Rejection method for supervised classification of plant species from passive optical remote sensing data

Léa Camusat[1], Eva Etheve[1], Julie Gonzalez[1] , Lila Roig[1]
[1] National Institute of Applied Science - Toulouse, Department of Applied Mathematics
May 23, 2022

Abstract.

The identification of plant species is essential for biodiversity analysis and the monitoring of environmental degradation. Many studies have highlighted the interest of optical remote sensing to map plant species on a given satellite image. Each pixel of the satellite image can be well paired with a plant since each plant has its own spectrum. However, the performance of mapping algorithms is affected by the presence of outliers: some plant species are a minority and should not be labelled. Rejecting these outliers would reduce pixel misclassification. Five classification algorithms (Regularized Logistic Regression with l1 or l2 regularization, Linear Support Vector Machine, Radial Basis Function Support Vector Machine and Random Forest) applied on an image give a vector of probabilities that a given pixel belongs to each tree species. Unfortunately no efficient post-rejection algorithm has been developed yet in the field of hyperspectral and multispectral images. Our aim is to find relevant rejection methods which can be applied post classification on probability vectors. Boundaries between species and minor species must be rejected. We compare different clustering methods applied on probability vectors: K-means, SVM (Support Vector Machine), DBSCAN (Density-Based Spatial Clustering) and GMM (Gaussian Mixture Models). We improve our results taking the context into account. We define qualitative and quantitative ways to evaluate the rejection performance. K-means and SVM have significant visual results as they reject some specific point of interest on the image. K-means algorithm applied on the dataset Random Forest gives the best result with a True Rejection Rate of 57%. The False Rejection Rate obtained with K-means and SVM is quite low on every dataset, around 0.5-1%, which means that most of the time we do not reject wrongly. Our classification assists the ONERA laboratory (National Office for Aerospace Studies and Research) in their research. For example, to quantify contaminating anthropogenic impacts in soils, the vegetation condition is analysed and it is necessary to know the species corresponding to every pixels. Further research can use our results to compare the efficiency of a rejection option performed during classification with a rejection option performed post-classification.

keywords: remote sensing, hyperspectral, multispectral, reject option, post classification, pdf clustering

1. Introduction. As a result of the significant development of anthropic activities, we become witnesses of a worrying degradation of our environment. The impact of our activities on the planet is becoming increasingly serious. Monitoring this environmental degradation is therefore necessary in order to protect our ecosystems.

Earth observation satellites are proven to be an effective tool to study the variation of our planet. For more than forty years, the numerous sensors of orbiting satellites have offered a wide range of spatial, spectral and temporal resolutions. The processing of this information has many applications, for example, forests can be mapped to monitor their health, determine their structure and estimate their biomass (Cf. [Lourenço, 2021]). Wetlands can also be mapped and classified to conserve and use the resources intelligently and evaluate the impacts of climate change and human activities on the coastline. These studies use remote sensing, which is a set of techniques used to determine remotely the properties of natural or artificial objects according to the radiation they emit or reflect. Remote sensing involves three steps: capture and recording of the radiation received, processing of the resulting data and finally analysis of these data. Here, a particular interest is given to optical remote sensing in the reflective spectral range (0.4-2.5 μm) adapted to the mapping of plant species and/or habitats.

To map plant species, supervised classification methods based mostly on machine learning techniques are used. However, these methods usually require a large training set of data representative of the species to be mapped, which is a problem in the field of teledetection. In the domain of teledetection, images are large and are not labelled most of the time because it can be difficult to obtain field data on all the images (for example if the areas are difficult to access). Thus, some of the species are forgotten or not enough present on the images. We can also mention the problem of mixed pixels which is inherent in remote sensing data. A pixel has a defined real size on the ground, which means that we have no finer information than this pixel and that it can be composed of several species. The efficiency of a classification algorithm depends on its ability to "trade-off the risk-coverage (RC) for higher accuracy" [El-Yaniv and Wiener, 2010]. A good model is supposed to assign many features to a class and to avoid misclassification at the same time. It requires

the implementation of a reject option to build a rejection class grouping pixels (representing plant species) likely to be misclassified.

