Universitat Oberta de Catalunya

Master's degree in Data Science

MASTER THESIS

Area: Specialized fields

Density-based clustering for X-ray source detection on XMM-Newton EPIC-PN data

Application of DBSCAN, OPTICS and HDBSCAN

Author: Rodrigo Rico Gómez

Supervisors: José Vicente Perea Calderón[1] and Laura Ruiz Dern[2]

Advisors: Pedro Rodríguez Pascual[1] and María Santos Lleó[1]

Professor: Jordi Casas Roma[2]

---

[1]XMM-Newton Science Operations Center, European Space Astronomy Center (ESAC-ESA)

[2]Estudios de Informática, Multimedia y Telecomunicación, Universitat Oberta de Catalunya (UOC)

June 6, 2021

Contents

Copyright

FICHA DEL TRABAJO FINAL

Title:	Density-based clustering for X-ray source detection on XMM-Newton EPIC-PN data
Author name:	Rodrigo Rico Gómez
Supervisor:	Laura Ruiz Dern
Professor:	Jordi Casas Roma
Date (mm/yyyy):	06/2021
Title:	Master's degree in Data Science
Area:	2
Language:	English
Keywords	Unsupervised Learning, X-ray Astronomy, XMM-Newton

Acknowledgments

I would like to start acknowledging my parents and my brother, for being always there for me.
To Pere, for your patience over each and every weekly meeting discussing new ideas for the project.
To Laura, for your attention during all this work.
To María and Pedro, for all the knowledge and experience provided to this project.

Abstract

The main purpose of this work is to explore X-ray astronomical data from EPIC cameras on board XMM-Newton observatory by using machine learning techniques. As a result, and depending on the scientific validation, part of the techniques/algorithmts produced could be implemented in future XMM-Newton processing pipeline versions. XMM-Newton is short for X-ray Multi Mirror observatory [1], and it is one of the main science missions of the European Space Agency. As X-radiation is absorbed by the Earth's atmosphere, XMM-Newton should necessary be a space observatory, launched in 1999 by ESA's Ariane 5 rocket [1]. Data collected by XMM-Newton is sent to a complex set of algorithms and calibrations methods resulting in scientific data products elaborated and distributed by XMM-Newton Science Operations Centre [1]. This work is marked in the last stage of data processing. The data treated in the project has been previously processed so the focus is kept on Machine Learning techniques. Although, as this work will be developed in an astronomical environment and cosupervised by ESA scientists, there will be a strong astronomical point of view.

Keywords: Machine Learning, Clustering, DBSCAN, X-ray Astronomy, XMM-Newton.

Resumen

El principal propósito de este trabajo es explorar los datos astronómicos recogidos por las cámaras EPIC del satélite XMM-Newton. Como resultado, dependiendo de la validación correspondiente, parte de las técnicas empleadas podrán ser implementadas en futuras versiones de la pipeline de procesamiento de los datos de XMM-Newton. XMM-Newton es un observatorio espacial de la Agencia Espacial Europea (ESA) que opera en el rango de los rayos X. Debido a que la radiación a esa energía es absorbida por la atmósfera, es necesario que XMM-Newton se encuentre en órbita. Los datos recogidos por XMM-Newton son enviados a un complejo sistema de algoritmos y métodos de calibración que resultan en productos científicos elaborados y distribuidos por el Centro de Operaciones Científicas de XMM-Newton (XMM-Newton SOC). Este trabajo se enmarca en la última parte del procesamiento de los datos. Los datos tratados en este proyecto han sido previamente procesados para evitar desviar el foco de las propias técnicas de aprendizaje automático que se pretenden utilizar.

Palabras clave: Aprendizaje Automático, Clustering, DBSCAN, Astronomía de rayos X, XMM-Newton.

Chapter 1

Introduction

1.1 Justification of interest and relevance

The nature of data products obtained by XMM-Newton data processing pipeline makes them suitable for applying unsupervised Machine Learning techniques, particularly clustering methods. One of the best capabilities of XMM-Newton EPIC cameras is that they are able to detect X-ray photons individually [1], so the type of data stored are events lists, where each event is supposed to represent the arrival of an X-ray photon to the detector. Each event is characterized by its 2-dimensional position on the detector, which can be converted to sky coordinates applying the proper transformation, the energy of the photon and the time it arrived. In other words, an EPIC observation is a set of thousands of data points in a multidimensional space. As no prior labeling is provided, the analytical problem is fully unsupervised.
XMM-Newton Science Operations Centre (SOC) applies its own analytical methods to treat this multidimensional arrays data. The main interest of this work is the use of Machine Learning techniques to obtain the compatible results and discover new patterns unnoticed by the actual analytical engine. Some primary objectives when dealing with XMM-Newton observations are the detection of sources, the selection of good time intervals and the characterization of the background noise.

1.2 Personal motivation

As a BS in Physics holder and MS in Data Science student, the possibility to apply Data Science and Machine Learning to astronomy is the ideal mixture. Currently, I am enjoying an internship experience at ESA, concretely at XMM-Newton Science Operations Centre in ESAC, which means that I am working with the same scientists that elaborate and analyze the data products I am working with, which is a privilege and an opportunity to write my master's

thesis in this area.

1.3 Objectives definition

As a result of the Machine Learning methods applied, we aim to obtain:

- An alternative system to detect optimal observation time intervals. That implies retrieving time intervals with a special characterization of the background level, which could be used to filter out periods of time with high noise flares. Those episodes are caused by changes in the spacecraft external environment. These changes are mainly produced by the solar activity and the well-known belt of high energy charged particles around the Earth (the Van Allen belt), and of course the relative position of the satellite with respect to those effects.
- An alternative clustering-based method to detect astronomical X-ray sources from the multidimensional data of one single observation.
- Detection and characterization of the background.
- A measure of the significance of each source detected. By significance it should be understood a normalized scale, in which detected sources with a high chance of represent a real X-ray source will have higher values.
- Finally, a scientific validation of the results using those obtained by traditional XMM-Newton SOC methods.

As part of the validation process, all this objectives should be applicable and probed by a wide variety of XMM-Newton observations in order to cover different situations.

1.4 Methodology followed and project deployment

As we can see in Fig. 1.1, the project stages has been divided in three groups:

- Planning (in blue): these stages require exploring and reconsidering the objectives.
- Reading and writing (in green): stages requiring only investigation of related work and writing the thesis.
- Technical stages (in red): they require designing, implementing the methods and validating them with multiple XMM-Newton observations.

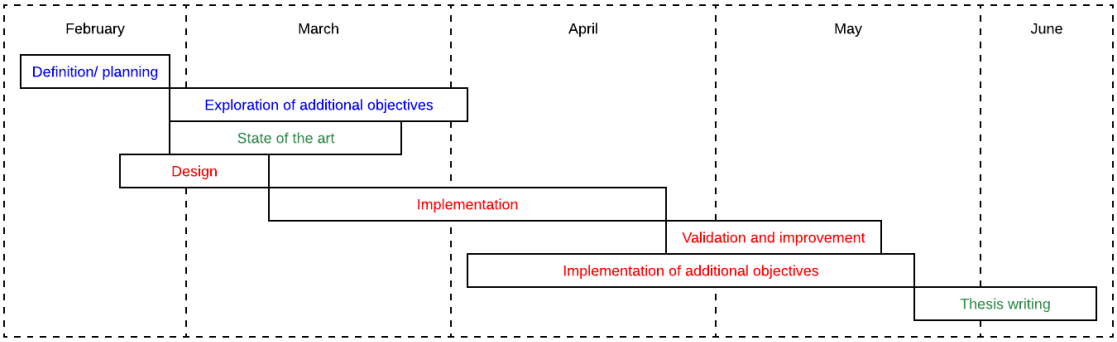

Figure 1.1: Stages of the project

Notice that during most of the time, there are multiple stages overlapping. This phenomenon occurs due to a few facts:

- Reading/writing stages and technical stages are independent. E.g. there is no need to finish the state of the art investigation to start designing and implementing the technical methods.
- The need of including two stages dedicated to possible additional objectives. As this work is cosupervised by ESA scientists in daily contact with XMM-Newton data products, new requirements and ideas will surely arise. There is a blue stage dedicated to explore new objectives and study their viability. Subsequently, there is another red stage for deploying this new ideas.
- Most of this activities are part of my current job at ESA, which means that I am dealing with technical daily tasks, regardless of the master's thesis stage I am involved in.

As for the tools used in each stage, the environment in which the project will be deployed is Jupyter Lab, working with both Python Notebooks and scripts.

Chapter 2

State of the art

This chapter is intended to give the reader a picture of the actual situation. It will be divided in two main parts: firstly, the current procedures used in XMM-Newton Science Operations Centre to analyze the incoming data from the satellite. And secondly, the actual marriage between astronomy and machine learning, making special emphasis on those clustering algorithms used in this work.

2.1 XMM-Newton observatory

First of all, a brief description of XMM-Newton space observatory (Fig. 2.1) will be given for the reader to be familiar with the instruments that make this work possible.

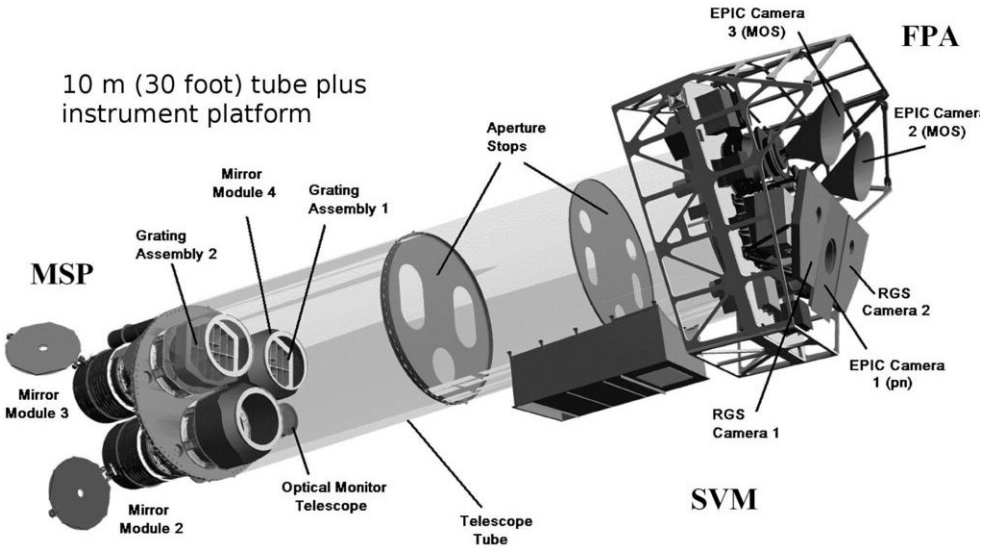

Figure 2.1: XMM-Newton X-ray space observatory. Obtained from [1]

XMM-Newton carries three main scientific instruments [2]:

- **European Photon Imaging Camera (EPIC)**: consists on three CCD-based cameras that perform moderate-resolution spectroscopy, X-ray imaging and X-ray photometry. Two of them are MOS-CCDs and the other is PN-CCD. The EPIC-PN camera is the one whose collected data will be used in this project.
- **Reflection Grating Spectrometer (RGS)**: two identical spectrometers for high resolution X-ray spectroscopy and X-ray photometry.
- **Optical Monitor (OM)** to have an optical/UV view of the observations performed by EPIC.

The specifications of the EPIC-PN camera should be taken under consideration when analyzing the data. The incoming data, as it will be explained in the following section, is a list of events, where each event is a photon, and the main attributes are the position, time and energy. When dealing with this quantities, it is necessary to keep in mind the spatial, temporal and energy resolution of the instrument. Spatial resolution is given by the Point Spread Function of the telescope and the size of the pixels in the camera, each of the 12 CCDs have 200x64 pixels [2]. Temporal resolution depends on the readout mode. The unit of resolution in terms of space are the pixels and in terms of time, the frames, both can be obtained directly from the FITS file header. The energy range observed is $0.2keV - 15keV$ and the resolution is $1eV$ [2].

2.2 XMM-Newton SOC data analysis

Current XMM-Newton data analysis tasks are performed using SAS (Scientific Analysis System) [1]. SAS is a collection of scripts, libraries and tasks specifically designed for scientific data analysis of XMM-Newton observations [1].

Fig. 2.2 shows the process followed by raw observational data (ODF - Observational Data File) until it is converted into high-quality calibrated science data products such as spectra, time series, images or source lists.

The input data for the techniques developed in this work is not the primitive ODF product, but the reduced event list in FITS format. The process from ODF to event list is performed by a set of complex algorithms and calibration methods [1] developed by XMM-Newton SOC staff and instrument developers. This first stage of the pipeline is not the objective of this work.

Once the event list is ready, depending on the type of analysis demanded, there is a SAS thread to be executed [1]. Some of the most important threads available in SAS will be briefly explained below, in order to give an idea of how XMM-Newton data is processed to generate

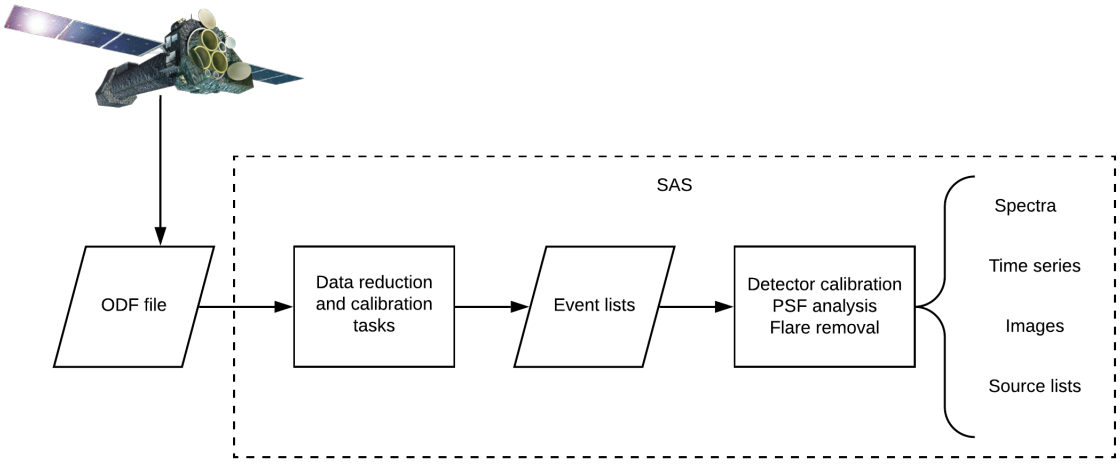

Figure 2.2: XMM-Newton data analysis scheme. Information obtained from [3].

scientific products.
A thread should be understood as a sequence of SAS tasks that yield to an specific objective. There are threads for generating light curves, spectra, removing flaring particle background or detecting sources.

2.2.1 Flaring background particle filter

The input of this SAS thread is the event list of the observation. This thread defines a threshold of counts per second [1]. Time intervals with higher counts per second rate are classified as flaring background intervals. The rest of the time, where the counts per second rate is below the threshold, are the so called GTI (Good Time Intervals) [1]. This is how SAS removes the flaring background particles. The purpose of filtering out periods of high background is to increase the signal to noise in the source products and to decrease the count rate lower limit in source detection.

2.2.2 EPIC source detection thread

When following this thread, the user will generate optimally background-filtered images in five energy bands and apply source detection techniques on them [1]. Also, the list of sources detected is provided.
Initial steps of this thread involve generating the calibrated and concatenated event list, and filter the flaring particles background. Both are mentioned previously in this section.
This SAS thread is composed by a set of complex algorithms that are open to be consulted on XMM-Newton SOC site [1].

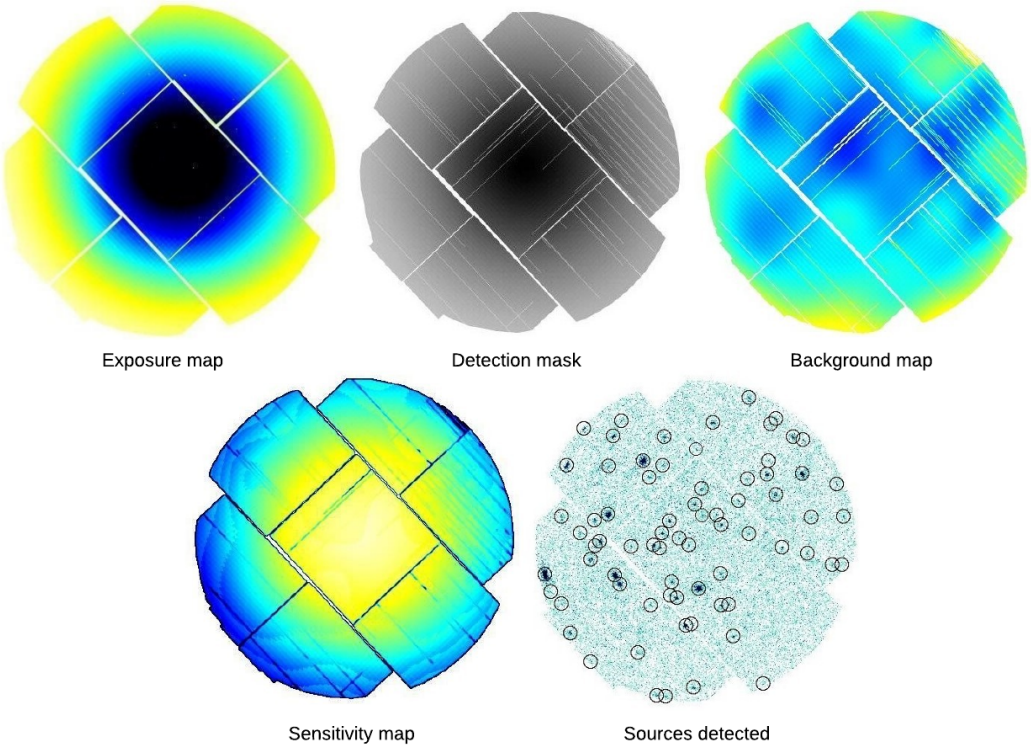

Figure 2.3: Different maps produced to achieve source detection with SAS. Individual images obtained from [1].

In summary, detection is performed following a chain of steps:

- Firt of all, the input of the process should be an EPIC event list with GTI defined by removing flaring background periods [1].
- The next step is the creation of an exposure map [1]. It gives the exposure time per instrument pixel, taking invalid pixels and relative detector efficiency into account [4]. They serve as input to the detection masks and background maps [4].
- After that, the detection masks are created from the exposure maps [1]. This masks represent the valid pixels per image [4].
- Once obtained the detection mask, a first round of detection is performed using the *eboxdetect* command from SAS [1]. This step is necessary to prior calculate the background map with non-source locations [1]. Point-like sources are detected using the PSF (Point Spread Function) adjustment (considering a box of 3x3 or 5x5 pixels for the source and 4 more pixels for the surrounding background [1]). Extended sources are obtained by searching in up to three consecutive detection rounds, increasing the number of pixels in the box [1].

- Then, the background map [1] is obtained, which gives the background counts [4] in each pixel of the image.
- A second detection round is performed using the background map calculated in the previous step [1]. Also, fitting all sources by maximum likelihood criteria, obtaining their centres and extension by fitting the local PSF [1].
- A sensitivity map, which indicates point source detection upper limits for each pixel [1]. The upper limits are derived by assuming Poissonian count statistics in each 3x3 pixel detection box [1].
- Finally, detected sources location are displayed over the EPIC original image [1].

2.3 Machine Learning clustering in astronomy

In the previous sections of this chapter, an introduction to XMM-Newton space observatory and actual data analysis techniques has been exposed. In contrast, this section will not be oriented to XMM-Newton, but to Machine Learning clustering methods and their latest applications in astronomy, regardless of the branch.
Before diving into possible applications of clustering algorithms for solving astronomical data analysis tasks, it is necessary to give the reader an introduction to clustering as an important branch of machine learning.

