity. In the control case, only 5 Principal Components (about 1.5% of 327 possible components) are needed to explain 100% of the variance. This is proof that most of the information in the original dataset is redundant.

Nevertheless, after adding noise, this percentage is no longer reasonably achievable. Aiming for close to 100% of the variance would eliminate almost no dimensions, and to retrieve just over 90% of the variance, around 218 components are needed.

Other than dimensionality reduction, Principal Component Analysis can also be used to filter out random noise. For this specific purpose, very few components were needed, as the inverse transform of the PCA in the case with noise returned fully identifiable profiles that matched the original ones reasonably well (though they were visibly imperfect) with as few as 5 components, like in the control case. However, in order to achieve good clustering results, we noticed that more components were required.

7.3. Statistical comparison between preprocessing schemes.

In order to better understand how the preprocessing affected k-means' performance, we tested the algorithm on 300 runs, each with regenerated noise and cosmic rays (if applicable), all under the same conditions of tolerance, number of initializations, etc. We quantified the average number of errors made with respect to the expected layered distribution and the average time taken by k-means for each case. We considered the control case, the case with noise in the 5-10% range with respect to the signal and the case with 70 (1%) cosmic rays. Each of these cases was analyzed for the following situations: no preparation, only feature scaling, only PCA at 80% of variance explained, and combinations of both PCA and scaling. The results are shown in Table 1.

	Control		Noise		Cosmic (70)	
	Errors	Time [s]	Errors	Time [s]	Errors	Time [s]
No prep	0	1.83	0	3.47	5.25	3.59
PCA 80%	0	0.42	0	1.79	5.29	1.72
Scaling	0	1.86	0.07	5.78	>800	5.33
PCA 80% + scaling	0	0.54	18.58	4.23	>800	3.96
PCA 95% + scaling	0	0.58	0.30	4.70	>800	4.69

Table 1: Average errors and time per k-means run for different cases and preprocessing situations.

Table 1 shows very clear trends: Principal Component Analysis consistently reduces the computational time, while introducing little to no errors compared to the case with no preparation. As a curious fact, although the run for 70 cosmic rays with PCA had a slightly higher average error rate than with raw data, the maximum amount of errors (11) was smaller than the one for no preprocessing (14). However, this could merely be a coincidence as any statistical analysis is subject to bias due to the number of observations not being infinite.

On the other hand, feature scaling invariably increases the computational time while either maintaining the previous error rate or worsening it. Singularly large values of >800 are present because the clustering has begun to fuse two of the spectral behaviors together, therefore often making over 800 mistakes as each profile is placed in 864 examples. The minor variations in making additional mistakes were considered negligible in the face of mixing up two profiles, so they were not specifically quantified.

Finally, the combined effect of both PCA and scaling may also be discussed, as they are often used together. For the control case, the use of scaling along with PCA at 80% explained variance represents a small step back in terms of computational time, and adding additional information further increases the required time for convergence. No worsening of the error rates is seen, but nothing can be said about whether or not this makes the examples slightly more differentiable. It could be argued, given the scaled profiles in Figure 27 that scaling can help the clustering in this case, but seeing that it only slows down the process and does not improve the results, it is deemed completely unnecessary. Applying PCA at 80% is the optimal strategy in this case.

As for the case with noise, we see that all the cases with feature scaling contain some wrongly clustered examples, while the ones without it obtain perfect results. Additionally, the results with PCA at 80% and scaling are significantly worse than the rest, suggesting that the loss of 20% of the variance is being aggravated by the inadequacy of the scaling for the dataset at hand. Once again, only PCA at 80% variance comes out on top.

Finally, the case for 70 cosmic rays is problematic in that there are always errors present, but the results become deplorable as soon as scaling is introduced. This is in line with the previously mentioned enlarging of the cosmic rays and noise and reduction of the signal. The glaring issues with this scheme are now definitively proven seeing as two different spectral behaviors are being mixed up, resulting in an irretrievable loss of information.

As previously mentioned, the use of PCA marginally increased the amount of errors for 70 cosmic rays. However, the computational time is halved. In large datasets, this could be significant, as the clustering could take minutes or hours, and the trade-off seems to be quite small. Therefore we also declare this strategy to be the optimal one in this case.

A special mention is due for the control case, where PCA at 80% reduced the time for convergence by a factor of 4. This is due to the very small number of components required to achieve 80% of the variance (as specified in Section 7.1, 5 components account for $\sim$ 100% of the variance), which greatly reduces the number of arithmetic calculations that k-means needs to perform. As per the academic nature of this control case, though, it is not expected that this will be true for any real observations.

In conclusion, we see consistent trends among the data that show that feature scaling is detrimental both in terms of accuracy and time, or at the very least, rather inconsequential. The biggest issues happen when there are cosmic rays involved, which is why masking the outliers is of utmost importance. Conversely, PCA shows time improvements whilst minimally impacting the error rate. This will under no circumstances be true for every Machine Learning implementation or dataset, of course. The particular needs of every application must be studied with care, regardless of how popular the use of certain schemes are. Regardless, the act of masking a small amount of pixels in favor of maintaining the integrity of the information presents no downsides and has the potential to greatly improve the clustering results.

8. Conclusions

En este Trabajo de Fin de Grado se ha analizado el comportamiento del algoritmo k-means y de algunos métodos de *clustering* jerárquico utilizando un conjunto de datos artificial y sencillo al que se le han añadido ruido y píxeles defectuosos, estos últimos tanto enmascarados como sin enmascarar. Además, se ha discutido el efecto del Análisis de Componentes Principales y del escalado por *feature* en los resultados del agrupamiento.