A reject option in classification was first considered by [Chow, 1957], who developed a reject option based on the Bayes rule, which consists in not classifying a data point if the maximum post classification component of the probability vector is lower than a given threshold. Unfortunately, very few rejection methods have been developed in the field of hyperspectral and multispectral images. Moreover, most of the rejection methods are being applied during the classification, while we have to implement a post-classification method. For example, [Condessa et al., 2015] presents two classification methods with rejection in the field of hyperspectral images. One method consists in jointly computing context and rejection, and the other one computes them sequentially. An extra class models the probability that the classifier fails to sort a pixel into a class. These two methods applied on the dataset *Avriris Indian Pines* respectively provide a non-rejected accuracy of 80% and 76% and a classification quality of 78% and 74%.

Our objective is to find relevant rejection methods which can be applied post classification on probability vectors. We consider two categories of rejection options: novelty rejection, which involves rejecting pixels that are from minority species, and ambiguity rejection, which refers to the rejection of boundaries between several species.

We use clustering techniques to implement a post-classification rejection method and classify probability vectors. One of the main advantages of clustering is that it is an unsupervised technique, that is to say we do not need the labels of the data points. Pixels are grouped by similarity, according to their probability vectors, in which each component is the probability to belong to the different species.

Clustering of probability vectors has already been investigated in several works. For example, [García-García et al., 2012] aimed to classify a set of vectors. Vectors were considered as independent and identically distributed (i.i.d) samples and their probability density function (PDF) was estimated using Gaussian Mixture Models (GMM). Then, one has to define an adequate distance measure, such as the Kullback-Leibler divergence so that clustering algorithms can be applied on probability vectors. They tested their clustering algorithm on the *MoG dataset* and tunned their parameters, the best result they obtained gave a clustering error of about 15%.

We can also mention [Vo-Van and Pham-Gia, 2010], who estimated PDFs nonparametrically from the data and grouped them into homogeneous clusters. Several clustering algorithms were then applied, such as K-means, hierarchical and non-hierarchical methods, clustering with a bound on cluster widths and intercluster distances.

We perform different supervised and unsupervised methods applied on probability vectors: K-means, SVM (Support Vector Machine), DBSCAN (Density-Based Spatial Clustering) and GMM (Gaussian Mixture Models).

K-means is an unsupervised partitioning-based algorithm. The main advantage of an unsupervised algorithm is that one does not need to know beforehand the labels of the data. Given a number K of clusters, K-means minimizes the distance between the data points and the cluster centers. This method cannot detect outliers; however, it can predict for new data. Rejection methods based on K-means clustering have been investigated in other works. For example [Ahmed and Mahmood, 2013] used the fact that K-means computes clusters centers, thus one can predict if a point is an outlier or not by setting up a maximum distance to the cluster center. Within this distance, the point are considered close enough to be associated to the cluster. They presented the ODC algorithm (Outlier Detection and Clustering) which is based on K-means algorithm. In this algorithm a data point that is at p times the average distance away from the cluster's centroid is considered to be an outlier. To measure the efficiency of the ODC algorithm, they computed the ratio of within-class variance against the inter-class variance. Detecting and removing outliers should reduce the value of this ratio. For example, in Glass data, the value of the ratio was 0.92 and the value dropped down to 0.67 with the detection of the outliers.

DBSCAN is an unsupervised density-based algorithm. It cannot make predictions on new data, but it detects outliers. Thus, it can be directly applied to our problem [Martin Ester, 1996].

SVM algorithms can be considered as efficient algorithms to carry out a reject option in binary classification problems. They consist in finding a p-dimensional hyperplane which separates the data into classes maximizing the boundaries on both sides of the hyperplane(Vapnik rule). The rejection based on SVM models has been implemented by [Djeffal, 2012] and [Grandvalet et al., 2008].

GMM is an unsupervised and probabilistic soft clustering method. Since many data sets can be modeled by Gaussian Distribution, the GMM algorithm assume that the clusters come from different gaussian distributions and tries to model the dataset as a mixture of several gaussian distributions. Like K-means, this algorithm needs the number of clusters and does not require the labels of the data. However, rather than giving hard assignments into clusters as K-means, GMM gives soft assignments. This means that for each data point, GMM gives a vector of probability that the point belongs to each class. However, the work of

[Deng et al., 2019] proposes to use a modified version of GMM to implement a post-classification rejection method to select representative paintings of an artist. Using convolutional neural network, this method first extract a content-style feature vector for each painting. Then it performs Robust Continuous Clustering (RCC) and uses an algorithm similar to GMM to reject non-representative paintings. The K-means and Agglomerative clustering methods are also used instead of RCC for comparison. The algorithm is tested on various paint databases and has a classification accuracy of about 91.42%.