2.3.1 Unsupervised learning

Machine learning can be divided in two main parts, according to the nature of the problem they aim to solve:

- **Supervised learning**: when the goal is to get the value of one or more response variables ($Y = (Y_1, ..., Y_m)$) as a function of a set of predictor variables ($X^T = (X_1, ..., X_p)$) [5]. So the target of a supervised machine learning algorithm should be approaching to a function (F) that fulfills:

$$Y = F(X^T) \tag{2.1}$$

  To achieve this target, one should provide pairs of samples (Y_i, X_i^T) to the algorithm, so it can "learn" from them. That is why this process is called *supervised learning*. In order to quantify the learning direction, a loss function is defined to measure the difference between Y_i and $F(X_i^T)$, so the aim of the algorithm should be finding its minimum.

- **Unsupervised learning**: in this case, the only input for the algorithm is a set of attributes ($X^T = (X_1, ..., X_p)$) measured on n observations. But, as long as there is no response variable provided, there is no prediction to make, and the goal is to obtain interesting insights from the set of observations X_n^T [6].

The techniques developed in this work are unsupervised since the data conformed on XMM-Newton event lists is unlabeled. An event list is a set of attributes (time, position, energy, ...) measured on a large amount of points (events).
Unsupervised learning is a very broad term. To be concrete, the techniques used in this work are part of a subfamily of unsupervised learning called *clustering*.

2.3.2 The clustering problem

Clustering means finding subgroups (or clusters) of observations inside the total dataset [6]. Subsequently, the output of a clustering algorithm is another attribute of the data set that represents the cluster each observation belongs to.

$$X^T \longrightarrow \{C_1, ..., C_k\} \tag{2.2}$$

Up to now, the inputs and outputs of a clustering problem are defined. Now, it is necessary to give the algorithm rules on what is is considered a good or a bad output.
As told before, the aim of a supervised learning algorithm is basically optimize a loss function. The problem is that this loss function is defined over the prediction error, which makes no sense in unsupervised domains, where there is no variable to predict. That is why, it is necessary to find another way to quantify the goodness of a clustering algorithm output. It will depend on the specific algorithm used to perform clustering, but the main goal is to divide the data set into groups, such that observations belonging the same group are as similar as possible and observations belonging different groups are as different as possible [5].
To express this target mathematically, a formal expression for similarity between observations must be provided. Actually, dissimilarity is much easier to define than similarity, since the ground level of dissimilarity in the attributes space can be set to zero. Therefore, the dissimilarity of two observations (i and j) will be a function of the attribute values of both observations:

$$\text{Dissimilarity } = d(X_i^T, X_j^T) \tag{2.3}$$

The function d is commonly called the distance between the two observations in the attributes space. The most widely used distance measure in clustering applications when dealing with continuous attributes is the euclidean distance. For an m-dimensional space each observation

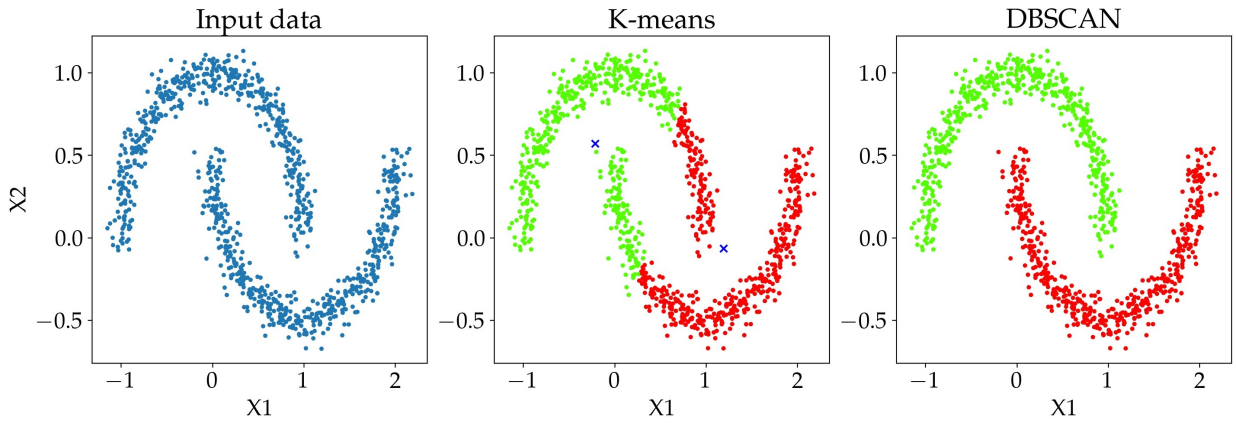

Figure 2.4: Difference between distance (K-means) and density-based (DBSCAN) clustering algorithms on data with arbitrary shape. Self-generated image with *Python* library *sklearn* sample data set *moons*.

i will be represented by m attributes ($i \leftarrow X_i^T = (x_{i1}, ..., x_{im})$), and the euclidean distance between two observations (i, j) will be defined as:

$$d(X_i^T, X_j^T) = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \tag{2.4}$$

Most of the models used in clustering rely on the concept of distance. Although, the way they use distance is something that varies. Some of them use the distance as the main criteria to form clusters, those are the distance-based clustering algorithms [7]. Distance-base algorithms use is widely spread for their simplicity [7] and good results on the majority of domains [6]. Unfortunately, the nature of the problems faced in this work makes distance-base clustering a poor approach.

When dealing with event lists, where interesting features in the observation are characterized by unusually high counts/(time × area) rates, and observations are full of background noise events, the best approach to solve the clustering problem are density-based algorithms.

2.3.3 Density-based clustering

The main point of density-based clustering is that makes no assumption on the distribution of the data treated. Some other clustering approaches, like *k-means* [8] or *EM* [9] suppose data is generated by some probability distribution (in this case the gaussian). As a result, the output of both algorithms are always spherical clusters [7]. In contrast, density-based algorithms adapt to any distribution followed by the data, and is able to output clusters with arbitrary shape [7]. One example is provided in Fig. 2.4, where the input data has two natural groups, but they

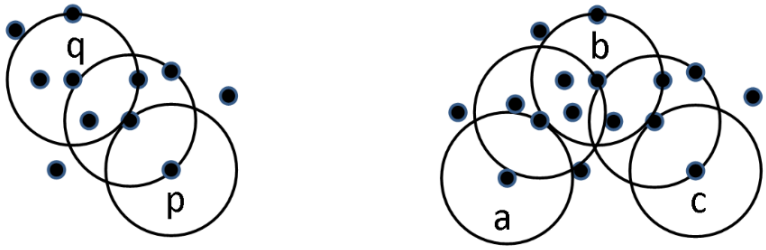

Figure 2.5: Left: density-reachability. Right: density-connectivity. Obtained from [7].

have arbitrary non-spherical shape. We can see how a distance-base algorithm like *k-means* is unable to give an appropriate output. On the other hand, a density-based algorithm like DBSCAN [10] (the most widely used) is able to adapt the output to the original data groups and reaches a much more coherent result.
In depth, the objective of density-based algorithms is to group densely connected areas of the attribute space, surrounded by sparser areas [7]. Although not being distance-based, they still use the previously defined concept of dissimilarity, as the distance in the attribute space (eq. 2.3), to evaluate whether a region is dense or not. And in most cases, the distance used would be the euclidean distance (eq. 2.4).

2.3.4 DBSCAN, the most common density-based algorithm

It is a clustering algorithm that works using observations density criteria. All the observations belonging to the same cluster are densely connected [7]. An observation point in the attribute space (p_i) is densely connected to other p_j if exists a chain of points: $p_i \rightarrow p_{i+1} \rightarrow p_{i+2} \rightarrow ... \rightarrow p_j$, where p_i and p_{i+1} are neighbors [7]. Another condition is that every point in the chain (except the two edges $\{p_i, p_j\}$) should be core points [7].
Two points are considered as neighbors when the distance between them is lower than a limit (ε). Core points are those having a minimun number of neighbors (*min_ samples*) or more [7]. In consequence, to execute the algorithm, two hyperparameters must be provided:

- ε: the maximun distance two points can have to be considered neighbors.
- *min_ samples*: number of neighbors required to a neighborhood in order to be considered a cluster.

A point is labeled as noise when it is not neighbor of a core point. Therefore, a cluster is a set of densely connected points with an amount of them larger than *min_ samples*. Formally, a cluster C can be defined as a subset of the total points D that fulfills [7]:

$$\forall\{p,q\} \text{ if } p \in C \text{ and } q \text{ is density reachable from } p \Rightarrow q \in C \tag{2.5}$$

This is what it is called density-reachability, see Fig. 2.5. Another condition related to every cluster is the following:

$$\forall\{p,q\} \in C : p \text{ is density-connected to } q \tag{2.6}$$

Then, the formal definition of noise points is this subset:

$$\text{noise} = \{p \in D | p \notin C_i \forall i\} \tag{2.7}$$

As a result of this conditions, DBSCAN reaches a final state where every point belongs to a cluster or is classified as noise, and inside each cluster, there are core points and border points. E.g. in Fig. 2.7, q and b are core points, and p, a and c are border points [7].

2.3.4.1 The algorithm

The algorithm that reaches this final state, given an input data set D is described by the following pseudocode:

Algorithm 1: DBSCAN

Result: $P_i \longrightarrow \{C_1, ..., C_k, N\}$
Input: D, ϵ, min_samples;
Initialization;
$C = 0$;
for *each unvisited point* P_i *in data set* D **do**
  mark P_i as visited;
  $NeighborsPts_i$ = regionQuery(P_i, ϵ);
  if $length(NeighborPts_i) < min\_samples$ **then**
    mark P_i as noise ($P_i \in N$);
  else
    C = next cluster;
    expandCluster(P_i, $NeighborPts_i$, C, ϵ, min_samples)
  end
end

As it can be seen, the algorithm takes one arbitrary unvisited point and evaluate its neighborhood until the distance limit ϵ. If it has the required number of points $min\_samples$ it starts

expanding the clusters using another algorithm: *expandCluster*. Otherwise, the point is labeled as noise.

Algorithm 2: expandCluster

Input: P_i, $NeighborPts_i$, C, ϵ, min_samples;
Initialization;
add P_i to cluster C ($P_i \in C$);
for *each point* P_j *in* $NeighborPts_i$ **do**
  if P_j *is not visited* **then**
    mark P_j as visited;
    $NeighborPts_j$ = regionQuery(P_j, ϵ);
    if *length*($NeighborPts_j$) $\leq$ *min_samples* **then**
      $NeighborPts_i$ = $NeighborPts_i$ joined with $NeighborPts_j$
    end
    if P_j *is not member of any cluster* **then**
      add P_j to cluster C ($P_j \in C$)
    end
  end
end

The temporal complexity of the algorithm is governed by the number of times *regionQuery* is called. Temporal complexity is naturally $o(n^2)$ but can be reduced to $o(n\log(n))$ when using indexing structure to query neighbors [10]. Memory complexity will be $o(n^2)$ for distance matrix-based implementation but can be $o(n)$ if not [10]. The distance matrix D_{ij} contains the distance between every pair of points P_i and P_j and it is used to avoid recalculating distance during the clustering process.

Algorithm 3: regionQuery

Input: P_i, ϵ;
Output: $NeighborPts_i$;
Initialization;
Add to $NeighborPts_i$ every point P_j with a distance to P_i lower than ϵ.

2.3.4.2 Hyperparameter selection

Hyperparameter selection in DBSCAN is not trivial. As it is an algorithm really sensitive to ϵ and *min_samples*, it is necessary to find a method that, at least, would give a hint. Fortunately, in [10], the creators of DBSCAN showed a way of estimating the values of ϵ and *min_samples*.

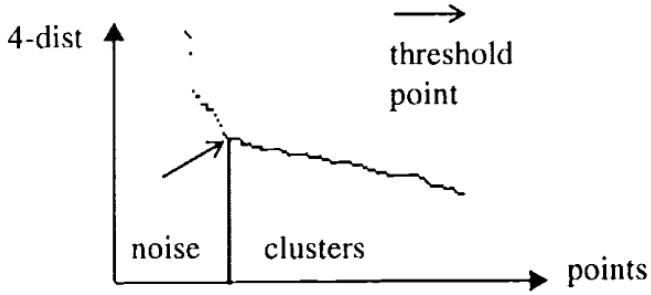

Figure 2.6: E.g. points sorted by dist(k, p) with $k = 4$. Obtained from [10].

Let d be the distance between a point p and its k^{th} nearest neighbor. If $\epsilon = d$ the number of members on its neighborhood will be k. Now, let dist(k, p) be a function that maps every point p in D and returns the distance to its k^{th} nearest neighbor:

$$\text{dist} : D \longrightarrow \Re \tag{2.8}$$

If now, points on D are ordered by decreasing value of dist(k, p), the result is a graph with the following shape (see Fig. 2.6). Taking an specific point p_i as reference, if ϵ is set to dist(k, p_i), and $min\_samples$ to k, the point p_i will represent a threshold. Points on the left will be noise points, since its k^{th} nearest neighbor's distance is larger than p_i's k^{th} nearest neighbor's distance, which is the ϵ. And by the same reasoning, points on the right will belong to a cluster.When representing this plot, the distribution has an "elbow". As stated in [10], the optimal value for ϵ is the dist(k, p) of this elbow, when $min\_samples = k$.
Notice that, this method aims to reproduce clusters with optimal shape, and gives an idea of the distribution of the input data. But depending on the specific domain, the user should consider its own hyperparameter selection. In the case of XMM-Newton EPIC source detection, varying the hyperparameters will detect brighter or fainter sources.

2.3.5 Other density-based algorithms: OPTICS and HDBSCAN

Apart from using DBSCAN as a density-based algorithm to perform clustering, another algorithms will be tested in this work to compare results.
Both OPTICS [16] and HDBSCAN [18] are based on the mechanism and the concept of density established by DBSCAN, so the main idea is common, but they introduce new advantages that will be presented below.

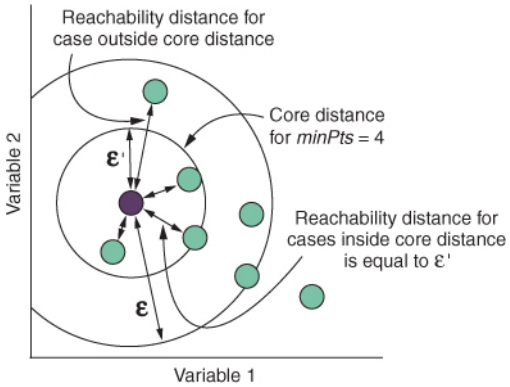

Figure 2.7: Representation of core distance and reachability distance concepts used by OPTICS clustering algorithm with $min\_samples = 4$. In this graph the core distance is ϵ' and ϵ is the hyperparameter. Obtained from [17].

2.3.5.1 OPTICS

OPTICS [16] is a viable alternative to DBSCAN for those cases in which local densities vary as a function of the position [7]. This is the case of astronomical observations in which the background is not homogeneous in space. In general, XMM-Newton observations background map is not constant (e.g. in Fig. 2.3). That makes OPTICS an algorithm that must be tried in this domain. To sum up, the main advantage of OPTICS over DBSCAN is that it is able to detect clusters of different densities. The behavior behind OPTICS is really similar to DBSCAN as both are density-based algorithms that perform clustering. One remarkable difference between both is that the output of OPTICS is a certain order of the points that can yield to cluster structures [17]. Then, the clustering can be performed by using this ordering on another clustering method, like DBSCAN.

OPTICS uses the same hyperparameters of ϵ and $min\_samples$ previously introduced by DBSCAN. However, their function is not exactly the same. In DBSCAN, the radius of the region in which a point's neighbors are searched (search radius) is constant. The value is ϵ (a DBSCAN hyperparameter) regardless how dense is the surrounding of it. The key difference introduced by OPTICS is that a point's search radius depends on the points surrounding it. Points in dense regions will have a lower search radius associated, whereas points in sparser regions will have their search radius enlarged. This way, in principle, the algorithm will be able to adapt the process of cluster formation to the density of different regions in the EPIC observation. In other words, in denser regions the requisites for a group of points to form a cluster will be harder to fulfill.

The search radius is established as the distance from a point to its k-th nearest neighbor, being k the $min\_samples$ OPTICS hyperparameter, the only one mandatory [17]. This radius is called the core-distance of a point (eq. 2.9). The other hyperparameter used is ϵ, defined to

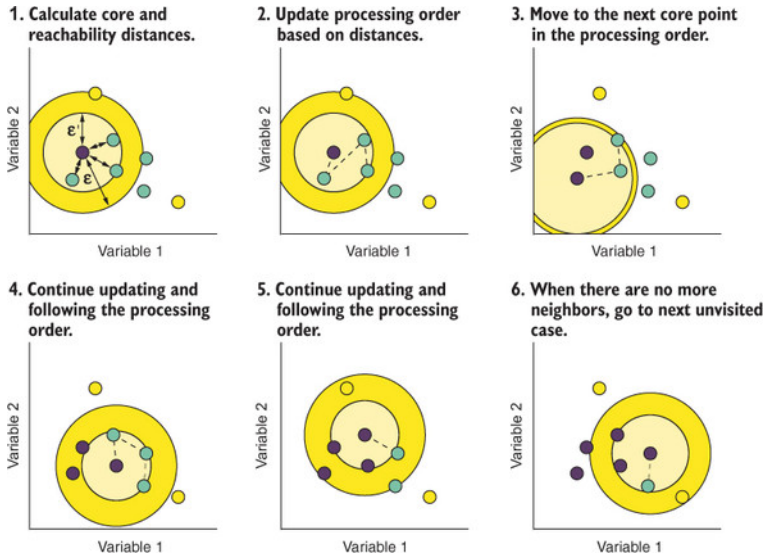

Figure 2.8: Steps followed by OPTICS clustering algorithm with $min\_samples = 4$. In this graph the core distance is ϵ' and ϵ is the hyperparameter. Obtained from [17].

reduce the workload of the algorithm, so that points with core-distance higher than ϵ are not considered [17]. As it is not mandatory, if undefined its value is set to infinity, which means that no upper limit applies.

Mathematically, core distance is defined over each point P in the data set D using the hyper-parameters ϵ and min_samples:

$$\text{core-dist} = dist(P\text{'s } min\_samples\text{-th nearest neighbor}, P) \text{ if } N(\epsilon, P) > min\_samples \quad (2.9)$$

Where $dist(P_i, P_j)$ is normally the euclidean distance (eq. 2.4), despite sometimes another metrics can be used (e.g. Minkowski). Notice that *core-dist* is defined only if $N(\epsilon, P) > min\_samples$ being N the number of points belonging the circle of center P and radius ϵ. In other words, if the neighborhood of P has less than $min\_samples$ points, the core distance of P remains undefined [16]. Which is the condition of a point to have a core-distance lower than ϵ expressed in a different manner.

The most important concept to order the points is the reachability distance. The reachability distance from a point P to a point O inside its core distance radius region is equal to the core distance. If O is outside the core distance, the reachability distance is equal to the euclidean (or another metric) distance between P and O. Mathematically, it can be said that the reachability distance is the maximum between the core and euclidean distance (eq. 2.10).

$$\text{reach-dist} = max(dist(P, O), \text{core-dist}(P)) \text{ if } N(\epsilon, P) > min\_samples \quad (2.10)$$

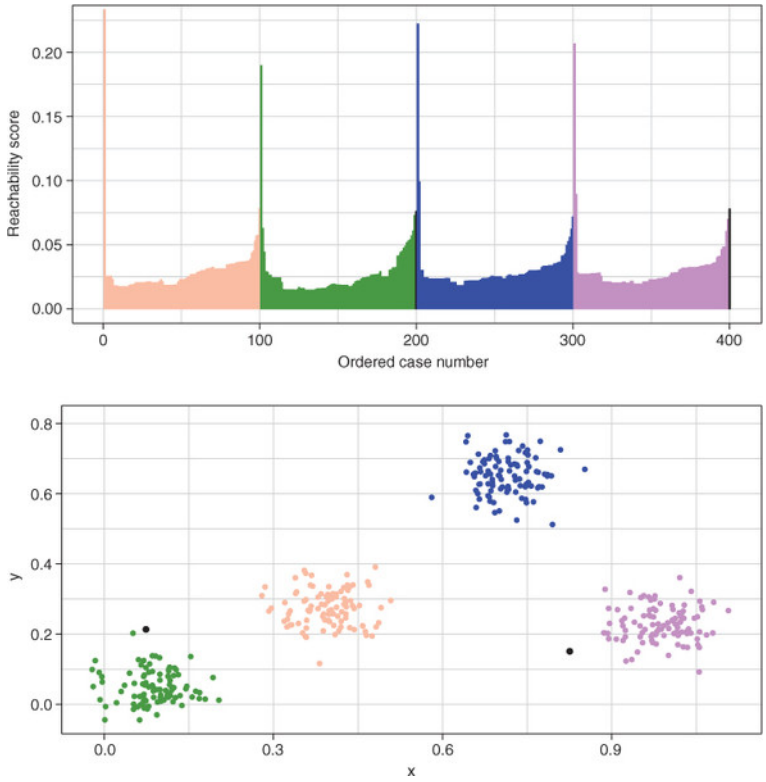

Figure 2.9: Output given by OPTICS. Reachability plot. Obtained from [17].