El agrupamiento con k-means ha demostrado ser viable tanto en el caso de control como en el caso con ruido, donde los resultados fueron totalmente satisfactorios. Las diferencias entre ambos casos fueron caracterizadas visualmente y a través de la inercia.

Al añadir píxeles defectuosos, se evidenció la importancia de conocer y tratar el conjunto de datos antes de emplear métodos de *Machine Learning*, dado que en ausencia de una máscara para obviar los ejemplos anómalos, estos tendían a "falsear" la información espectral de algunos ejemplos si se daba el caso de que cayeran sobre las líneas de h&k. Tras aplicar máscaras, los resultados volvieron a ser agrupados a la perfección.

Los métodos jerárquicos, por su parte, han demostrado ser peores que k-means para la mayoría de los casos de este Trabajo de Fin de Grado, a excepción de 'ward', que solo mostró ligeras deficiencias en los casos con rayos cósmicos.

Finalmente, la comparación de las estrategias de preprocesado demostró las ventajas del PCA y los inconvenientes del escalado, reforzando la tesis de que el enmascarado es imprescindible.

In this Bachelor's Thesis we analyzed the behavior of the k-means algorithm and some hierarchical clustering methods using a simple, artificial dataset. The initial data were modified by adding noise and defective pixels, in one case adding masks to conceal the latter. Furthermore, we discussed the effect of Principal Component Analysis and feature scaling on the clustering results.

The results of the control case and the case with noise both exhibit the viability of the k-means algorithm to cluster spectral profiles, proving its utility in reducing the volume of a sample of IRIS-like data for its subsequent analysis.

The addition of defective pixels that differed greatly from the noisy examples took a toll on k-means' performance. Nevertheless, this effect was neutralized by the implementation of a mask function, which ensured that the outlier data points were clustered separately, and the clustering returned to the previous accuracy levels.

In light of the results in this Bachelor's Thesis, k-means is acknowledged as a practical clustering method, given that the data is analyzed and prepped beforehand. This conclusion is backed by the aforementioned examples of scientific articles in which k-means has been used to cluster solar profiles (Panos et al., 2018;

Bose et al., 2019; Nóbrega-Siverio et al., 2021).

The ‘ward’ criteria proved to be almost as useful as k-means, for the most part having an equal performance, but being marginally worse when clustering data with cosmic rays. Other agglomerative hierarchical clustering methods were objectively less accurate than k-means for most of the cases considered.

Finally, while PCA-prepared data took notably less time to converge and contained roughly the same amount of errors as raw data, scaling in this case hindered the clustering, making it take longer and increasing the number of mistakes. This strengthened the argument that masking is essential for the results to be optimal.

References

Alzubi, J., Nayyar, A., and Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, 1142(1):012012.

Arthur, D. and Vassilvitskii, S. (2007). K-Means++: The Advantages of Careful Seeding. volume 8, pages 1027–1035.

Bose, S., Henriques, V. M. J., Joshi, J., and Rouppe van der Voort, L. (2019). Characterization and formation of on-disk spicules in the Ca II K and Mg II k spectral lines. *A&A*, 631:L5.

de la Cruz Rodríguez, J., Leenaarts, J., and Asensio Ramos, A. (2016). Non-LTE inversions of the Mg II h&k and UV triplet lines. *The Astrophysical Journal Letters*.

de la Cruz Rodríguez, J., Leenaarts, J., Danilovic, S., and Uitenbroek, H. (2019). STiC: A multiatom non-LTE PRD inversion code for full-Stokes solar observations. *A&A*, 623:A74.

De Pontieu, B., Title, A. M., Lemen, J. R., et al. (2014). The Interface Region Imaging Spectrograph (IRIS). *Solar Physics*, 289(7):2733–2779.

Dwivedi, Y. K., Kshetri, N., Hughes, L., et al. (2023). "So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management*, 71:102642.

Kriginsky, M., Oliver, R., and Kuridze, D. (2023). Temperature diagnostics of chromospheric fibrils. *Astronomy & Astrophysics*.

Leenaarts, J., Pereira, T. M. D., Carlsson, M., et al. (2013a). The formation of IRIS diagnostics. I. A quintessential model atom of Mg II and general formation properties of the Mg II h&k lines. *The Astrophysical Journal*, 772(2):89.

Leenaarts, J., Pereira, T. M. D., Carlsson, M., et al. (2013b). The formation of IRIS diagnostics. II. The formation of the Mg II h&k lines in the solar atmosphere. *The Astrophysical Journal*.

Lockheed Martin Solar and Astrophysics Laboratory (Last updated 2022). IRIS data notes. https://iris.lmsal.com/itn26/data_notes.html. Last checked on May 7, 2023.

Nóbrega-Siverio, D., Guglielmino, S., and Sainz Dalda, A. (2021). Solar surges related to UV bursts. Characterization through k-means, inversions and density diagnostics. *Astronomy & Astrophysics*.

Panos, B., Kleint, L., Huwyler, C., et al. (2018). Identifying typical Mg II flare spectra using Machine Learning. *The Astrophysical Journal*.

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229.

scikit-learn (2007-2023a). scikit-learn documentation on hierarchical methods. https://scikit-learn.org/stable/modules/clustering.html#different-linkage-type-ward-complete-average-and-single-linkage. Last checked on May 4, 2023.

scikit-learn (2007-2023b). scikit-learn documentation on k-means. https://scikit-learn.org/stable/modules/clustering.html#k-means. Last checked on April 1, 2023.

SciPy (2008-2023). Scipy documentation on hierarchical methods. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy-cluster-hierarchy-linkage. Last checked on April 1, 2023.

Sokal, R. and Rohlf, F. (1962). The comparison of dendrograms by objective methods. *Taxon*, 11:33–40.