We use Python programming language and QGIS software that facilitates the visualization of species classification in space. We also carry out rejection methods with K-means, DBSCAN and SVM algorithms which are in Python's Sklearn library and implement our own GMM algorithm. Finally, we evaluate their performance and compare the results.

2. Description of the data. Within the framework of this research, all the datasets were provided by the ONERA research center. We work on an airborne hyperspectral image of a given area, which size is 2.45 km^2 represented by 2516 $\times$ 2473 pixels and a smaller image of size 624 $\times$ 361 pixels extracted from the original image. This smaller image allowed us to test our algorithms more quickly on a smaller dataset, the size being sometimes a problem to store the data, as is shown with **DBSCAN** algorithm. The image was preprocessed and some pixels that do not correspond to vegetation (buildings, paths, eletricity wires) or correspond to shaded pixels were discarded. These pixels were not considered as training data and we did not apply any clustering to them. We note the presence of 40% shadow pixels on the small image and 78% shadow pixels on the complete image.

Initially, the ONERA research center implemented five algorithms: Regularized Logistic Regression with l1 regularization (Lasso regularization) - (**RLR l1**), Regularized Logistic Regression with l2 regularization (Ridge regularization) (**RLR l2**), Linear Support Vector Machine (**SVML**), Radial Basis Function Support Vector Machine(**SVMRBF**) and Random Forest (**RF**). These algorithms were applied on the remote sensing data captured in the region of which we have the satellite image. For each pixel, the algorithms gave a vector of probabilities that the pixel belongs to each tree specie, as well as the class to which the pixel would belong according to this vector. Thus, they obtained two matrices containing the result of this first classification. The first matrix is a 3-dimensional matrix, containing the probability vectors of each pixel. The first two dimensions of the matrix in the plane corresponding to the dimensions of the image (width w and length l). The third dimension corresponds to the probability vector of each pixel and its length is equal to the number of tree species n. The ten following species are being considered:

- *Platanus*
- *Salix*
- *Populus*
- *Quercus*
- *Alnus*
- *Robinia*
- Grass mixtures
- Shrubs mixtures
- *Reynoiutria*
- Corn crops

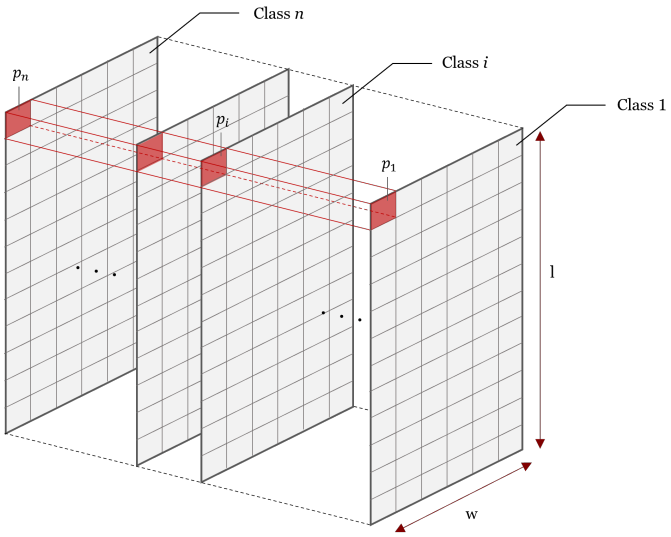


Figure 2.1: 3D representation of the input data

A representation of the 3D matrix is provided in Figure 2.1. Each square in the grid corresponds to a

pixel of the image and the probability vector of size n associated to a same pixel is symbolized by the set of squares colored in red. We denote p_i the probability that a given pixel belongs to the class i.
The second matrix is a 2-dimensional matrix of width w and length l. For each pixel, it contains the class predicted by the algorithm according to the obtained vector of probabilities.