It is the distance between P and O or the *core-dist* of P, the highest value is chosen. If P is not a core point, mathematically: $N(\epsilon, P) > min\_samples$, reachability distance remains undefined.

Both core (eq. 2.9) and reachability distance (eq. 2.10) definitions are used by OPTICS algorithm in its execution. A graphical representation of both concepts can be seen in Fig. 2.7. Once both core and reachability distance have been defined, it is time to see how OPTICS use them to order the points. The first thing it does is label every point as core-point or not. Then the algorithm executes a sequence of steps illustrated in Fig. 2.8.

Before visiting the next point, OPTICS calculates the reachability score and updates the processing ordering of points [17]. If a point is not core point, its reachability score will be the reachability distance to the nearest core point [16]. And if it is a core point, the reachability score will be its core distance [16]. The processing ordering of points is updated and the algorithm visits the next point in the queue [17]. Once, all points have been visited, there is a certain ordering of points that is finally the output of OPTICS. As the processing sequence joins reachable points in the queue, ordered by reachability score, a cluster structure arises (see Fig. 2.9).

2.3.5.2 HDBSCAN

The other alternative, HDBSCAN [18], is the hierarchical version of DBSCAN. Its strength is that it is able to "decide", under new hyperparameters (*min_cluster_size* and *cluster_slection_epsilon* and *min_samples*) [19] whether two formed clusters should be joined as one or not. In the astronomical domain, it can be useful, e.g. when extended sources create regions of multiple clusters and it is interesting to resolve them individually.
HDBSCAN needs an efficient estimate of local density to perform clustering [19]. The concept of core-distance, previously defined in OPTICS section (see eq. 2.9), is also used by HDBSCAN under the same mathematical expression. Unlike DBSCAN, HDBSCAN transforms the space changing the euclidean distance by the mutual reachability distance (see eq. 2.11).

$$\text{mreach-dist}(a,b) = max(\text{core-dist}(a), \text{core-dist}(b), d(a,b)) \tag{2.11}$$

Where $d(a,b)$ is the euclidean distance between a and b. Notice that the core-distance is defined using the hyperparameter *min_samples* (eq. 2.9). Once this new distance metric is defined, the algorithm proceeds to find dense regions in order to form clusters of points. The approach to solve this problem is to see the data as a weighted graph where points are vertices whose edges are weighted by the mutual reachability distance [19].
A threshold value is defined so that every edge with a higher weight will be dropped from the graph [18]. This threshold starts high and becomes lower with "time" so at the beginning all connections between points are "switched on". As the threshold starts decreasing, points with high mutual reachability distance will start to lose connection. Here is where the concept of hierarchy arises, since the algorithm produces a hierarchy of connected components (from fully connected to fully disconnected) [19].
If the observation has n points, the number of edges to compute will be n^2 [18], so in practice this method will not be efficient. As a result, HDBSCAN uses a subset of the total set of edges that fulfills two conditions [18]:

- Dropping any edge of the subset would cause a disconnection between points. In other words, this subset will keep essential edges and drop redundant.
- There is no lower weight edge that could connect two points.

The desired subset of edges is analogous to the ideal road map to connect cities. E.g. in Spain, building a road from Seville to Barcelona that does not visit any other city would be inefficient. It would be better to establish the connection from Seville to Barcelona using shorter roads that connect a bigger amount of cities in the way, so they can also make use of the same road. This subset of edges is known in graph theory as the minimum spanning tree of the graph [19],

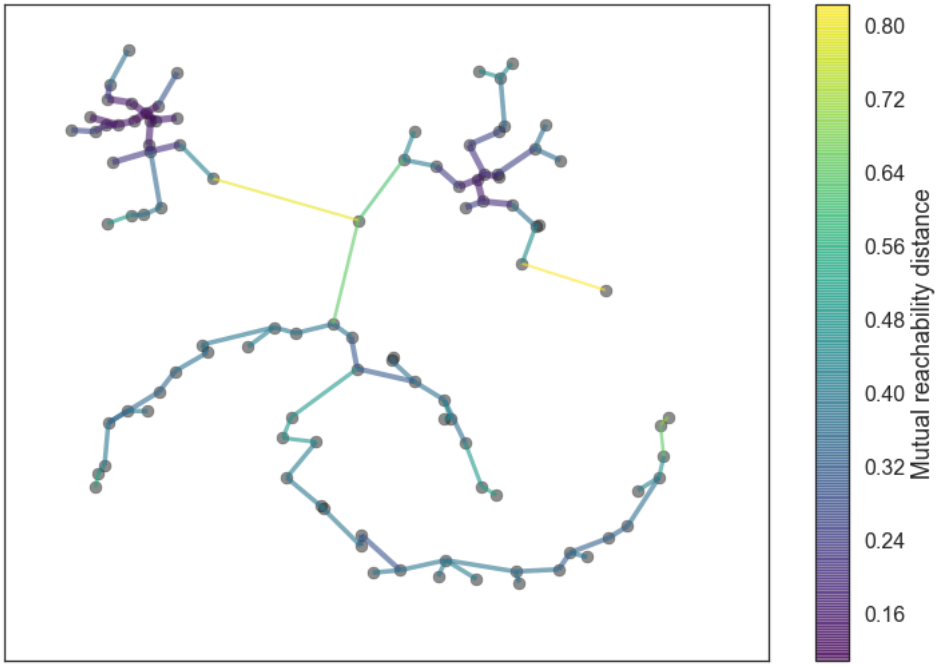

Figure 2.10: Example of the minimum spanning tree of a data set. Obtained from [19].

and there are several methods to compute it. To visualize the idea of the concept, an example is provided in Fig. 2.10.

Once this graph is obtained, using the weights given by the mutual reachability distance, a cluster hierarchy is built. The process is the same stated below using an upper threshold that decreases, but this time over the minimum spanning tree instead of the total graph.

A cluster structure can be obtained from this hierarchy by cutting it at a certain level. The first step of cluster extraction is condensing this hierarchy into a simpler tree (see Fig. 2.11). The majority of the splits in the resulting hierarchy are a cluster loosing one or two points rather than the division of a cluster in two [19]. It would be interesting to identify this splits to neglect them, since they do not contribute to understand the underlying cluster structure. To establish a criteria, a new hyperparameter (min_cluster_size) is introduced [18]. For each split, if one of the two outcoming clusters has less than min_cluster_size points, this split is declared as a cluster losing points [19]. This is how the condensed tree is built (Fig. 2.11).

Once the condensed tree is built, the cluster structure, which is finally, the output of HDBSCAN is got selecting those clusters with longest lifetime [19]. The lifetime o a cluster is measured with the λ value from the condensed tree using the y coordinate (see Fig. 2.11). In other words, for each point, the cluster in which it spends the majority of its ”life” is the cluster it belongs to. Noise points are those neglected using the min_cluster_size.

As there is now a cluster label for each point, HDBSCAN execution has now ended.

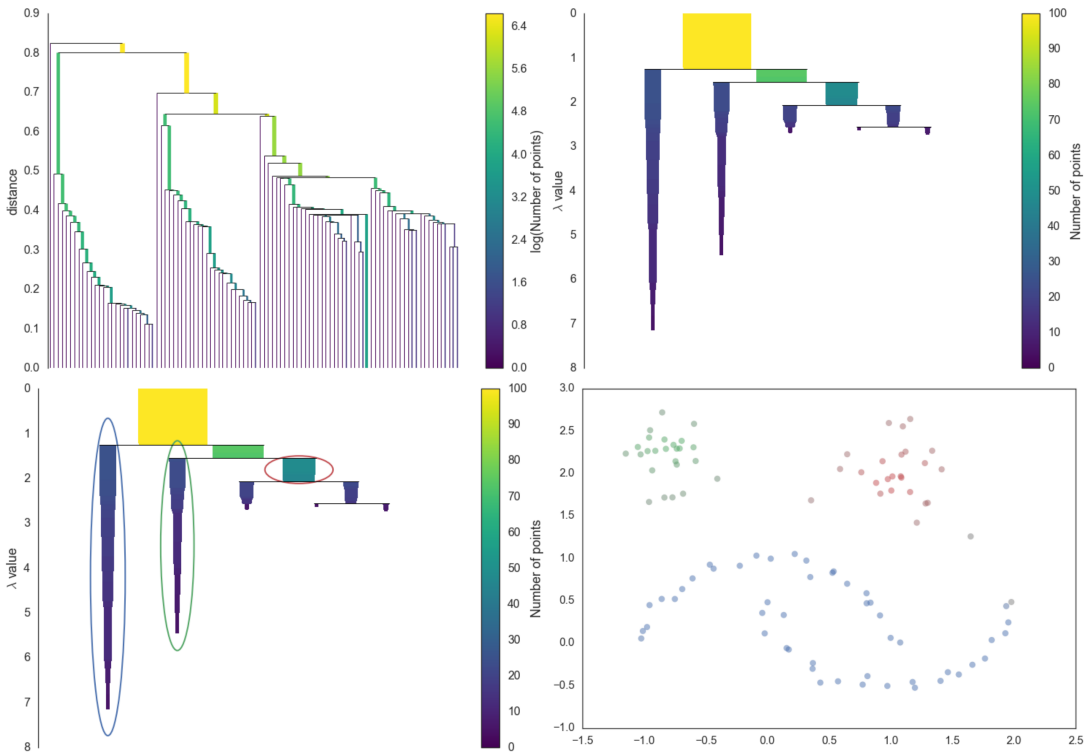

Figure 2.11: Up-Left: total hierarchy. Up-Right: condensed tree. Bottom-Left: selection of clusters. Bottom-Right: final output Obtained from [19].

2.3.6 Previous astronomical applications of machine learning

In this section, a brief description of previous works developed in the field of astronomy using Machine Learning, and a few specific examples of density-based algorithms will be given.

Astronomy, and specially X-ray astronomy, is a branch of natural sciences where data is gold, since sophisticated instruments (like XMM-Newton) are needed to obtain it. This fact, and the large amount of data actually stored, makes astronomy one of the main fields of application of machine learning.

At ESAC-ESA, a group dedicated to machine learning in astronomy has been recently formed. In one of their most cited publications they proposed a new method for denoising astronomical images taken by Hubble Space Telescope using U-nets [11], a fully convolutional neural network architecture with U-shape [11]. There are also two publications in which they use density-based clustering algorithms. In [12] they used DBSCAN to identify new member candidates in the star cluster located in Upper Scorpius subgroup. They also discuss the use of density-based DBSCAN over distance-based *k-means* claiming that it is able to find clusters of arbitrary shape [12]. Another publication by ESAC machine learning group related to density-based clustering is [13], where they concluded that density-based clustering (in particular DBSCAN, OPTICS

and HDBSCAN) is a promising method to identify candidate members of star forming regions. Out of ESAC, we will put the focus on the algorithms used in this work exclusively: DBSCAN, OPTICS and HDBSCAN. The most relevant papers using this techniques will be mentioned so the reader can have an initial idea on how clustering algorithms are applied in astronomical domains.

In 2018, a group of astronomers from the University of Barcelona used DBSCAN to explore data from Gaia [21]. Concretely, the function of DBSCAN was to find clusters of sources (mostly stars) with similar astrophysical properties, called Open Clusters (OCs). The results obtained led to the proposal of new OCs. Although it may seem a similar work than the one developed here, there is an important difference: in [21], the points in the data set are the proper sources and the resulting clusters are groups of them, while here, points correspond to events (X-ray photons), and sources are still supposed undetected. This works aims to solve a previous problem than [21], since the only information taken as an input is the X-ray radiation data. Another obvious difference is that Gaia performs measurements in the optical band, not X-rays.

DBSCAN has also been used to classify binaries' light curves in Gaia [22]. Again, sources have already been detected before applying DBSCAN, which makes a great difference with respect to this work.

A group of researchers from Changzou University in China, have used DBSCAN for detecting nearby open clusters from Gaia DR2 [14], where they stated that DBSCAN is very useful when finding densely populated structures in space data. On the other hand, they regret that it could not be able to calculate cluster membership probabilities [14].

2.3.6.1 Clustering algorithms in astronomical source detection

The use of DBSCAN for detecting astronomical sources has been proved for the first time over *Fermi*-LAT data [15], and concluded that DBSCAN was successful in detecting sources. It stated that the cluster centroids obtained with DBSCAN are located in similar positions than those obtained using the current *Fermi*-LAT software [15].

There is another application of DBSCAN in *Fermi*-LAT for detecting high energy ($E_\gamma > 100$GeV) γ-ray sources [23], a work similar to [15]. One of the most relevant problems detected when applying DBSCAN in [23] is that it is unable to deal with non-homogeneous spatial distribution of the background, issue that could be tentatively fixed by using alternative clustering algorithms (OPTICS and HDBSCAN).

In [15] and [23], the points in the cluster structure are γ-ray photons characterized by the position in sky coordinates. DBSCAN is applied to this lists of photons (the event list of the observation) in the same way than this work.

Another inspiring ideas from [15] and [23] to this work is the way they calculate the significance of each detected cluster. They first calculate the effective radius of each cluster using eq. 2.12.

$$r_{eff} = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{2.12}$$

Where x and y are the principal components of the cluster's points distribution, and σ is the standard deviation [23]. When having this r_{eff}, the significance is obtained from 2.13.

$$s = \sqrt{2\left(N_s ln\left[\frac{2N_s}{N_s + N_b}\right] + N_b ln\left[\frac{2N_b}{N_s + N_b}\right]\right)} \tag{2.13}$$

Where N_s are the number of points belonging to a cluster and N_b are the number of points in a region a distance between $2r_{eff}$ and $3r_{eff}$ from the centroid of the cluster [23]. In this work, a different calculation of the significance of each cluster is proposed, although the concept behind is similar.

Chapter 3

Design and implementation

This chapter contains a step by step explanation of how the final solution has been reached in order to fulfill the objectives stated in the first chapter.

3.1 Data acquisition and machine learning pipeline

The data needed in this project is available at XMM-Newton Science Archive, a web repository supported by XMM-Newton Science Operations Centre of ESAC. It holds all the files containing the event lists and the region lists to validate the results of clustering.

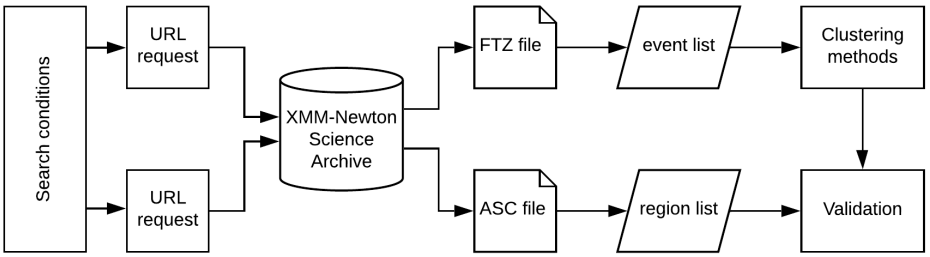

Figure 3.1: Data pipeline designed for the project.

The pipeline of the project is shown in Fig. 3.1. We download the required files from XMM-Newton Science Archive using a Python script that builds an URL request from the search conditions. For simplicity, as the only files with interest on this work is the EPIC-PN complete event list and the corresponding sources list (technically called region list), in the whole energy range, the only search filter we use is the observation ID. However, the archive accepts a wide range of customizations on the URL.
Once both files are downloaded, another Python script is executed to retrieve the events and

region lists and save them in local variables so they can be used in the pipeline.
As we can see in Fig.3.1, the event list inputs in the clustering methods, where the output will be a list of detected sources. We characterize detected sources with a few attributes (position, counts, significance) and labeling columns that corresponds to the cluster each event was assigned in each of the clustering threads built. The region list contains the list of sources detected by XMM-Newton pipeline, we use them in the validation stage to perform cross-match validation with our clustering detected sources.
The event list comes from the archive in FITS format. A FITS file is composed by a set of extensions that contains all the relevant information an astronomer would need when handling the data. Every extension has a header with information about the data it stores. The event list is located in an extension called *EVENTS*. It is a table with multiple columns that we are filtering to keep the ones we show on Table 3.1.

TIME (s)	The time an event was detected
X (0.05 arcsec)	Relative position in right ascension axis
Y (0.05 arcsec)	Relative position in declination axis
DETX (0.05 arcsec)	Position in the X axis of the detector
DETY (0.05 arcsec)	Position in the Y axis of the detector
PI (eV)	Energy of the photon associated with the event
CCDNR	CCD chip in which the event was detected

Table 3.1: Variables obtained from the EPIC-PN event list.

One of the most striking aspects of theses variables is that there are two different set of axis to determine the position: one set referred to the standard equatorial system to locate an astronomical source in the sky (X, Y), and another set referred to the detector coordinates system (DETX, DETY). An example of an observation represented in both sets is available in Fig. 3.2. Changing from X, Y to detector coordinates implies a non trivial transformation.
The source list is, as its name states, a list of elements, each of them containing the position of the source in sky coordinates and the radius of the region where the source is located. Notice that the position is given in sky coordinates (right ascension, declination). To compare results, we use the World Coordinate System Information stored in the event list FITS file metadata. Up to now, we have presented three different sets of axis: detector, X,Y and sky coordinates. It is convenient to apply the clustering methods using detector coordinates since it is a coordinate system where all observations are equivalent. However, we need to give the result in real sky coordinates in order to perform a cross match validation with XMM-Newton pipeline.

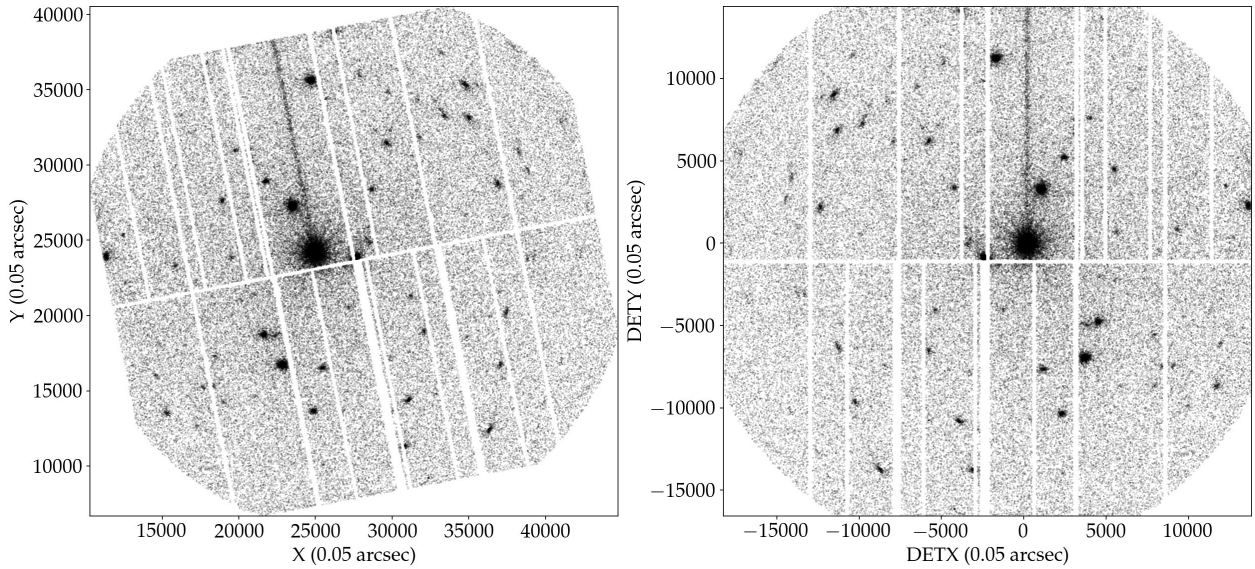

Figure 3.2: Observation seen from X, Y coordinates (*left*) and detector coordinates (*right*). EPIC-PN event list from observation ID: 0690200301 as example.