3. Methods. In this section, we implement several clustering and rejection algorithms on our dataset: K-means, SVM, DBSCAN and GMM.

3.1. Training set and context.

3.1.1. Classification rule for the algorithms K-means and SVM. **K-means** and **SVM** algorithms are two of the most used clustering algorithms. They can perform an accurate clustering; however, they cannot detect outliers. Thus, we have to build a training set and a set to be predicted so that we can perform outlier detection based on the clustering of these two algorithms.

We distinguish three categories of pixels:

- The pixels that have been well classified by the algorithms **RLR l1**, **RLR l2**, **SVML**, **SVMRBF** and **RF**, and that form a training set for the clustering algorithms **K-means** and **SVM**
- The pixels that the algorithms have been unable to classify and that we have to predict on the trained models, to allocate them to a cluster or to reject them if they are outliers.
- The remaining pixels are shaded and non-vegetated pixels, they are not taken into account in the training set.

To determine which pixels have been well classified by the algorithms **RLR l1**, **RLR l2**, **SVML**, **SVMRBF** and **RF**, we implement three classification rules based on different hypotheses.

First Hypothesis: the 0.5 rule
The first assumption we can make is that a pixel has been classified correctly almost surely when the highest component of its probability vector is greater than **0.5**. There are **10 classes**, which is quite large, so this assumption seems reasonable.

Second Hypothesis: the 0.5 and difference rule
This rule is based on the first one: we retain pixels whose highest predicted probability is greater than 0.5. However, we include a second restriction: **the difference between the two highest probabilities** in the vector of a pixel should be higher than a certain threshold. For example, if the probability that a pixel belongs to the *Platanus* class is 0.51 and that its probability to belong to the *Salix* class is 0.49 we cannot be sure that there is no confusion between the two classes. Thus, the pixel will not be part of the training set and will have to be predicted by the clustering algorithms. This rule may also be relevant to reject the mixed pixels.

Third Hypothesis: the cross rule
Initially, the classification of trees was made with five algorithms giving the 3D matrix of probability vectors. We only retain the pixels which have been classified correctly by all algorithms and in the same class to build the training set. It corresponds to the third hypothesis. We can thus cross results between each algorithm and expect a more accurate classification.

3.1.2. Classification with context. The context should be considered when classifying and rejecting pixels. For example, pixels that are surrounded by rejected pixels are likely to be rejected as well. Furthermore, pixels that have been rejected but that are surrounded by pixels of the same class are more likely to be classified in that class. Taking context into account allows smoother classification and avoids misclassification for punctual pixels. Context can be processed post classification to improve the rejection accuracy. Two parameters must be chosen: the size of the sub-image which surrounds each pixel which was not classified, and a threshold which quantifies the proportion of pixels from the majority class of the sub-image. If this threshold is lower than the proportion of the majority class, we classify the pixel as belonging to the majority class. A pseudo code for this method is presented in the appendix A.1.

3.2. Rejection algorithms.

3.2.1. K-means algorithm. K-means algorithm is an unsupervised partitioning-based algorithm.
Given a number K of clusters, K-means minimizes the distance between the data points and the cluster centers. The problem can be formulated as follows:

$$\min_{A,c} \sum_{i=1}^{M} \sum_{k=1}^{K} a_{i,k} \text{ s.t. } \begin{cases} \sum_{k=1}^{K} a_{i,k} = 1 \forall i \in 1, ..., M \\ a_{i,k} \in \{0, 1\} \end{cases} \tag{3.1}$$

This problem is solved by the **Expectation Maximization (EM) algorithm** (Cf. [Besse, 2022], as described in the pseudo code in the appendix A.2).

We use the already implemented python function **K-Means** of the library **sklearn.cluster**. The metric used by this algorithm is the euclidian distance, which might be criticised given that our dataset is composed of probability vectors. Nevertheless, the advantage of K-Means algorithm is that it is a fast algorithm which can predict the cluster of new data.

We have ten different species so we want to build ten clusters with K-means. First, we train the K-Means algorithm on the training data selected by one of the above classification rules. As the K-Means algorithm does not detect outliers and predict the pixels that have been unclassified, we then implement a post clustering rejection method using the **radius** of each cluster.

We compute the radius of each cluster, corresponding to the maximum distance between the cluster's centroid and the training data points belonging to this cluster. The idea is the following: if a unclassified pixel p falls within a distance of the radius times a given threshold ($T \leq 1$) from a cluster center, we can consider that the pixel belongs to this cluster. Otherwise, if the pixel p is at the border between two clusters or if it does not fall in any cluster, we reject it. The threshold T has to be calibrated to find the value that will best fit our data. If $T = 1$, we consider the entire radius thus we reject less. As soon as T decreases, the radius' length decreases, thus the specificity of the result increases and we reject more often as we consider that points that are at the edge of the cluster are not representative of the cluster. The pseudocode for this method is given in appendix A.3.

To calculate the radius of a cluster or to compare the distance between a data point and the center of a cluster, we need to define a measure. Because the distance we want to measure is between two probability vectors (which sum to 1), the Euclidean measure does not appear appropriate. We decide to use the Wasserstein metric which can be taken as a reasonable distance on spaces of random variables or probability distributions [Givens and Shortt, 1984].
Let (M, d) be a metric space. For each p with $1 \leq p < \infty$, let $\mathcal{P}_p(M)$ denote the collection of all probability measures (i.e. laws) μ on (the Borel sets of) M with finite p^{th} moment, that is, there are some x_0 in M such that:

$$\int_M d(x, x_0)^p \mathrm{d}\mu(x) < \infty \tag{3.2}$$

where $d(x, x_0)^p$ is for example the p-norm $\|x - x_0\|_p$. The p^{th} Wasserstein distance between two probability measures μ and ν in $\mathcal{P}_p(M)$ is defined as

$$W_p(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} d(x, y)^p \mathrm{d}\gamma(x, y) \right)^{1/p} \tag{3.3}$$

where $\Gamma(\mu, \nu)$ is the set of all couplings μ and ν.

3.2.2. SVM. We assume that the data can be linearly separated. Pixels which are rejected are those which are in the gap between the two boundaries. This problem can be extended to a multi-classification task. There are several methods such as OVR (One vs the Rest) and OVO (One vs One). OVO consists in calculating hyperplanes two by two. In our classification problem, we have optimized 45 hyperplanes (K(K-1)) with K, the number of classes. Due to the asymmetry of the dataset used to optimise each hyperplane, OVR is considered to increase the bias, as each hyperplane is built to separate one class and the nine other classes [Djeffal, 2012].

We implement a calibration method, which is based on the *decision_function* of the method *svm* in Sklearn. The decision function is computed according to the votes and can be seen as a way to measure the confidence on the prediction. We obtain a matrix containing each probability of the respective label for each pixel ($pixel\_to\_predict \times 10$ classes matrix), which can be considered as a way to measure the confidence in the prediction.

We implement the rejection method shown in Algorithm A.4.

The specificity of this algorithm is that we decide to set a rejection rate instead of determining a threshold. We predict only pixels which are not in the training set based on the rules of preclassification. All pixels that

have a relative difference of decision function lower than the median (or the 3rd quartile) of these relative differences must be rejected, as this indicates that the classifier did not decide between two classes, and that we are in an undecision area (Figure 3.1).