3.1.1 The clustering threads

In the pipeline, there are three clustering algorithms used and also three clustering threads. Each thread leads to a list of sources as an output. When sources are detected, we will validate them using two techniques: defining two significance metrics to evaluate the confidence we can give to each detection individually. And then, using XMM-Newton pipeline source list to perform cross match validation, as explained before.

In Fig. 3.1, this source detection threads are located in the cell called "Clustering methods", which is the core of this work.

In Fig. 3.3, an scheme for the reader to visualize the order of the clustering methods applied to the event list is given. As it can be seen, before the source detection is performed, flaring background intervals should be removed. Then the cleaned event list is sent to each source detection thread. Notice that two of them combine more than one clustering algorithm. In following sections each of them will be properly exposed. It is also appropriate to warn that, since all the facts related to the clustering algorithms have been widely explained in section 2.3, we will give no comments about the clustering theory in this chapter.

Before diving into the specific clustering threads of the pipeline, we would like to make considerations on the variables involved in the clustering algorithms. The flaring background removal thread will exclusively use temporal coordinate, whereas those threads designed for source detection involve only detector coordinates (DETX, DETY). One may wonder why the energy of the events is not involved. The reason is that astronomical X-ray sources are not characterized

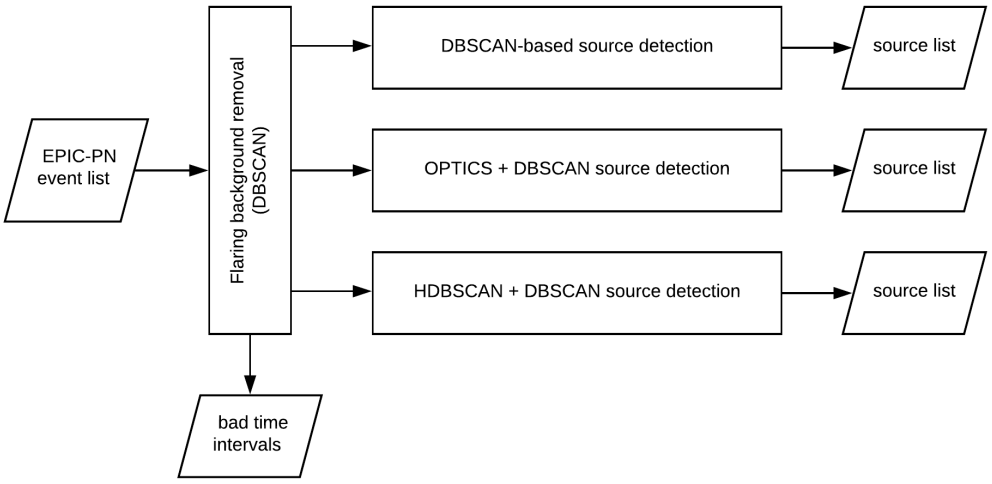

Figure 3.3: Clustering methods inside the pipeline.

by certain energy emission lines but by a continuous spectra. Not only that, but also that spectra of different sources normally overlap, lots of them have also similar shape. Having said that, it is logical to argue that, the energy of an individual photon tell us little about the source it belongs to. This is the reason why it is not used in this work to detect sources.

3.2 Flaring background removal with DBSCAN

As enunciated above, the first clustering technique we apply to the full event list has the objective of removing time intervals contaminated by high flaring background. Fortunately, these intervals are identified by an outstanding increment of the count rate (counts/second). From certain point of view, the count rate can be seen as density in the temporal axis of the observation. Therefore, the problem of detecting flaring background can be understood as the aim of finding regions densely populated by events considering only the temporal coordinate.
Having said that, we have redefined the problem so now density-based clustering is applicable, since it will lead to identify flaring background intervals as clusters of events. The key point is that DBSCAN algorithm is able to classify points as noise. Following the reasoning stated, noise points will be points not belonging to any flaring background interval. In other words, DBSCAN can be used over the total event list involving only the time coordinate in the clustering process and the cleaned events will be those classified as noise.

In Fig. 3.4 an example of this process can be seen graphically. We have selected the hyperparameters of DBSCAN considering the ”elbow” criteria exposed in chapter 2, but it must be said that we have tuned them manually to fit to XMM-Newton observations. Their final values are

eps (s)	min_samples (counts)
7	300

Table 3.2: Hyperparameters selected for DBSCAN in flaring background removal stage.

presented in Table 3.2. As the only coordinate considered in the clustering process is the time, the distance between points has also dimensions of time, subsequently, *eps* is given in units of time (seconds in this case). On the other hand, *min_samples* has dimensions of number of counts. if we rephrase this set of hyperparameters to astrophysical language, it can be said that we consider any time interval as flare when there are approximately a concentration of more than 300 events in a time interval of 7 seconds. This is the way the hyperparameters should be interpreted.

As this stage can be considered part of the data cleaning process, an acceptable output for the algorithm will be such that allows the source detection threads to obtain good results. In this case, this method will produce good results as long as the cleaned observation is digestible by the oncoming algorithms and sources are properly detected detected.

There is also some part of visual checking. E.g. in Fig. 3.4 we can see how DBSCAN is able to detect all intervals corresponding to flares in the observation and the output image is properly cleaned so the sources present in the observation are much more visible. After probing the method with a sufficient amount of observations, a coherent result has been observed in most of them. Consequently, this method is considered reliable.

An important fact to mention is that, in case that there are no flaring background intervals, all the points will be labeled as noise by DBSCAN algorithm and no event will be removed from the original list.

Nevertheless, we must warn some issues that could arise when using this method on certain observations. If the observation is overcrowded by sources until the point that they all together reach a count rate of 300/7, this method will consider this overload as flare, and all events will probably be removed from the event list. However, it is true that this count rate levels are really uncommon.

There is also another possible problem, but in this case is not fatal. We have been aware that in some observations there are small time intervals affected by fainter flares undetected by our method. E.g. in Fig. 3.4, at $t \sim 17000\ s$ we can see two faint flares that remain undetected after applying DBSCAN algorithm. All in all, this phenomenon do not imply major problems since the only outcome they produce is a slightly higher environmental background.

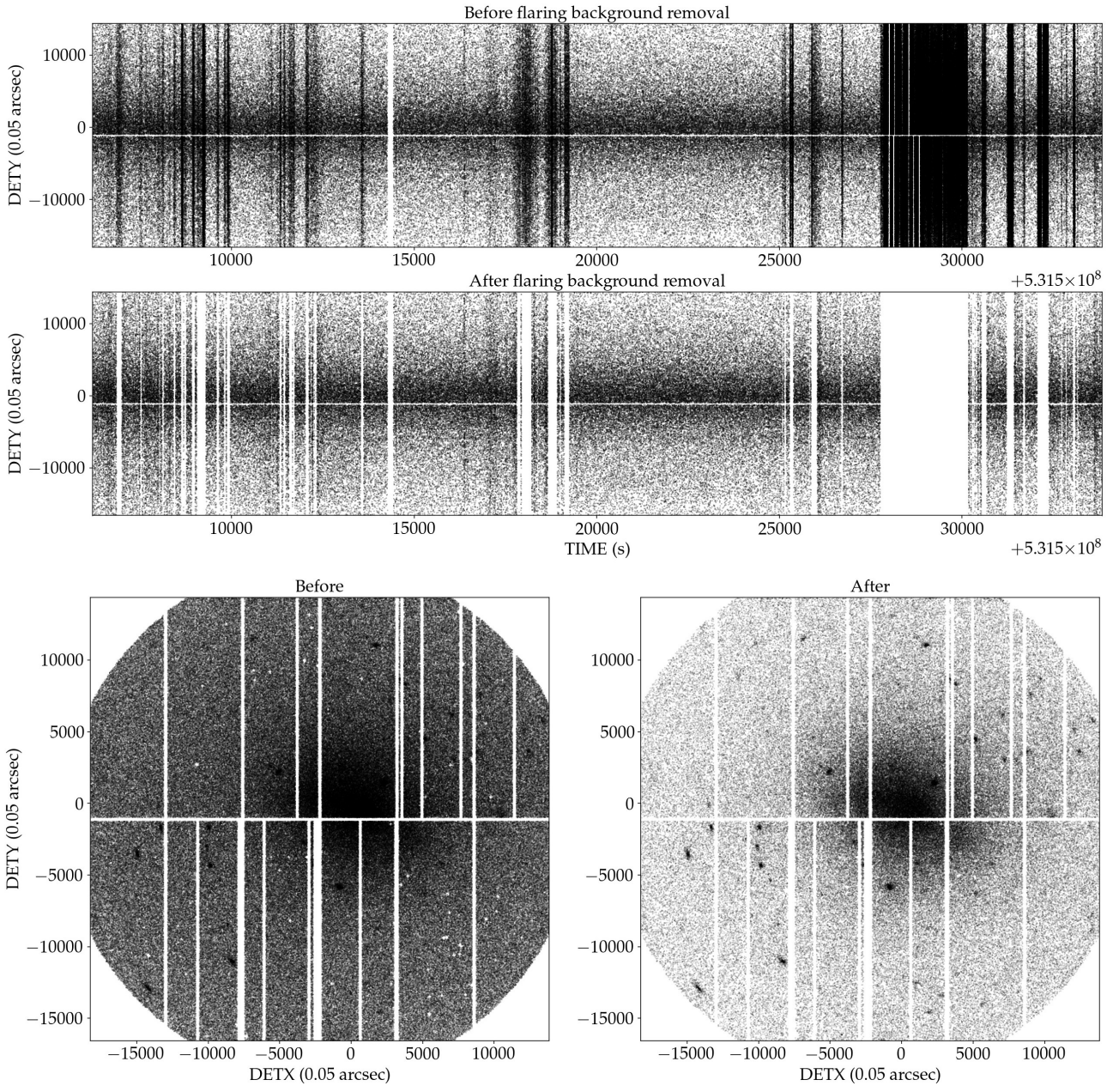

Figure 3.4: Comparison between an observation before and after removing the flaring background particles with DBSCAN. *(Top)*: observation collapsed on TIME-DETY plane. *(Bottom)*: observation collapsed on DETX-DETY plane (classical image). Self-generated figure with the EPIC-PN event list from observation ID: 0744414101 as example.

Once flaring background intervals are detected and removed from the event list, we proceed to apply source detection threads. The reason to design three different threads is not that each of them will detect a different type of source, but that each algorithm by its own has significant drawbacks, so it is necessary to try different algorithms and combinations. The threads are presented in chronological implementation order, so we can argue the creation of each of them basing on the issues generated by the previous ones.

3.3 DBSCAN-based source detection

As shown in Fig. 3.5, the only algorithm used in this case in this case DBSCAN.

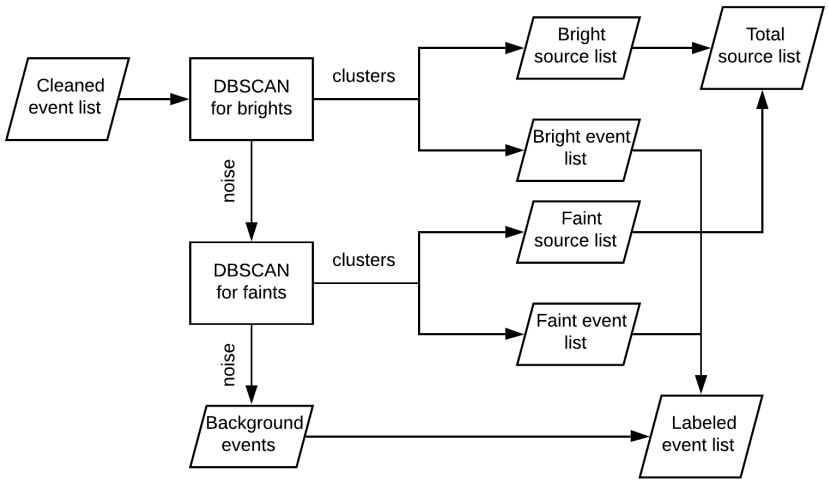

Figure 3.5: Data flow scheme for DBSCAN-based source detection thread.

The process is a combination of two rounds of DBSCAN algorithm: one designed to detect bright sources and the other to detect faint sources. The reason for that is the need to work in two different ranges of cluster formation hyperparameters depending on the type of sources we aim to detect. That is why there is a first round of the algorithm executing with restrictive set of the hyperparameters (restrictive means a high value of *min_samples* for a low value of *eps*) and a second one taking the noise events of the first one to find fainter sources with a less restrictive set of hyperparameters. This procedure also leads to an intrinsic classification of the resulting sources between brights and faints.

At the very beginning, the first approach we took to perform source detection was a single DBSCAN execution over the cleaned event list. The results showed that, when setting restrictive hyperparameters, the method was able to detect bright sources individually but was also unable to detect fainter sources. Otherwise, when setting permissive hyperparameters we obtained a

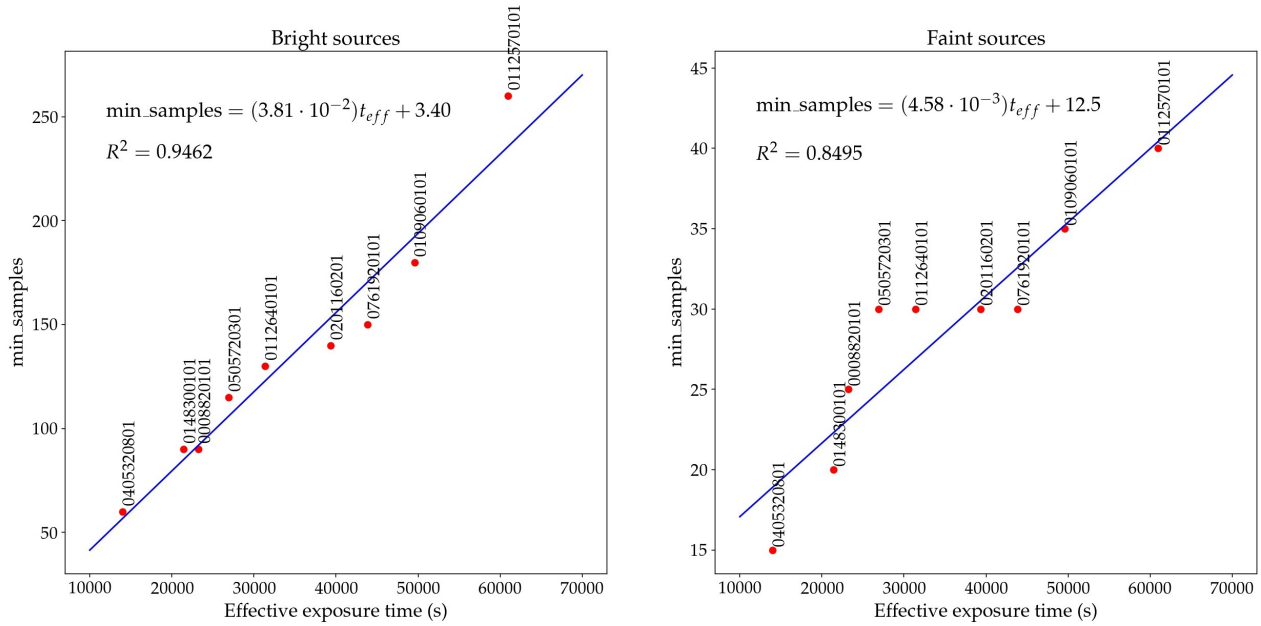

Figure 3.6: Linear regressions for calibrating *min_samples* in both bright and faint sources rounds in DBSCAN-based thread.

larger amount of faint sources, but, as a drawback, the method began to link brighter sources together. Those linkages were critical in overcrowded regions where the count rate is high enough. That troubles lead us to try a double execution method, in which we would benefit from both hyperparameters sets advantages.

A high value of *eps* will allow the algorithm to find more extended sources. In the first round of the pipeline we have set it to 120, which has dimensions of spatial coordinates: one unit corresponds to 0.05 arcsec. The value of *eps* in the second round will be 100. Both have been obtained by fitting its value manually to obtain relevant results on source detection.

As for *min_samples*, since it has dimensions of number of counts, and the number of counts of any observation depends on its exposure time, a relation between *min_samples* and the exposure time is needed.

In Fig. 3.6 it is shown how *min_samples* is determined from the exposure time using linear regression. For this method to be applicable, it has to be probed that the relation between the exposure time and the number of counts is linear.

Assuming an experiment in which the number of photons arriving the detector at an specific window of time Δt is measured. The number of photons arrived is an aleatory variable (X) following a Poisson Probability Distribution Function. Then, the probability for this number to be k is given by eq 3.1:

$$P[X = k] = \frac{e^{-\lambda}\lambda^k}{k!} \tag{3.1}$$

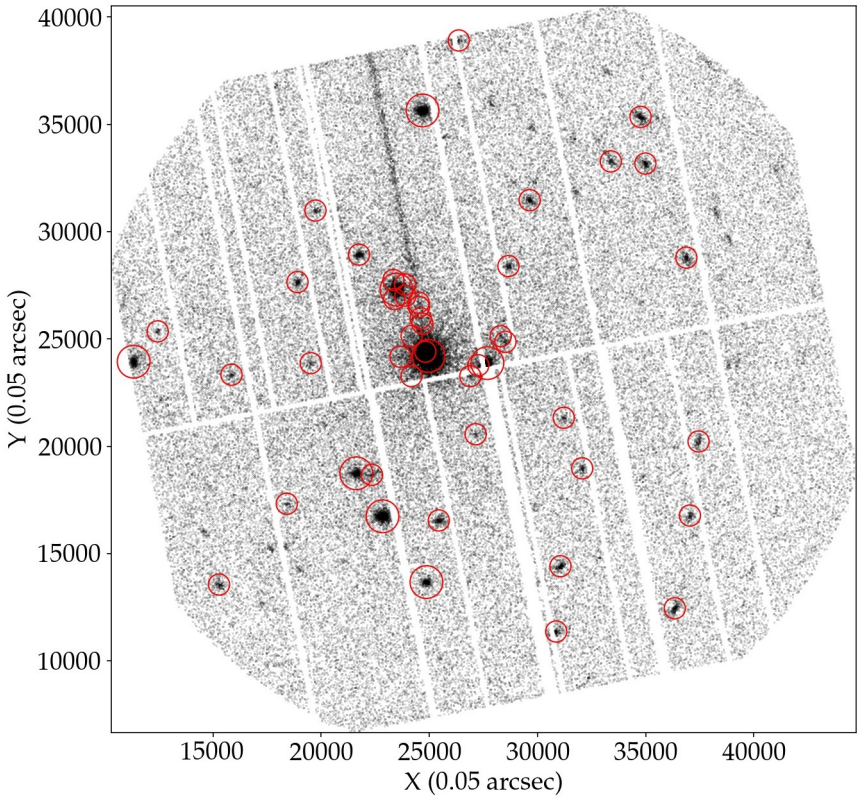

Figure 3.7: Example of an output of the DBSCAN-based source detection thread for the observation ID: 0690200301. More examples are provided at the annexed.

The expectation of this variable is $E[X] = \lambda$. So it is expected to get λ photons in a Δt window of time. If the experiment is repeated, under the same conditions, with a time window of $2\Delta t$, to calculate the expected number of photons arriving, the window can be divided in two with equal time duration Δt. In consequence, it ends up being two windows similar to the one of the first situation. Since before the expectation was λ, now that there are two identical windows, the expectation will be $\lambda + \lambda = 2\lambda$. As it has been probed, doubling the time produces the double amount of photons. This reasoning has been done for $t = 2\Delta t$ but it can be easily generalized to $t = \gamma\Delta t$ for $\gamma \in \mathbb{N}$. In conclusion, the relation between the exposure time and the number of photons detected must be linear, so linear regression is applicable in this situation.

The results of both brights and faints linear regressions can be consulted in Fig. 3.6. The procedure to obtain samples to fit both lines has been manual fitting the *min_samples* parameters of represented observations (each observation is named using its ID). The manual fit of each observation is not a difficult task, it consists on trying different values of *min_samples* until the best result is reached. Remember this techniques are unsupervised machine learning, so there is no loss function to optimize to get the hyperparameters. Also notice that the X axis of the linear regressions is the "effective" exposure time, which means that the intervals removed

in flaring background cleaning are not considered in the total exposure time. Otherwise, the method will be sensitive to the amount of time the observation was affected by flares.
Once the hyperparameters of the bright and faint round of DBSCAN algorithm are set, the thread is ready to detect sources. Then, we need to perform some data integration tasks in order to join sources detected in both DBSCAN rounds (see Fig. 3.5). This type of tasks are omitted in this document since they consist on purely coding stuff.

3.4 OPTICS-based source detection

The next thread for detecting sources in the pipeline uses OPTICS algorithm.

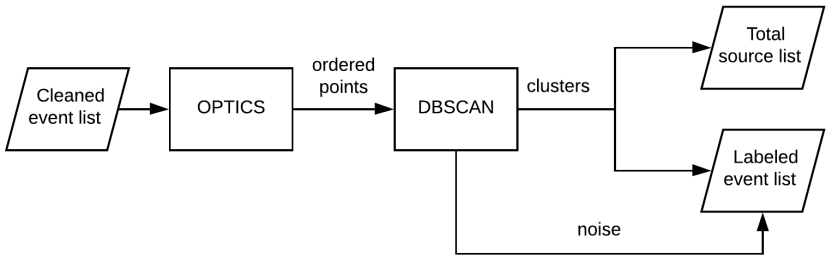

Figure 3.8: Data flow scheme for OPTICS-based source detection thread.

The main problem that arises when applying DBSCAN on this domain, is that it is unable to adapt cluster formation requirements to regions with different count densities. Because of that, we had to choose between good bright or faint detection. The way OPTICS transforms the space before ordering the points (see section 2.3) make us expect that it will be able to attenuate this effect.
As explained in section 2.3, OPTICS do not perform clustering until the end, it just gives an specific ordering of the points that can yield to a cluster structure. To obtain the clusters, we execute DBSCAN right after it. This thread is an improvement of the first one with the use of OPTICS before DBSCAN clustering, with the aim of making it adaptable to regions of different count densities. An scheme of the process can be consulted in Fig. 3.8. We can see how the data collection process is much simpler since all the sources are detected at once, with no need to find independently bright and faint ones, thanks to the capability of OPTICS to obtain clusters of different densities.

The hyperparameter selection in this case has been similar to the DBSCAN-based thread. Since *max_ eps* is an upper limit used by the algorithm to omit calculations and execute faster, we have set the value used in the previous thread to detect bright sources. That is to say that

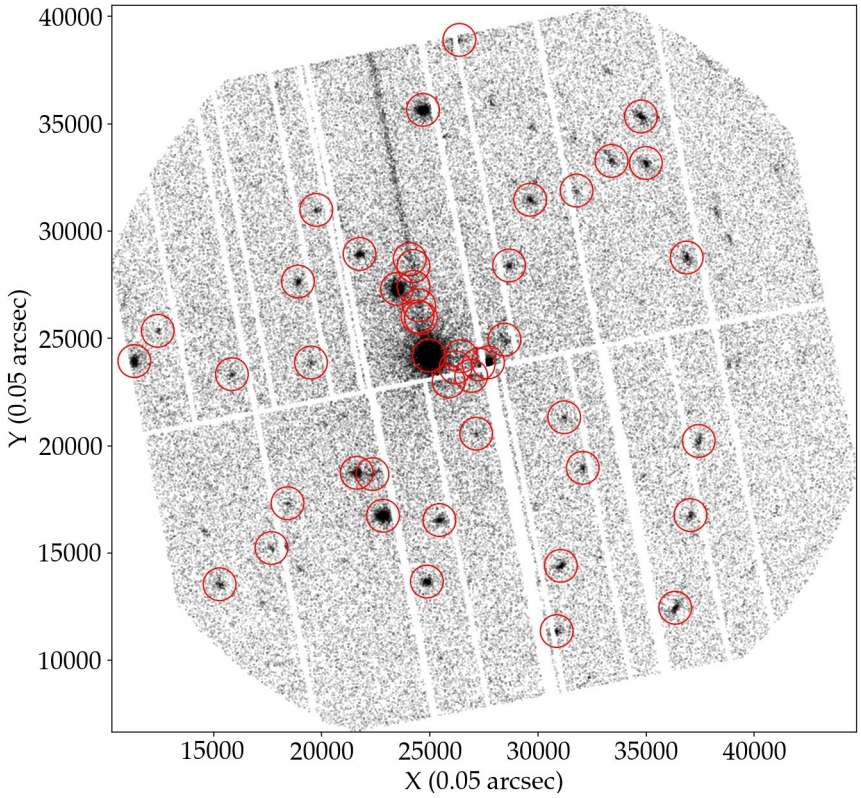

Figure 3.9: Example of source detection given by OPTICS-based thread. E.g. observation ID: 0690200301.

we are not interested in extend the neighborhood of each event longer than in DBSCAN-based thread. Which will produce undesirable neighbor sources linkages.

max_eps (0.05 arcsec)	min_samples (counts)
120	$0.0007t_{eff} + 12.5$

Table 3.3: Hyperparameters selected for DBSCAN in flaring background removal stage.

The *min_samples* has to be selected as a linear function of the exposure time. The coefficient has been selected slightly higher than the one obtained by linear regression in the faint round of the previous thread.

The final combination of hyperparameters is shown in table table 3.3. The hyperparameters used in DBSCAN are the same used in OPTICS (with $eps = max\_eps$).

This combination of hyperparameters has shown robustness on the results obtained in a wide range on observations. They may not be the optimal ones, but, as stated in previous sections, unsupervised learning makes it difficult to find the perfect hyperparameter set.

In Fig. 3.9 we show the position of the sources detected by OPTICS-based thread using the same example as DBSCAN-based thread.

3.5 HDBSCAN-based source detection

The last thread developed in this work uses HDBSCAN algorithm to perform clustering over the cleaned event list. However, once HDBSCAN produces its clusters, the method does not ends. In this situation, we apply a second round of DBSCAN to each cluster produced by HDBSCAN independently (see Fig. 3.10). The reason is different than in the case of OPTICS. While OPTICS forced us to apply a second algorithm to perform clustering end to end, HDBSCAN is able to complete the process by its own, but we use DBSCAN to attenuate some problems that arise when using HDBSCAN.

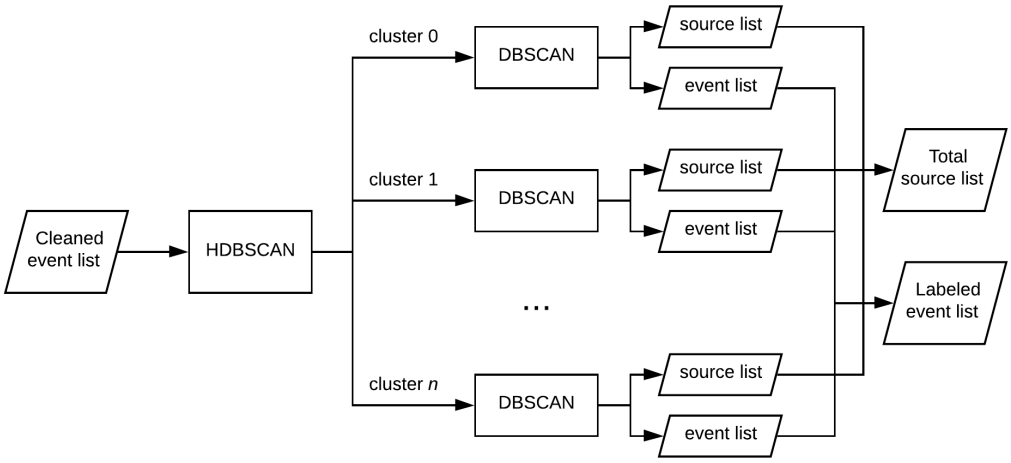

Figure 3.10: Data flow scheme for HDBSCAN-based source detection thread.

In order to understand the reason why a layer of DBSCAN has been situated after HDBSCAN, one must observe Fig. 3.10. The output given by HDBSCAN is an specific cut on a cluster hierarchy (see section 2.3), which provokes that in some regions of the observation the source detection keeps unresolved. In other words, the clusters formed are too sparse (lots of points belonging this clusters should be considered as noise) and sometimes there is more than one source inside each of them. This phenomenon has been observed in regions where there are not too many sources: underpopulated regions. In contrast, in regions where there is a high density of counts produced by extended sources overlapping, HDBSCAN shows better results than DBSCAN and OPTICS threads.

In order to solve the problem presented by HDBSCAN in sparse regions, we apply DBSCAN independently to each cluster detected by HDBSCAN. That way, we keep the capability of HDBSCAN detecting sources in dense regions, while forcing sparse clusters to achieve the desired level of resolution, given by DBSCAN hyperparameters. We can see an example of an output given after multiple DBSCAN layer in Fig. 3.12.

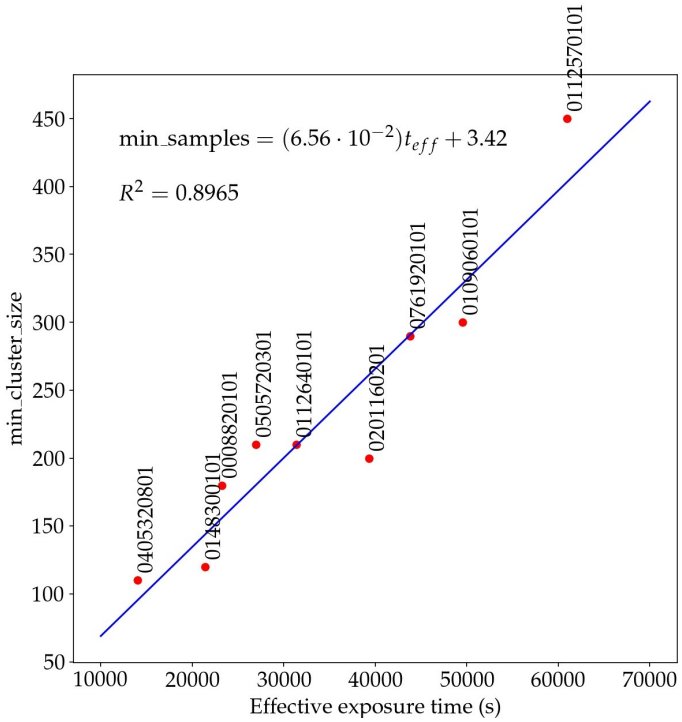

Figure 3.11: Calibration of *min_cluster_size* in HDBSCAN source detection thread using linear regression.

The hyperparameters of HDBSCAN are slightly different from DBSCAN and OPTICS, as discussed in section 2.3. The first one to be discussed is *min_cluster_size*, which is the minimum number of counts a group needs to be considered a cluster. Having probed that the relation between the number of counts and the exposure time is linear, the *min_cluster_size* must be calibrated by linear regression (see Fig. 3.11).

The next hyperparameter to be commented is *min_samples*. By default, it is set with the same value than *min_cluster_size*. When *min_samples* is lower *min_cluster_size*, the effect will be that more points (that normally would be considered noise) will now be included in a cluster. In other words, we would be more permissive for points located in sparser regions to be considered part of a cluster. In this case, *min_samples* is set with the same value than *min_cluster_size*, otherwise, the issue of HDBSCAN in sparser regions would be enhanced (see Fig. 3.12 top-right image).

The *cluster_selection_eps* has been approximated using the "elbow" criteria and then manually fit, since it is analogous to *eps* in DBSCAN. The complete set of hyperparameters can be consulted in table 3.4.

As for DBSCAN hyperparameters, they are set according those obtained in the first thread. Since the aim of DBSCAN in this thread is helping HDBSCAN in regions where it is not able to reach enough resolution, the idea is that, in those regions, the thread will return similar

results than DBSCAN-based thread. The selected expression must be a linear equation. The coefficients exposed in table 3.4 are the ones we have chosen for this thread and have presented good results in a wide range of observations, specially in those populated by extended sources.

HDBSCAN	min_cluster_size	$0.00656t_{eff} + 3.42$
	min_samples	$0.00656t_{eff} + 3.42$
	cluster_selection_eps	60
DBSCAN	min_samples	$0.0006t_{eff}$
	eps	120

Table 3.4: Hyperparameters selected for HDBSCAN-based thread.

In general, we have observed over a wide range of observations, that HDBSCAN-based thread shows the best results out of the three threads designed. The reason behind, which has been so commented, is that it is able to detect sources in both sparse and crowded regions. By its own, HDBSCAN has only showed good results on overpopulated regions. That is the reason why, at some point of the project, we deliberate about removing it from the pipeline, since it, regardless the set of hyperparameters used, was not able to achieve source detection over low background and point-like sources. Nevertheless, when we tried it over those overpopulated regions, we discovered its outstanding performance on this type of observations. Since then, we concluded that the best source detection thread would combine both good results achieved by DBSCAN on sparse regions and the outstanding ability of HDBSCAN when dealing with crowded ones. The final solution was this thread (see 3.10).

Having said that, we would like to warn that, it is not that HDBSCAN is the best algorithm on this domain, but that this thread has shown the best results out of the three designed. We have called this thread HDBSCAN-based because the difference between the others is the use of HDBSCAN, but their good results must be assigned to both HDBSCAN and DBSCAN, and obviously to this specific combination.

Notice that OPTICS does not take part on this solution, since it shows similar results than DBSCAN and therefore does not include a significant advantage. Also, as it will be discussed in section 3.7, the use of OPTICS would considerably increase the total execution time.

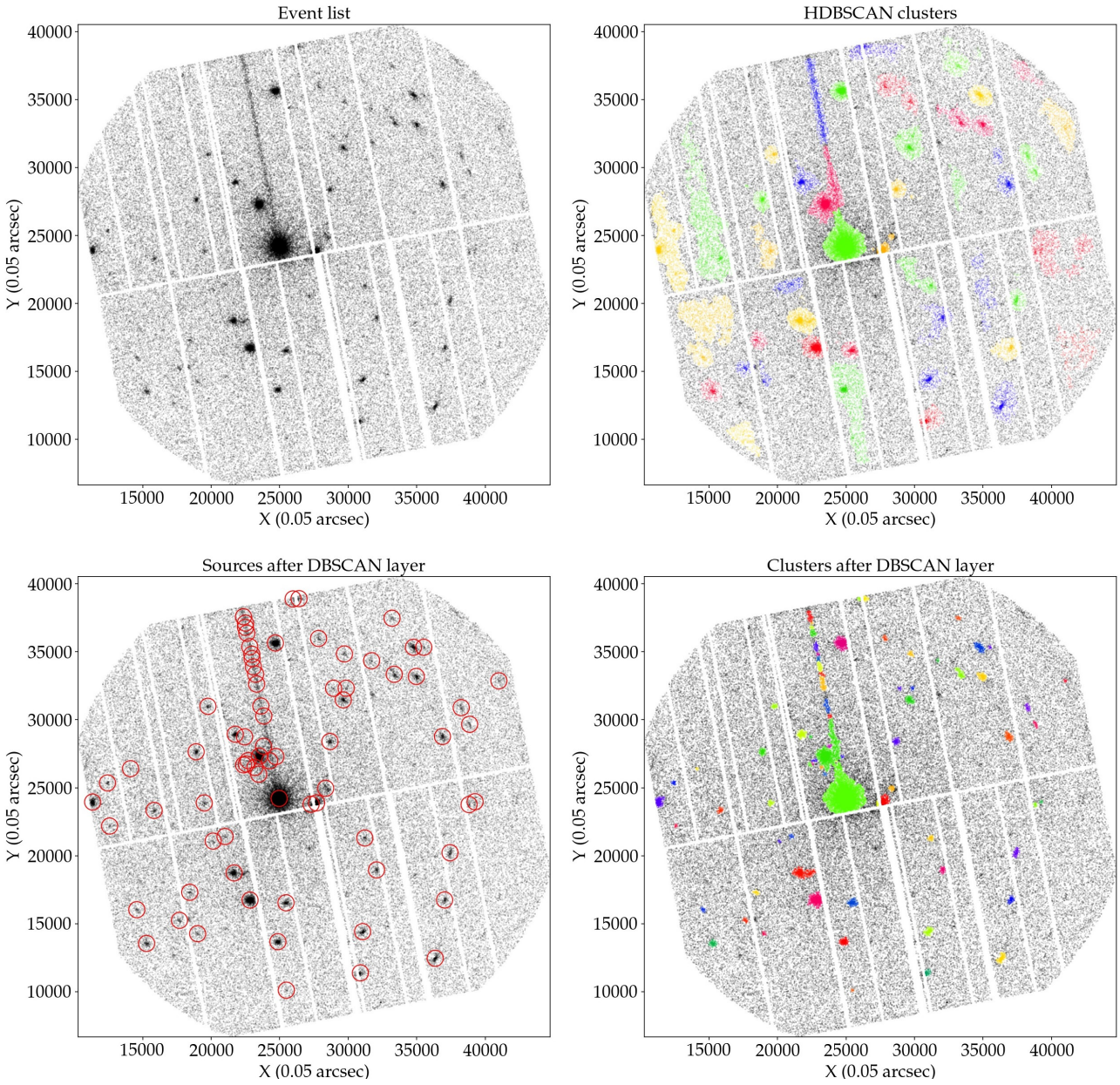

Figure 3.12: Example of an output of the HDBSCAN-based source detection thread. (*Top-left*): Input event list. (*Top-right*): Clusters output right after HDBSCAN execution. (*Bottom-left*): sources detected after DBSCAN layer. (*Bottom-right*): clusters output after DBSCAN layer. E.g. observation ID: 0690200301.

3.6 Extended sources

When dealing with crowded regions of the sky, the amount of counts concentrated in an specific region is such that it can be difficult for the clustering algorithms to detect sources. The example provided in this document to illustrate each step of the pipeline and validation is the typical EPIC observation with a low background and point-like sources, but the input is not always that simple.