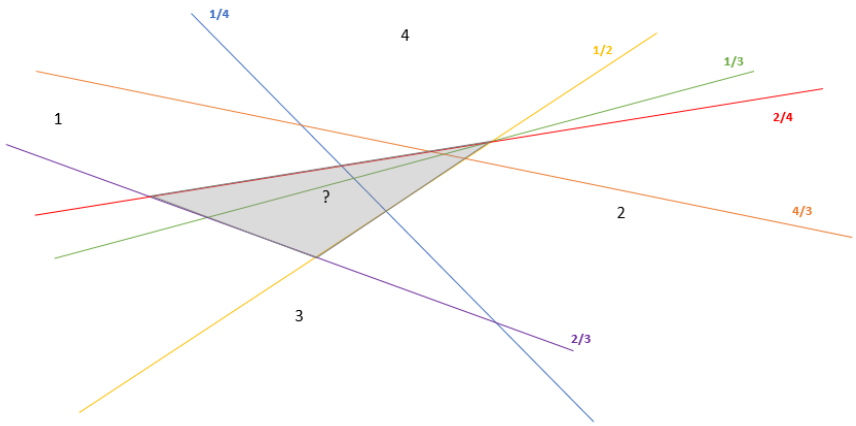


Figure 3.1: Undecision area in OVO

3.2.3. DBSCAN algorithm. DBSCAN, or Density-Based Spatial Clustering of Applications with Noise, is an unsupervised machine learning algorithm. The main asset of using an unsupervised algorithm on our dataset is that we do not need to know in advance the labels. DBSCAN is well suited for problems which requires minimal domain knowledge to determine the input parameters. For example, unlike K-means, DBSCAN does not need the number of clusters beforehand. We use an already implemented function from library **sklearn.cluster** called **DBSCAN**. As for many machine learning algorithms, the behavior of the model is dictated by several parameters. For this algorithm, the three most important parameters are *eps*, *min_samples*, *metric*. The choice of these parameters is explained below.

Definition and Choice of *eps*: Two points are considered as neighbors if the distance between them does not exceed the threshold epsilon. This parameter is the most important one since it provides the condition for two points to be considered in the same cluster. We can find a suitable value for the `eps` parameter by calculating the distance between P and its closest neighbour P', for each point P. After sorting these values, we obtain a plot that shows the change of the distance between two close points. Then, we look where the slope soars and select the corresponding epsilon. In a more quantitative way, this variation corresponds to the point where the curve has a change in slope of at least 1%. To compute these values, we used the function $NearestNeighbors$ from the library sklearn.neighbors. An illustration of this procedure is given in Figure 3.2.

Definition and Choice of *min_sample*: This parameter gives the minimum number of neighbors a given point should have to be considered as a core point. The point itself is taken into account in the setting of *min_sample*. By default, the parameter is set to 2 but by increasing its value we can make sure that we find the number of clusters in the database. The more *min_sample* increases the more we constrain the algorithm since a point P is considered to be core point if it has at least *min_sample* number of points in its neighborhood. This parameter varies depending on the sample.

Choice of *metric*: As DBSCAN computes distances between points in the database, it is important to decide how to evaluate these distances. The results can vary significantly depending on the metric used. We choose different metrics allowed by the function $DBSCAN$ and compare the number of rejected pixels they give.

Once these parameters are chosen, the DBSCAN algorithm provides the clusters it has found while detecting the anomalies, in our case the pixels that are in the margin. The advantage here is that DBSCAN directly gives the pixels to be rejected by giving the label *-1*. To avoid a kernel crash because of the size of the dataset, it can be better to apply this function to smaller parts of the image. A pseudo-code of the

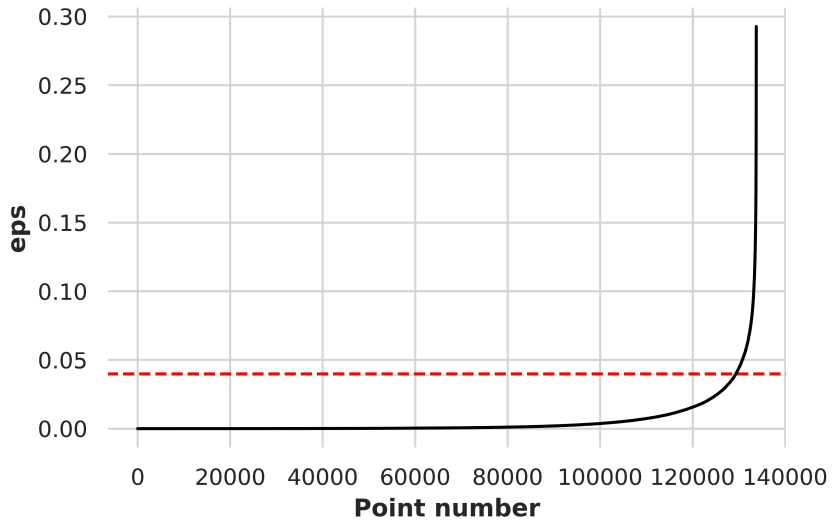


Figure 3.2: Points sorted by distance to the second nearest neighbor, best epsilon value found for 0.04

algorithm is provided in appendix A.5.