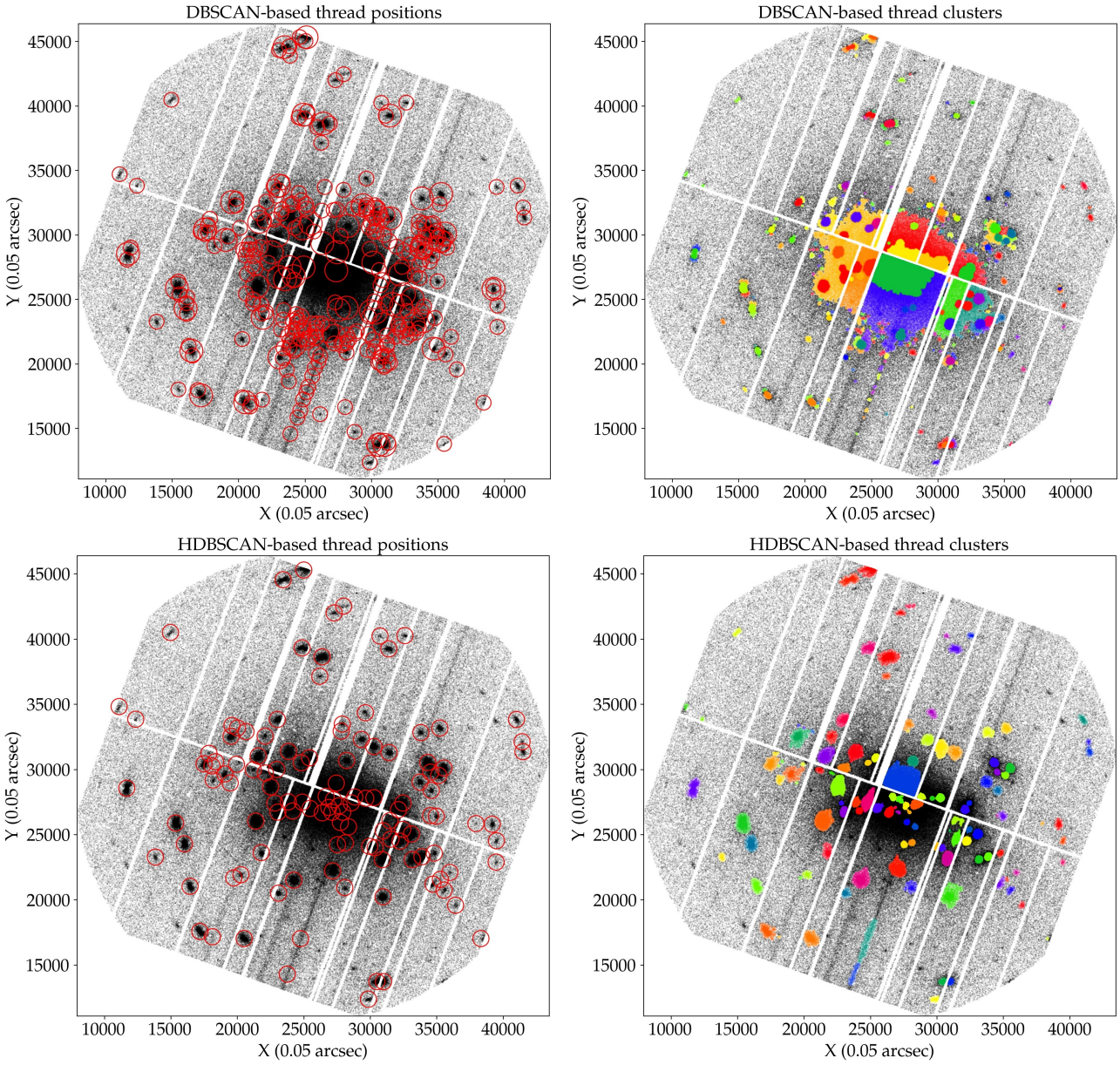

Figure 3.13: (*Top*): DBSCAN-based thread: sources and clusters. (*Bottom*): HDBSCAN-based thread: sources and clusters. E.g. with observation ID: 0112570101.

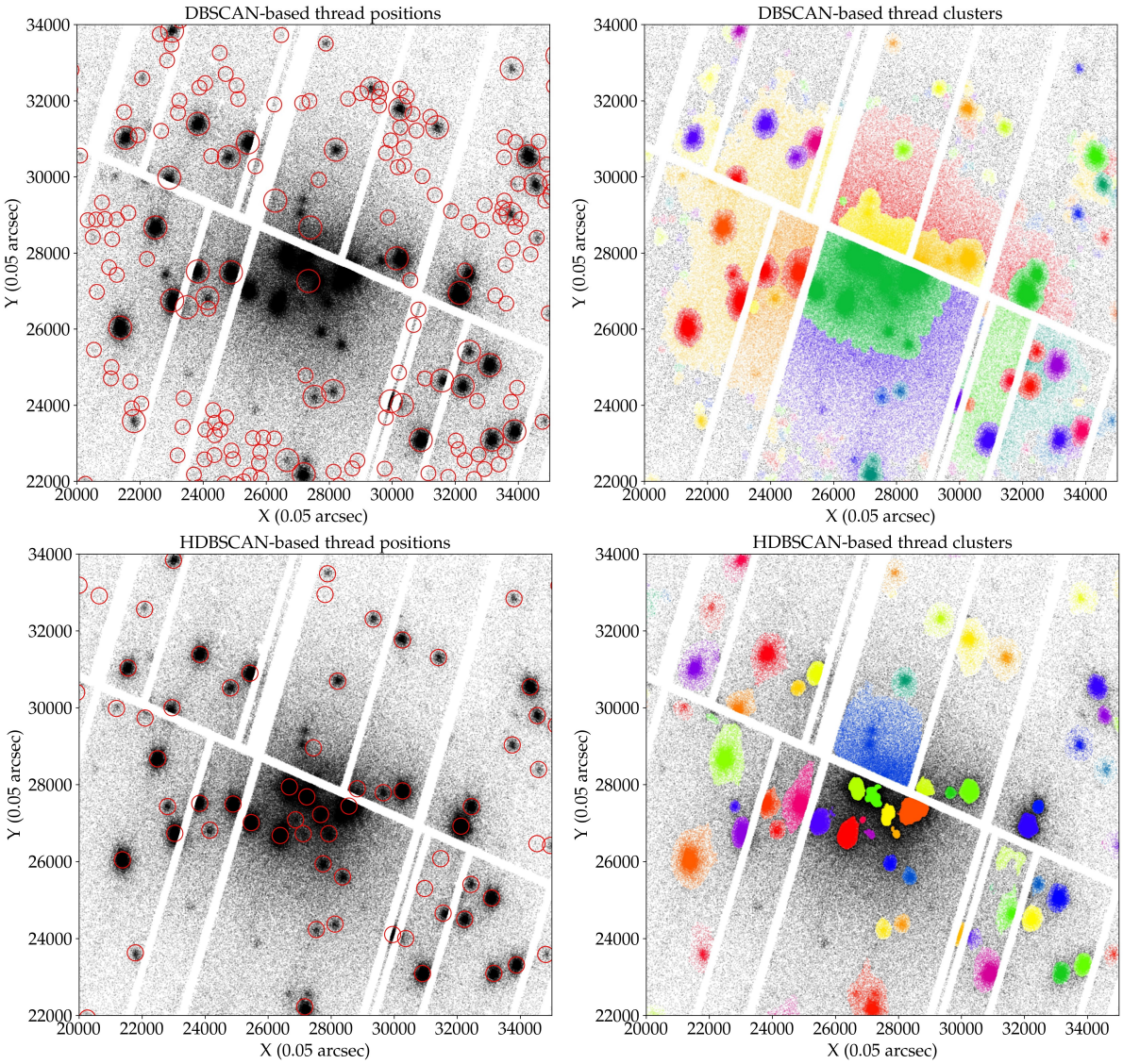

Figure 3.14: Zoom in the central region of Fig. 3.13 observation.

However, in this section we present an observation of M31 galaxy (as known as Andromeda galaxy). Those observations present an overcrowded region of the sky in the center of the field of view of the cameras. This kind of observations reveal the weakness of DBSCAN-based thread and make evident the need for designing HDBSCAN-based thread.

We give as example to comment the observation shown in Fig. 3.13, and its zoom in Fig. 3.14. We can see how DBSCAN-thread is not able to resolve individual sources in the center of the image. In addition, a group of spurious faint sources is formed surrounding it. Whereas HDBSCAN thread can resolve both situations quite deeper. The DBSCAN layer of HDBSCAN thread (see section 3.5) is important to try to resolve individual sources in sparser regions. While HDBSCAN layer is responsible for the deep resolution in the center of the image.

3.7 Computational performance

This section has the objective of measuring how much time is taken by every part of the pipeline. In order to study the dependency of the execution time with respect to the number of events in the input data, for each part of the pipeline, a scatter plot of the number of counts versus the execution time has been drawn. We show the results obtained in Fig. 3.15. The best performance is shown by the flaring background part. This is something really positive because this part receives the hole event file as an input. The length of an observation's event file depends on the exposure time and the region of the sky where the telescope is pointing to, so it is very variable. The largest files can have a length of the order of 10^6 events.
On the other hand, OPTICS-based thread is by far the least efficient when the number of counts increases. It is recommended to maintain OPTICS thread inactive for observations with more than 150000 cleaned events.
As for DBSCAN and HDBSCAN-based threads, the performance is more than acceptable considering the size of the input data this threads have to bear with. Even with 1 million of cleaned events as input, which is very unlikely, both will return the output in around a minute. Normally the input will be of the order of 10^5, and the execution time will be of approximately 10 or 20 seconds.
We shall warn that, despite the flaring background thread receives considerably larger size of input data, in Fig. 3.15, the execution time for DBSCAN, HDBSCAN and OPTICS threads has been tested with uncleaned events, so each thread is tested under the same conditions than the flaring background removal and we can control the input size. Otherwise, the source detection threads would have received significantly lower amount of data than the one shown in this plot.

Input size	Execution time
410793	4 min 12 s
151703	2 min 14 s
194986	2 min 5 s
185874	3 min 2 s

Table 3.5: Computational performance of the background and source density estimation maps calculation.

Another warning to make, is that the execution time of background and source density maps (see section 4.1) could take about 2 or 4 minutes in a range of input sizes, but this execution time is not clearly related to the input size (results given in table 3.5).
In this section, it is necessary to comment that the execution time is in the majority of the

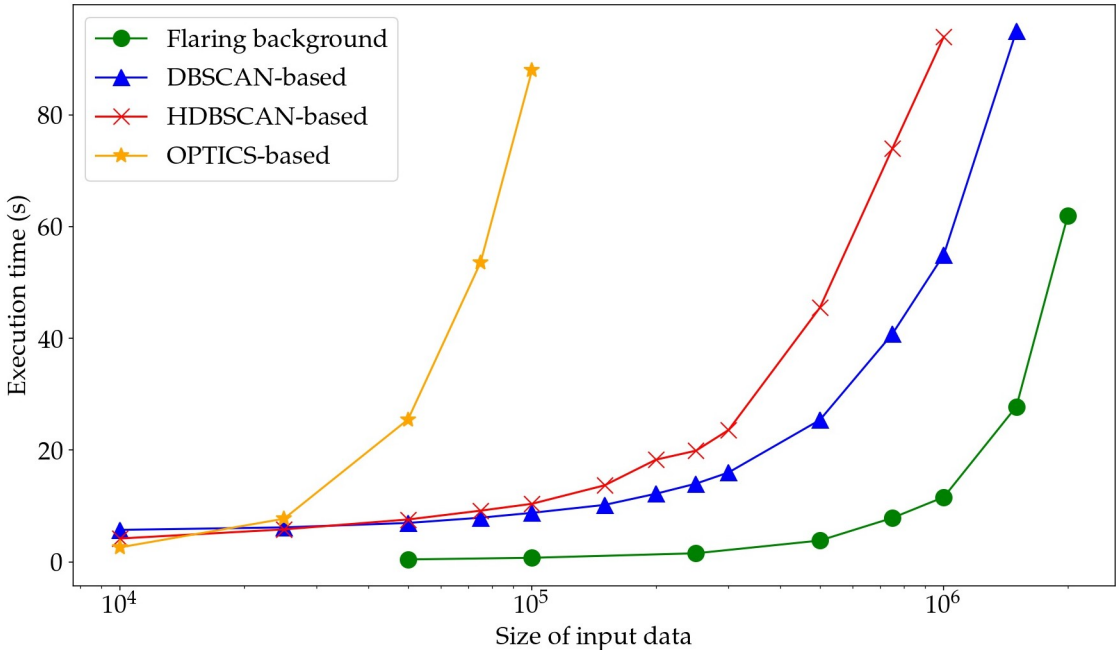

Figure 3.15: Execution time versus size of input data for each part of the pipeline.

cases justified to achieve the results. In principle, DBSCAN and OPTICS threads are not improvable, the same to say about the flaring background removal. But we are also aware that HDBSCAN-based thread can be optimized by parallelizing multiple DBSCAN layer executions, since they are independent from each other. Also, the computation of the background density map can be optimized by omitting some calculations that will remain unused.

Chapter 4

Validation

4.1 Significance metrics

So far, we have presented three methods for performing source detection. The aim of this part of the pipeline is to determine the level of confidence that can be assigned to each detection. Like the error level of every measurement in physics. Even the XMM-Newton pipeline source detection gives a probabilistic value for each detection: the Maximum Likelihood. This part should not be mismatched with the validation stage. The validation consist on comparing these alternative and novel clustering methods to the actual results of the XMM-Newton pipeline.

4.1.1 Density-based significance

The first of the two types of metrics evaluated to measure the significance will be based on the idea that a source should have a considerably higher count density than its surrounding background. The greater the difference, the more probable it is that the detection truly corresponds to a real X-ray source. This metric is the more natural, since the sources has been detected considering count density criteria.
In order to calculate this metric on a source, we need two quantities:

- The count density of the source.
- The count density of the background surrounding the source.

In other works, the significance of a clustering detected source was obtained counting events of the source and the surrounding background surface (see section 2.3.6 to consult how the significance is measured by [15] and [23]). Here, another approach to the problem is proposed.

The reason why the method used in [15] and [23] is not fully applicable in this project is based in three problems that arise when using it in XMM-Newton observations:

- When defining the region of the surrounding background of a source, other neighbor sources may fall inside. So, when counting the number of noise events there, the result will be less than if neighbor sources were not there. It is a source of error.

- When there are sources at the edge of a CCD chip (see Fig. 2.2), some fraction of the background events of a source will belong a different CCD. As it is known, instrumental background may depend on the CCD where the source is located, that is why including background counts from a different CCD is another source of error.

- Between CCDs of the EPIC camera, there is empty surface with no detected events (itcan be seen in any of the observation examples provided). So, if the background region of a source includes part of empty surface, less events will be counted compared with sources in the central region of a CCD. Which is another source of error that can not be controlled.

On balance, the method used on *Fermi* γ-rays ([15] and [23]) introduces a few complications due to sources surrounded by neighbor sources and sources located on the edge of a CCD. Now that the need of a different method has been argued, we are proposing a method that will intend to solve these problems as much as possible. However, this method is not going to be perfect, whereas some of the previously mentioned sources of error will be attenuated.
First of all, two density maps are obtained using Kernel Density Estimation with a gaussian kernel: one for the events labeled as background and the other for the events labeled as source. These last ones to be considered as part of a detected cluster.
The density distribution function of a set of points $X = \{x_1, x_2, ..., x_n\}$ can be approximated using KDE (Kernel Density Estimation) with the expression:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{4.1}$$

Being n the number of points, h the bandwidth or smoothing parameter and K the kernel, in this case a gaussian function. With this three elements, one can get the density estimation in any point of space x generated by a distribution of points X. We can see this process as a model that works under the hyperparameters h and K and is fitted with a set of points X, then is able to predict the density of any point x. In this project, the fitting set will be formed by the background events, in the case of the background density map and the source events in the case of the source density map.

Each of the fitting points $x_i \in X$ is substituted by the kernel. In this case, a gaussian distribution:

$$x_i \to N(\mu = x_i, \sigma = h) \tag{4.2}$$

Being $N(\mu, \sigma)$ a two dimensional gaussian distribution centered in $x_i = (\text{DETX}, \text{DETY})$, where DETX and DETY are the detector coordinates of the event x_i. The standard deviation of the distribution is h, which is the bandwidth of the KDE model.
At the end of the fitting, the two dimensional space (DETX, DETY) is populated by a set of overlapping gaussian peaks. Notice that the larger the value of h, the broader the peaks will be, and the smoother the adjust.
When trying to estimate the density in any position of the space x, the model will give the density as the superposition of each of the gaussian peaks (see Fig. 4.1).

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} N_{0,1} \left(\frac{x - x_i}{h} \right) \tag{4.3}$$

So, at the end, the value of the count "density" is given as a the value of the probability density function in this point. The advantage of doing so, is that density can be calculated as an intensive magnitude, giving only the coordinates of the point. Rather than extensively, where we would need to define a region, calculate its area and the number of counts inside. Which is the source of some of the errors exposed at the beginning of this section.
As stated before, following this method, we build two density maps, one for the background and the other for the sources. This way the influence of neighbor sources when estimating the background will be attenuated. However, it is necessary to advice that there will still be influence, it will be explained below.

After that, we define a grid with 200×200 points distributed in the observation (40000 in total), which means that 40000 density measurements will be made over the whole filed of view of the observation. Using neighborhood criteria, we established whether each of this measurements belongs to background or source regions (and the specific source), It is also evaluated the CCD where are located in. Notice that, as this grid is too discrete, some faint sources will not be represented by any measurement of the grid, whereas others will agglutinate a bunch of them. For those faint sources not represented by any grid point, we evaluate its density on its centroid. The centroid of a source i with n counts is given by:

$$c_i = \left(\frac{1}{n} \sum_{j=1}^{n} \text{DETX}_i, \frac{1}{n} \sum_{j=1}^{n} \text{DETY}_i \right) \tag{4.4}$$

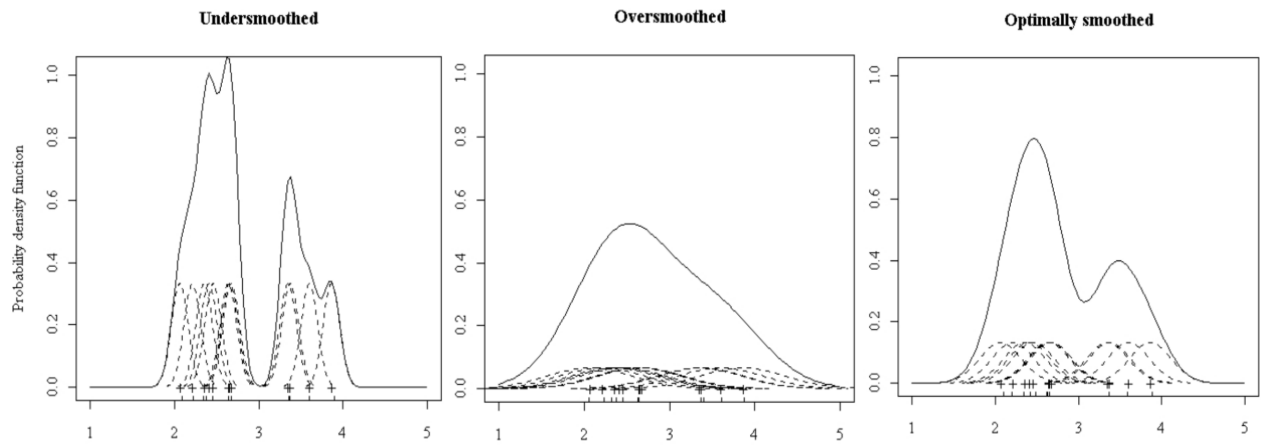

Figure 4.1: Illustration of gaussian KDE smoothing and its dependence on the bandwidth. Image from link.

Given a source, to calculate its significance, we need the density of the source and its surrounding background. With all the elements defined above, the density of the source is calculated as the median of all the density measurements of the grid belonging the source. This is the reason why the density is measured over a grid and not over every event. If the density was measured over each and every event, apart from being computationally heavy, denser areas would be overrepresented in the median calculation. The same argument is given for the case of the background.

As for the surrounding background, we establish a rectangular region of amplitude $14\sigma_x \times 14\sigma_y$, and centered in the centroid of the source as surrounding background (see Fig. 4.3). The density of the background is calculated as the median of every grid measurement belonging the background, located in the surrounding region and also in the same CCD where the source is located. We use the median with the aim of giving this calculation robustness over outliers. Density outliers can be provoked by measurements located at the edge of a CCD and measurements located at the edge of a neighbor position. Since the majority of the measurements are going to be performed over none of this situations, the median is expected to avoid that unwanted effect.