3.2.4. Reject option algorithm using GMM. In this section, we focus on a rejection method using multivariate Gaussian distribution and comparable to a modified GMM algorithm. The reject option occurs after a classic clustering algorithm such as K-means or GMM. To do so, we rely on the paper [Deng et al., 2019] which implements a rejection method in the case of the classification of paintings.

Adopting the notations of the paper, we denote $U = \{u_i | 1 \leq i \leq n\}$ the set of probability vectors where $u_i \in \mathbb{R}^D$ is the vector containing the probabilities that pixel i belongs to each class ($D = 10$ classes) and n is the total number of pixels. We assume that the elements in cluster U_M are mutually independent and obey the Gaussian distribution.
A clustering algorithm is then applied to all data that are not shadow pixels. In opposition to the rejection method presented in section 3.2.1, the clustering algorithm is also applied to the unclassified pixels according to the classification rules presented in section 3.1.1.
As the clusters are assumed to be Gaussian, the data is renormalized. Moreover, it seems appropriate to apply the GMM clustering algorithm (using the already implemented Python function) which also assumes that the clusters are Gaussian. In a second phase, we apply the K-means clustering algorithm for comparison.

For each cluster M, we formulate the class-conditional-probability density of probability vector u_i as

$$\mathbb{P}(u_i|M) = \frac{1}{\sqrt{(2\pi)^D det(\Sigma_M)}} e^{-\frac{1}{2}(u_i-\mu_M)^T \Sigma_M^{-1}(u_i-\mu_M)} \tag{3.4}$$

where μ_M and Σ_M are the mean and the covariance matrix of cluster M. Thus, we can obtain the probability $\mathbb{P}(M|u_i)$ defined by:

$$\mathbb{P}(M|u_i) = \frac{\mathbb{P}(u_i|M)\mathbb{P}(M)}{\mathbb{P}(u_i)} \tag{3.5}$$

which indicates the probability of u_i belonging to cluster M,
where

$$\mathbb{P}(u_i) = \sum_{m \in clusters} \mathbb{P}(u_i|m)\mathbb{P}(m) \tag{3.6}$$

$$\text{and} \quad \mathbb{P}(M) = \frac{number\ of\ pixels\ in\ cluster\ M}{total\ number\ of\ pixels} \tag{3.7}$$

Therefore, a pixel which is represented as u_i classified to cluster M will be rejected if $\mathbb{P}(M|u_i)$ is not higher than a threshold T, which means pixel u_i is more likely to be an outlier of cluster M.

Once this rejection is completed, we cluster the data again, modifying the probability distribution of each cluster. The cluster-rejection process is repeated until the stop criterion is met. The pseudocode for this method is shown in the appendix A.6. We use the python function **multivariate_normal.pdf** from **scipy.stats** to compute $\mathbb{P}(u_i|M)$.

3.3. Methods to evaluate the rejection performance.

3.3.1. Visual assessment of performance. Some areas in the image must be rejected by our algorithms. They correspond to specific areas such as the island, the fallow land, remaining non-vegetated or shaded pixels as well as mixed pixels, electricity lines, or paths. These areas are supposed to be rejected as their spectrum is different than the species we want to classify, as we can see on Figure 3.3.