Once we have calculated a value for the source density (d^{src}) and the background density (d^{bkg}), the significance of a source i (s_i) is calculated using the following formula:

$$s = \frac{1}{1 + 10e^{-\frac{d_i^{src}}{d_i^{bkg}}}} \tag{4.5}$$

It is known as a sigmoid function and its objective is to maintain the significance on the range

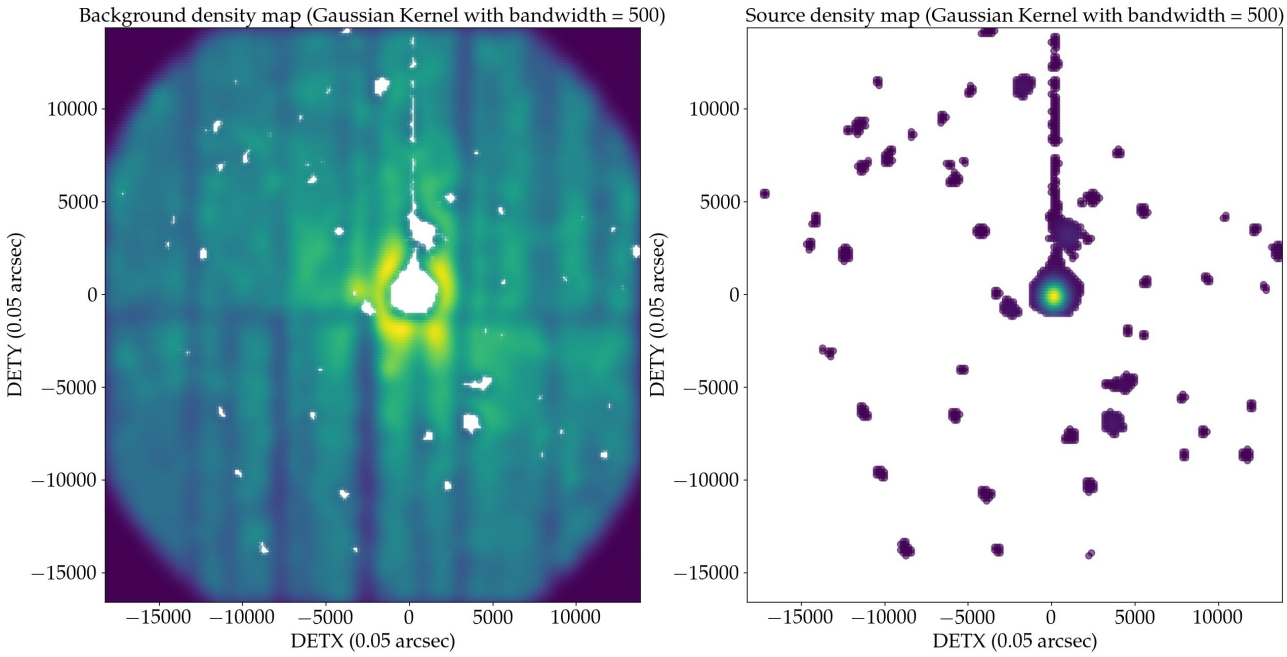

Figure 4.2: Measurements of the background and source density independently. E.g. with observation ID: 0690200301.

between 0 and 1. The coefficient 10 is there just for flatten the sigmoid, if it were not there, it would be too easy for a source to reach $s = 1.00$.
It is time to reevaluate the three problems this method was aimed to attenuate:

- The influence of neighbor sources was critical since they occupied part of the surrounding area that was considered as fully background. Now, we only measure the density of the background in those regions where it is sure that there is not a neighbor source. It is true that the background density measured in the surroundings of a neighbor source is affected by the effect of the bandwidth, but we avoid those outliers since there are much more measurements far from this problematic regions. Taking the median between all of them guarantees the robustness.

- The problem of having part of the surrounding background in another CCD is solved by only considering those measurements taken in the same CCD when calculating the median.

- The empty region between CCDs affects in the same way that holes provoked by neighbor sources. The argument is the same, those anomalous measurements are not hegemonic and taking the median will prevent their influence.

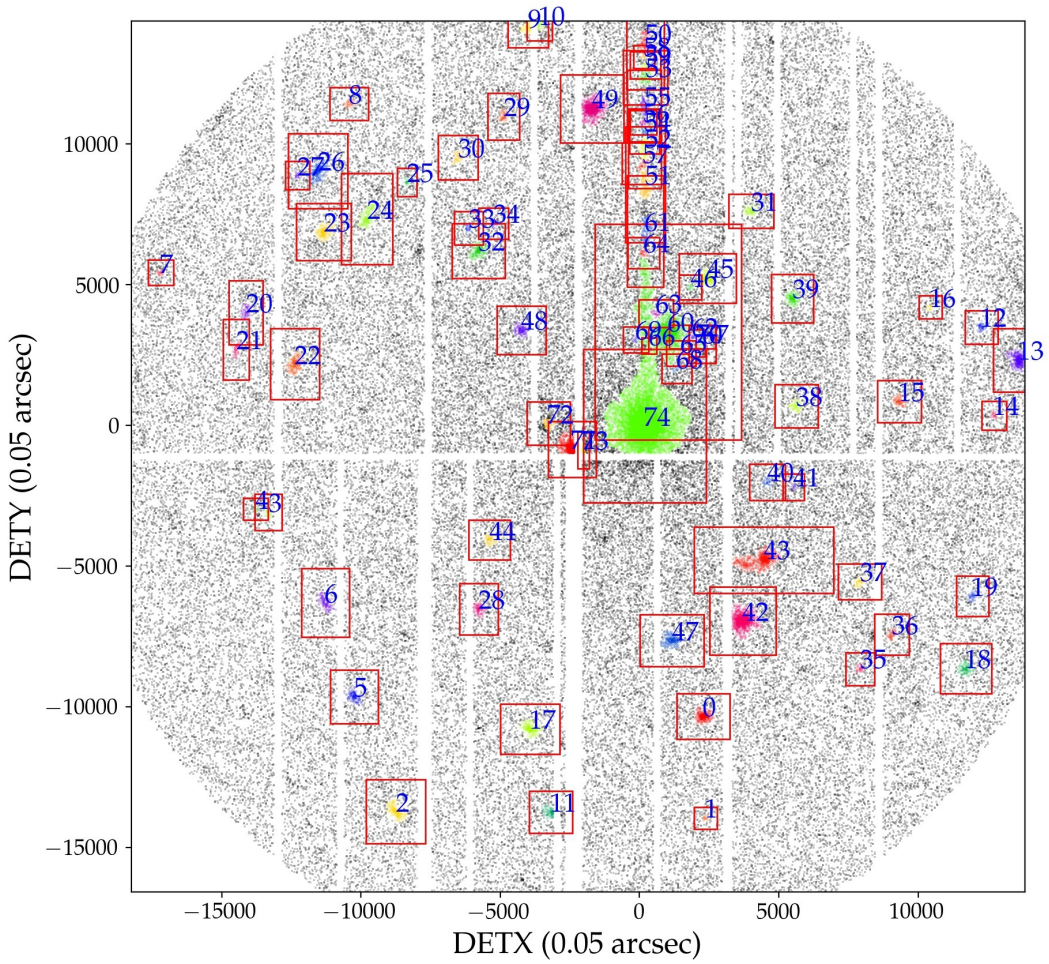

Figure 4.3: Surrounding background regions over each detected source using (e.g.) HDBSCAN-based thread output source list. E.g. observation ID: 0690200301.

4.1.2 Spectra-based significance

The other significance metric proposed in this work has to do with the energy spectrum of the source compared to those from the background. The idea behind this metric is that a well detected source will have a different energy distribution than its surrounding background. The spectrum of the source and the background are expected to be different because they are generated by different physical phenomena. To calculate this metric, we need to find a way of calculating the correlation between the spectra of the source and the spectra of its surrounding background.

Since both spectra are a group of points representing a variable (the energy of the photon), with different sample sizes and no particular ordering, the only way of observing the behavior of the energy in the source compared to the background is plotting both histograms on the same image. However, as we aim to go further and get a correlation metric between them, we

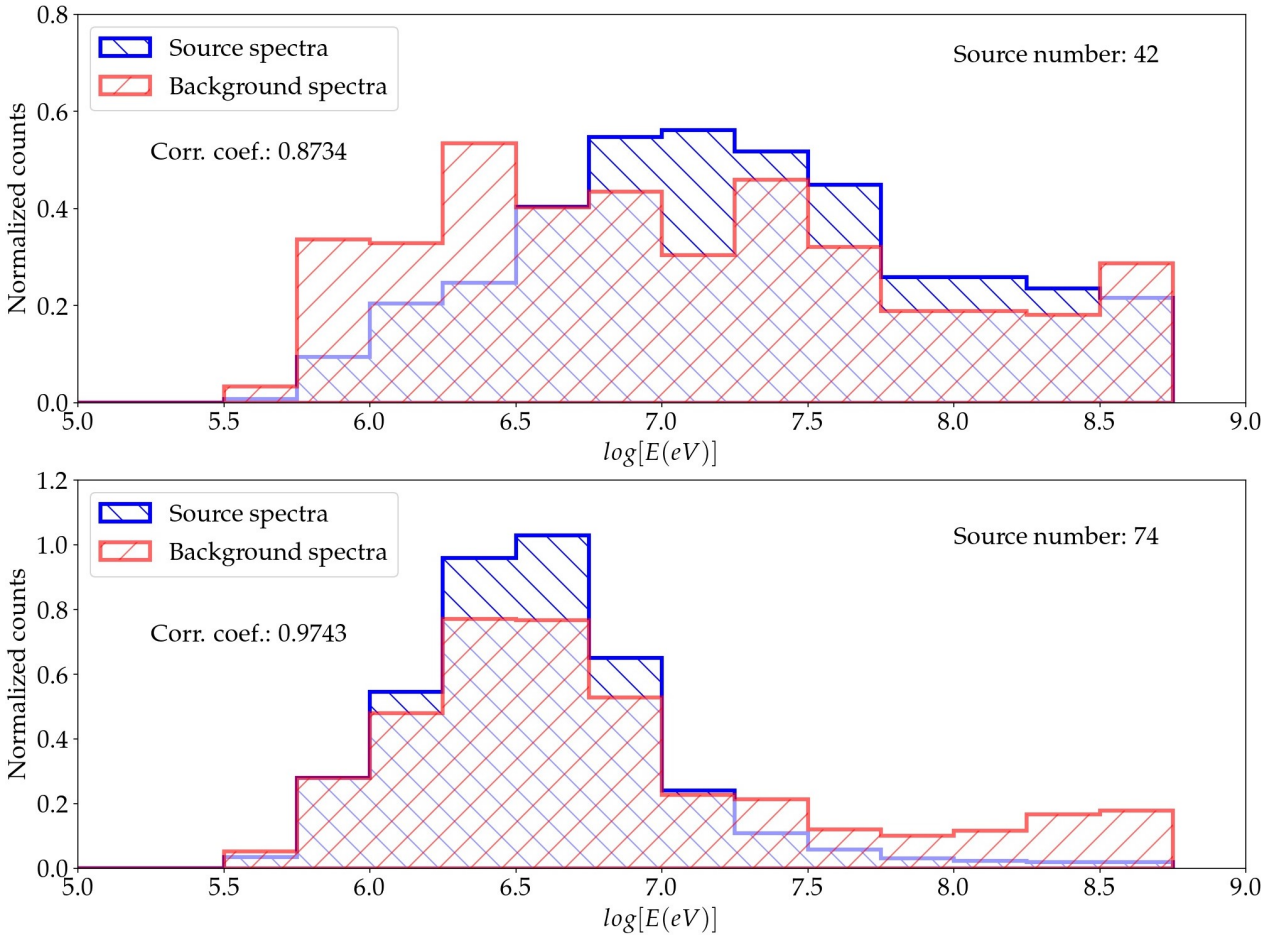

Figure 4.4: E.g. of two spectra correlations using sources 42 (*Top*) and 74 (*Bottom*) from Fig. 4.3. Observation ID: 0690200301.

will define both histograms using identical energy bins, so the amount of events inside each bin is comparable between both spectra and the correlation coefficient between the two histograms can be calculated.

We calculate the correlation between this two histograms using the determination coefficient R^2. In other words, the idea is to probe if the number of counts of source and background in each energy bin are correlated, that is the reason why both histograms must have the exactly same bins. Those bins go from 0 to 8.5 taking steps of 0.5 in a logarithmic energy scale. The correlation coefficient R^2 is given by eq: 4.6:

$$R^2 = \frac{\sigma_{SB}^2}{\sigma_S^2 \sigma_B^2} \tag{4.6}$$

Being σ_{SB} the covariance between source and background histograms, σ_S^2 the variance of the source histogram and σ_B^2 the variance of the background histogram.

In Fig. 4.4, we give two examples of sources from observation of Fig. 4.3 (source id: 42 and source id: 74). We can see how in the source 42 the correlation coefficient (R^2) is equal to 0.8734, whereas is the source 74 it is equal to 0.9743. The 42 does not have a significant correlation with its background, as we can deduce from the histogram plots. So if this coefficient comes together with a density-based significance near 1 ($s > 0.95$), the detection is considered reliable. On the other hand the source 74 spectra is strongly correlated with its background, but its density-based significance goes to 1.00, since it is a really bright source, so maybe what this correlation is telling us is that surrounding events are also produced by the proper source, instead of being background.
As the objective is to build a metric that, the more different the source with respect to its background, the more reliable is its detection, we need to reverse the R^2 coefficient, so the spectra-based significance metric will be equal to $1 - R^2$.
Using this metric, source 42 will have a value of 0.1266 and source 74 will have 0.0257. Which indicates that, in terms of energy spectra, we give the detection of the source 42 more confidence.

4.1.3 Interpretation of both metrics

First of all, it is necessary to establish an idea of what is considered a high/low density/spectra significance metric. Based on our experience, we can consider that a source has a high density-based significance metric when its value is higher than 0.7. Whereas a high spectra-based significance metric will be higher than 0.1. These numbers are just considerations in order to be able to build categories. To be rigorous, each source should be revised individually and its particular situation must be considered. An example that will make this metrics fail are read out streaks, caused by out of time events. Out of time events are caused by the electronics of the detector. In the observation used in this chapter as example there is a read out streak (vertical line). Those events are artificial and not created by X-ray sources in the sky, but under the view of both significance metrics, they can be accepted as a source.

4.2 XMM-Newton SOC pipeline cross match

The aim of this section is checking the position of the sources obtained by this work's pipeline in each of the three threads with those detected by XMM-Newton pipeline from the same observation of an specific region of the sky. In consequence, the position of the sources detected by our clustering pipeline should be converted to sky coordinates (ra, dec), which can be done using metadata from the header of the FITS file.

An example of cross match validation is provided in Fig. 4.5, Fig. 4.6 and Fig. 4.7. There,

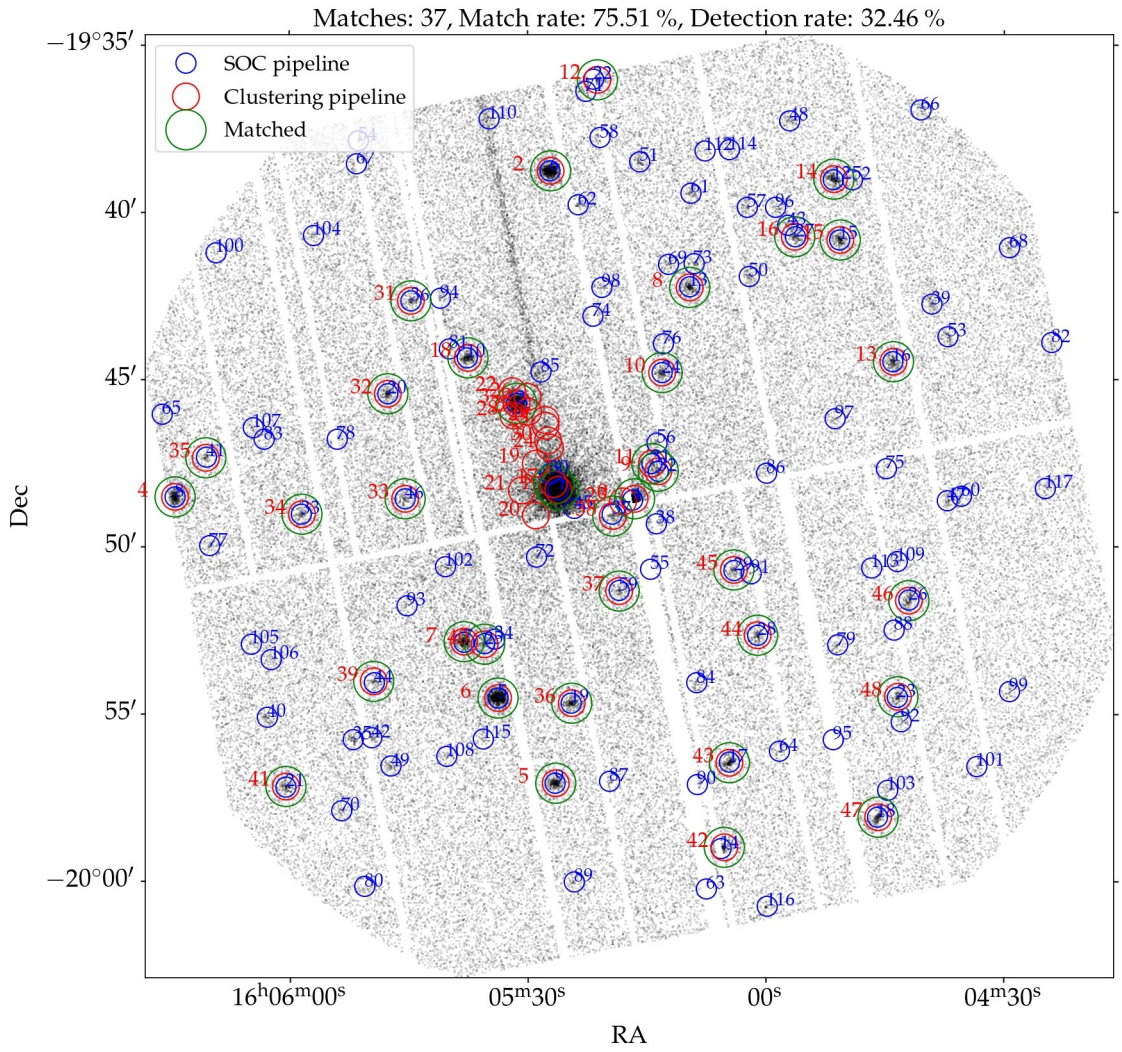

Figure 4.5: DBSCAN-based thread cross match validation. E.g. with observation ID: 0690200301.

one can see that DBSCAN and OPTICS give the approximately the same results. HDBSCAN-based thread stands out as the best alternative It is able to recognize the higher amount of sources, whereas is true that distracts with the read out streak.

In this section, one can check the goodness of each source detection thread visually. But, in order to establish a quantitative criteria to evaluate each detection thread, we have defined two metrics: the match rate and the detection rate.

First of all, we should introduce the concept of "match". A match happens when the position of a clustering detected source coincides with a SOC's pipeline source. The more amount of matches, the better. Mathematically, two sources (one from clustering and the other from SOC) are considered to match when the distance between them is lower than an angular distance threshold. We have set the threshold to 15 arcsec. Using nearest neighbor criteria,

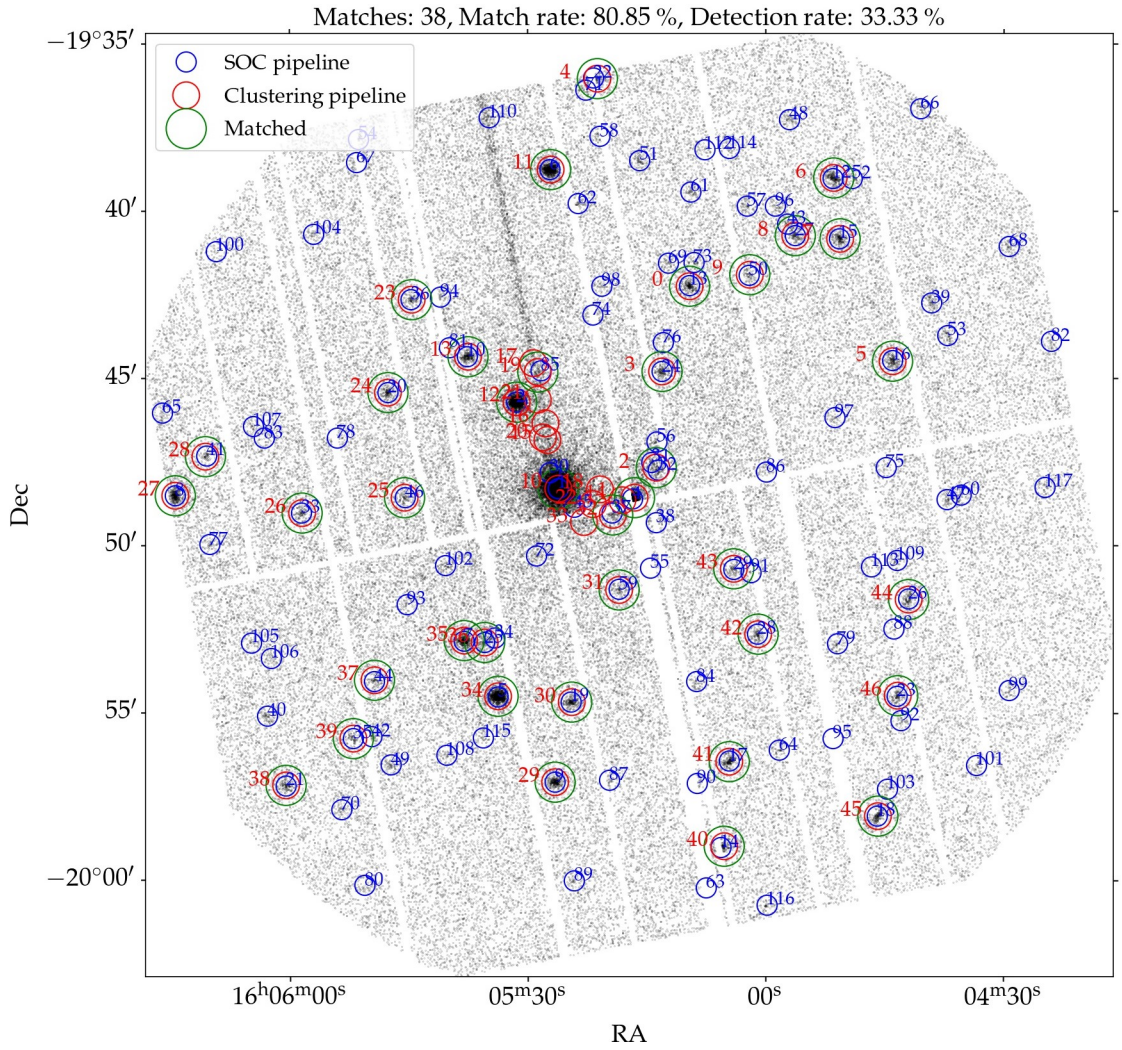

Figure 4.6: OPTICS-based thread cross match validation. E.g. with observation ID: 0690200301.

one can know the nearest SOC source to every clustering source. For every clustering source, its neighbor and the distance between them is saved. Then, a boolean column is defined to save whether if this clustering source has achieved to mach or not. Once the number of matches is obtained, we can define the two metrics:

- Match rate: is the number of matches obtained over the number of sources detected by clustering. It is a way of measuring the percentage of the clustering detected that end up matching with the SOC pipeline.

- Detection rate: is the number of matches over the number of sources detected by the SOC's pipeline. The aim of this metric is to measure the percentage of actual sources detected.

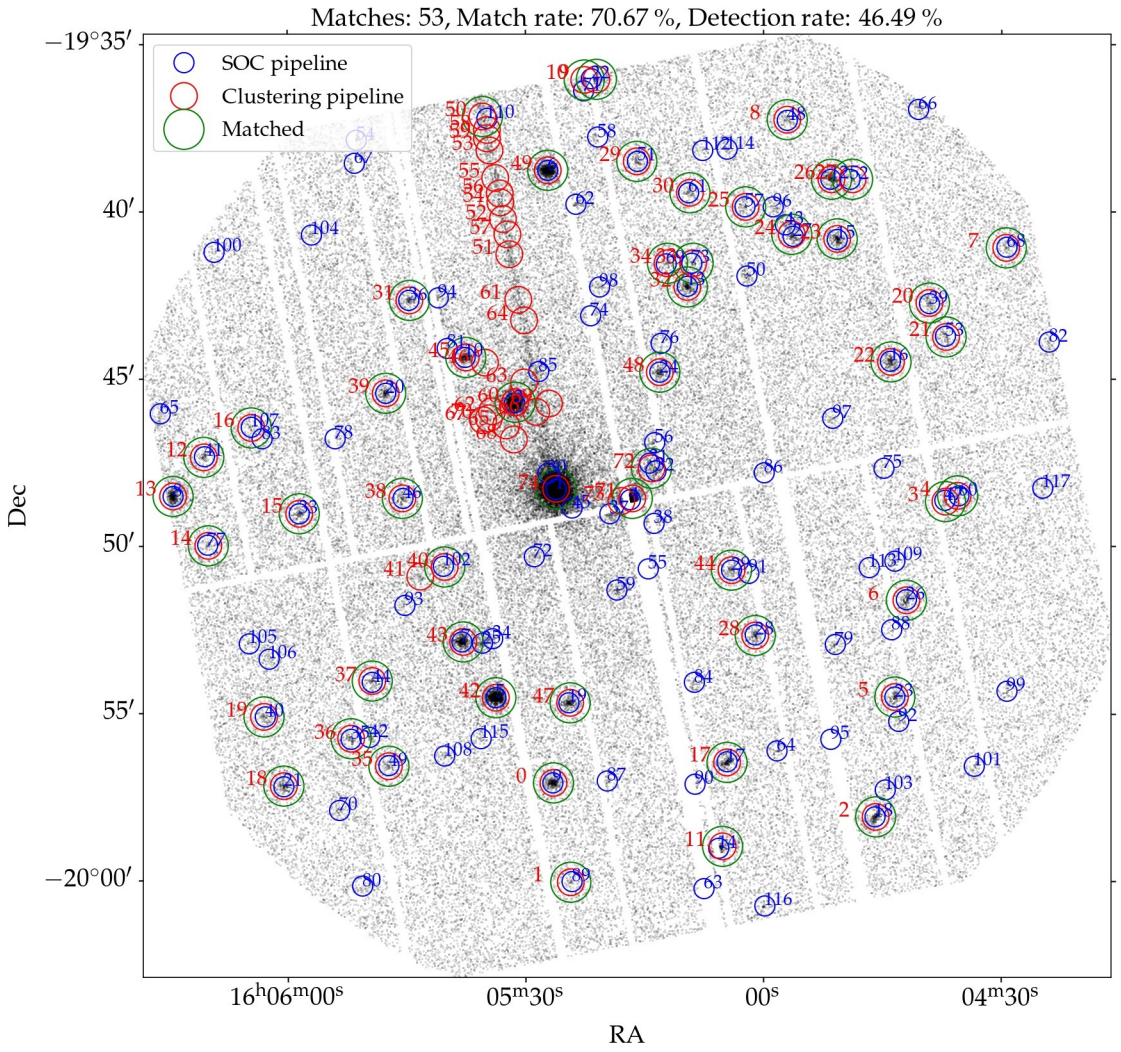

Figure 4.7: HDBSCAN-based thread cross match validation. E.g. with observation ID: 0690200301.

As for the match rate, a low value is caused by the difference in the source detection techniques used in each case. Remember low match rate means that we are detecting presumed sources that were not detected by SOC pipeline. In most of the cases this detections are clearly fake sources.

In the observation given as example (see Fig. 4.5, Fig. 4.6 and Fig. 4.7), the best detection rate is for HDBSCAN-based thread, which achieves to detect more SOC sources than the two others. On the other hand, it has lower match rate, provoked partially by the read out streak mentioned above, that lead us to detect a bunch of undesirable fake sources.

Chapter 5

Conclusions

In this work, we tried to apply density-based clustering algorithms to flaring background filtering and source detection on XMM-Newton EPIC-PN observations. Source detection was approached using three methods: DBSCAN-based, OPTICS-based and HDBSCAN-based. **The results obtained lead us to the following conclusions**:

- **The flaring background filter shows good accuracy when dealing with either long or short flaring intervals**. Nevertheless, sometimes is unable to detect flaring intervals of low count rate, since they will not fulfill the minimum requirements to be considered a cluster by DBSCAN.
- **DBSCAN-based source detection thread shows consistent results in sparse regions populated by point-like sources and low background**. Whereas it is not able to resolve individual sources on overpopulated regions.
- **We have designed OPTICS-based source detection thread as an aim to maintain well results given by DBSCAN in sparse regions and improve its performance on densely populated ones**. This thread gives, as expected, similar results as DBSCAN-based thread when dealing with low background and point-like sources regions. Although, seeing the results given by OPTICS on M31 galaxy, we can conclude that the resolution problem on densely populated regions remains unsolved. In addition, due to its high execution time, it is not efficient for observations containing dense regions, since they are characterized by a large amount of events.
- **The last source detection thread, HDBSCAN-based, has been able to deal with the problems that arises when using DBSCAN-based thread on densely populated regions, as we show with several M31 observations**. It also **reduces**

significantly the execution time compared to OPTICS. Also, **the problem introduced by HDSCAN on sparser regions has been successfully fixed applying a second layer of DBSCAN** to each cluster independently.

It is also important to mention that we tried to develop a transient source detector involving both spatial and temporal coordinates on DBSCAN algorithm. The process is discussed more extensively in the next chapter. This is still under investigation and therefore not included yet in the final pipeline. It must be properly revised and developed, that is why we have included it in the list of future steps (next chapter 6).

It is necessary to argue that we have set the hyperparameter values of the whole pipeline with the aim of making it functional for a wide range of different observations. One could obtain a better result by fitting them for specific observations.

Another drawback of this pipeline, common to each of the three source detection threads, is that we did not found the way of making these methods unaffected by empty space between two consecutive EPIC-PN CCDs. This causes that sources located in the middle of these regions, end up divided in two, since clustering algorithms consider this gaps as a reason to divide a cluster in two.

As for the two significance metrics defined, we also find necessary to warn that they give indicative results on each source, but they should not be seen as a conclusive result of this project. It is just a method for validating the sources detected. They should be understood as a proposal for validating sources detected using density-based clustering methods in a different way than [15] and [23], aiming to deal with some specific problems intrinsic to XMM-Newton observations. In other words, more formal considerations are needed to develop a more rigorous version of these two metrics.

Finally, we observed by cross validating our clustering detected sources with the ones obtained by the traditional method, that we normally detect between 35% and 65% of the sources detected by the XMM-Newton pipeline. However, it should be taken into account, that XMM-Newton pipeline is really sensible when performing source detection. We observed that the sources undetected by our pipeline tend to have a low maximum likelihood in XMM-Newton pipeline. Having said that, we can conclude that we had similar results than the actual XMM-Newton pipeline, despite there are sources it detects that we do not and vice-versa.

Chapter 6

Future steps: transient source detection

This project is the first step of a long road of applying clustering algorithm to XMM-Newton data. The road has started from the basics, being very promising able to clean the observation from flaring background particles and detect sources. But there are some other elements that can be explored on EPIC event files. In this section the focus will be in the detection of transient sources.

Transient sources are those that are not present in the whole period of time, they appear at a certain instant and disappear after a short period of time. That makes them sometimes really difficult to detect.

In this work, a method to detect transient sources has been tried, but with no success yet.

As we can see in Fig. 6.1, the input data for transient detection are those events who still do not belong to any source after searching for bright and faint sources. To detect transients, temporal coordinate should be included in the clustering process. In order to include time on the clustering model, we have to rescale the data. Normally, one would standardize all variables involved in the clustering algorithm so they will be equivalent.

In this work we have used a different strategy, since we have a preliminary idea of what we are looking for. According to astronomical and instrumental criteria, we assume that, to be a transient source, it has to produce around 10 counts in a duration of 5 frames and a detector area of one pixel. This idea of transient source is hypothetical and has been slightly re-tuned along the implementation process.

Having said that, we have transformed spatial coordinates (DETX, DETY) to that predefined resolution scale, which means that a square unit (x, y) in this scale will be equal to a pixel of the camera, and a time unit will be equal to 5 frames. Now we set DBSCAN hyperparameters: we would have a value of *eps* of 1 and *min_samples* of 10. Notice that *eps* equal to 1 defines a neighborhood of 1 pixel and 5 frames, and the value of *min_samples* set to 10 defines the constraint of having 10 counts on this region. That way, we have a more convenient normaliza-

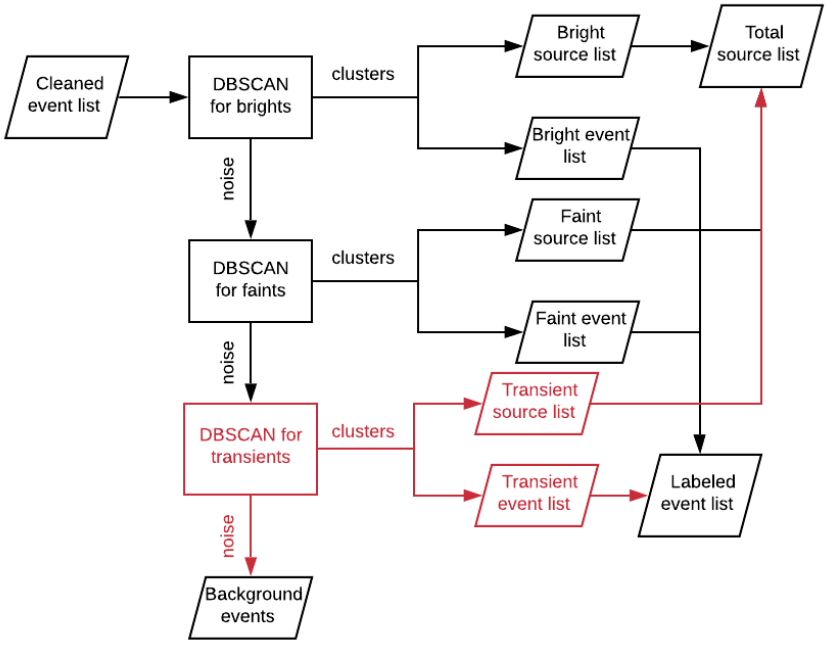

Figure 6.1: Data flow of DBSCAN-based thread with a transient detection cell included.

tion of the data. It is important to notice that the parameters we try to fit in this part of the pipeline are directly the hypothetical shape (counts per area per time) of the transient sources we aim to detect. We have tried to find this type of sources over a wide range of EPIC-PN observations. In fact, the transient detection with DBSCAN is implemented in our clustering pipeline, in DBSCAN-based thread, in the way that is showed in Fig. 6.1.
Unfortunately, this method has not been able to detect any transient yet. A fine tunning of the model is needed for this pipeline to have a functional transient detection method available.

6.1 Other possible future steps

Another natural step forward could be a possible source classification model. This step is less related to this work and is completely independent, since actually XMM-Newton Archive has plenty of spectra and light curves to train and test the models. In this work, apart from obtaining a list of sources, one can get the spectra and the light curve of any of the sources detected. However, it is necessary to warn that, spectra and light curves extracted from this pipeline are far from being a scientific and valid spectrum / light curve, since they have not been generated using the required calibration data. One should better extract spectra and light curves from XMM-Newton archive. There are currently initiatives at the ESAC ML group to classify astronomical objects using deep learning.
Finally, improving the performance of the pipeline could take big advantages. There are two

main important high workload phases in this work: the background estimation map and the OPTICS-based thread. This does not mean the rest of the pipeline is perfectly optimized. The focus in this work was on the results themselves. Focusing on the computational efficiency can yield to a new improved version of the pipeline. We are aware that there are parallelizable tasks inside the pipeline. E.g. on HDBSCAN thread, DBSCAN layer should be parallelized since it executes over each of the HDBSCAN clusters independently. As for the significance metrics, there are density measurements that can be omitted.
Transient sources detection and parallelization of certain phases of the pipeline will be explored. So hopefully a new version of the pipeline will be available soon.

Bibliography

[1] XMM-Newton Science Operations Centre site.

[2] XMM-Newton Science Operations Centre staff. (2020) *XMM-Newton Users Handbook.* XMM-Newton Science Operations Centre. European Space Astronomy Centre (ESAC-ESA).

[3] Guainazzi, M. (2012) *An introduction to XMM-Newton data analysis and the SAS grand-scheme.* XMM-Newton Science Operations Centre. European Space Astronomy Centre (ESAC-ESA).

[4] I. Traulsen, A. D. Schwope, et al. (2019). *The XMM-Newton serendipitous survey.* Leibniz-Institut für Astrophysik Potsdam (AIP), Germany.

[5] Hastie, T. & Tibshirani, R. & Friedman, J. (2001) *The elements of Statistical Learning.* 2nd Edition. Springer Series on Statistics.

[6] Gareth, J. & Witten, D. & Hastie, T. & Tibshirani, R. (2013) *An introduction to Statistical Learning.* Springer Texts in Statistics.

[7] Kumar, V. (2014) *Data clustering: algorithms and applications.* University of Minnesota. Department of Computer Science and Engineering. Minneapolis, Minnesota, USA.

[8] MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations.* Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press.

[9] Jin X., Han J. (2011) Expectation Maximization Clustering. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA.

[10] Ester, M. & Kriegel, H. P. & Sander, J. & Xu, X. (1996) *A density-based algorithm for discovering clusters.* Institute of Computer Science. University of Munich.

[11] Vojtekova, A. & Lieu, M. & Valtchanov, I. et al. (2020) *Learning to Denoise Astronomical Images with U-nets*. European Space Astronomy Centre (ESAC-ESA).

[12] Wilkinson, S. & Merín, B. & Riviere-Marichalar, P. (2018) *New member candidates of Upper Scorpius from Gaia DR1*. European Space Astronomy Centre (ESAC-ESA) & Instituto de Física Fundamental (CSIC).

[13] Cánovas, H. & Cantero, C. et al. (2019) *Census of ρ Oph candidate members from Gaia Data Release 2* European Space Astronomy Centre (ESAC-ESA).

[14] Shou-kun, X. & Chao, W. et al. (2019) *DBSCAN Clustering Algorithm for the Detection of Nearby Open Clusters Based on Gaia-DR2*. School of Information Science and Engineering, Changzhou University, Changzhou.

[15] Tramacere, A. (2014) *γ-ray DBSCAN: a clustering algorithm applied to Fermi-LAT γ-ray data.* University of Geneva, Italy.

[16] Ankerst, M. & Breunig, M.M. et al. (1999) *OPTICS: Ordering Points To Identify the Clustering Structure*. Institute for Computer Science, University of Munich, Germany. *HDBSCAN: Hierarchical density based clustering*. Journal of Open Source Software, The Open Journal, volume 2, number 11.

[17] Rhys, H.I. (2020) *Machine Learning with R*. Manning publications.

[18] Campello, R. & Moulavi, D. & Sander, J. (2017) *Density-Based Clustering Based on Hierarchical Density Estimate*. Dept. of Computing Science, University of Alberta, Edmonton, AB, Canada.

[19] L. McInnes & J. Healy & S. Astel (2017) *HDBSCAN: Hierarchical density based clustering*. Journal of Open Source Software, The Open Journal, volume 2, number 11.

[20] Versloot, C. (2020) *Performing OPTICS clustering with Python and Scikit-learn*. MachineCurve.

[21] Castro-Ginard, A. et al. (2018) *A new method for unveiling open clusters in Gaia. New nearby open clusters confirmed by DR2* . Universitat de Barcelona, Spain.

[22] A. Kochoska et al. (2018) *Gaia eclipsing binary and multiple systems. A study of detectability and classification of eclipsing binaries with Gaia*. University of Ljubljana, Deptartment of Physics, Slovenia.

[23] Armstrong, Thomas et al. (2015) *The detection of Fermi AGN above 100 GeV using clustering analysis*. Department of Physics and Centre for Advanced Instrumentation, Durham University, UK.