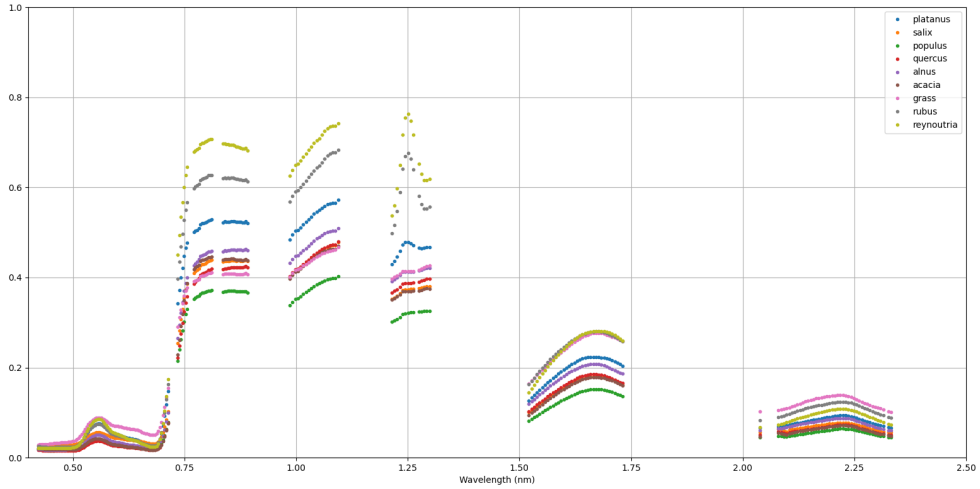


Figure 3.3: Spectrum of the species

We visually assess the performance of our algorithm in QGIS. On Figure 3.4 the elements we should reject are circled in red.

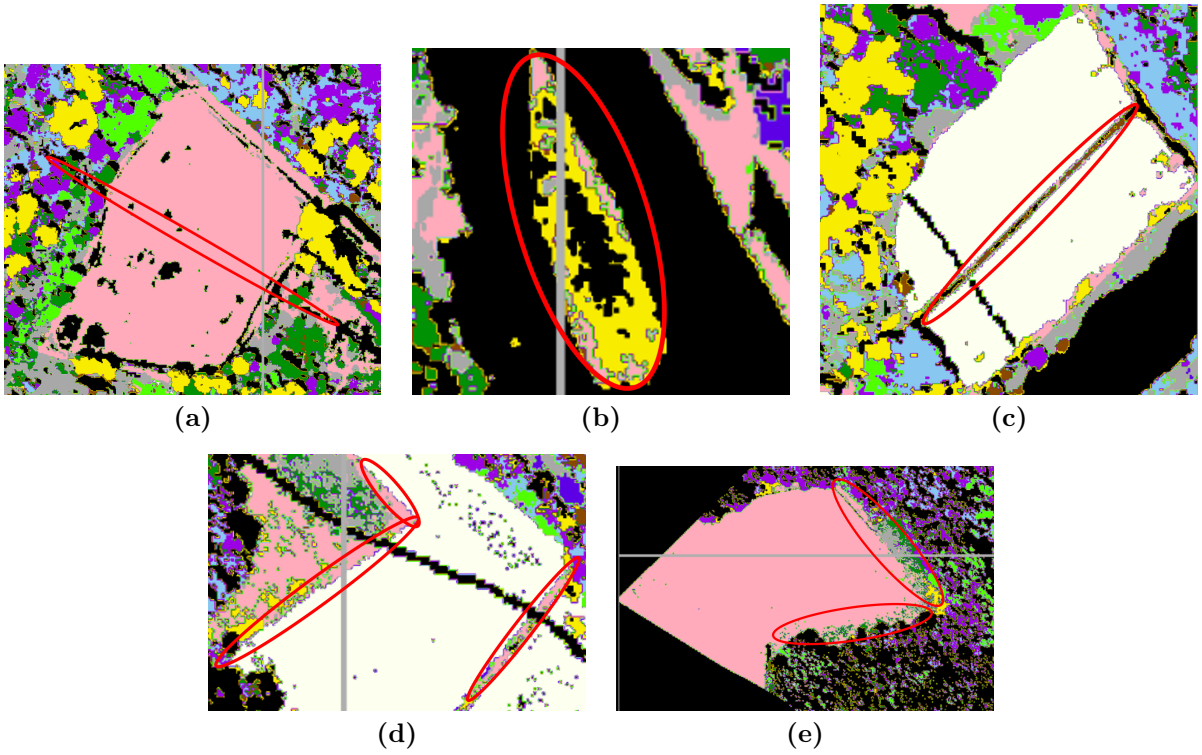


Figure 3.4: **(a)** Electricity line **(b)** Island **(c)** Dirt road crossing a field **(d)** Borders between two species **(e)** Fallow land

3.3.2. Field data and performance. Biodiversity data are collected in the field in 391 areas by specialists. These areas are listed in polygon masks and can be visualized in QGIS in the Figure 3.5 and extracted using rasterio in the Figure 3.6. Each polygon is associated to a plant species. We use these labelled data to evaluate the performance of our rejection methods. Specialists identify 17 plant species. In our algorithms, only ten types of plants must be classified, so the seven remaining classes should be rejected.

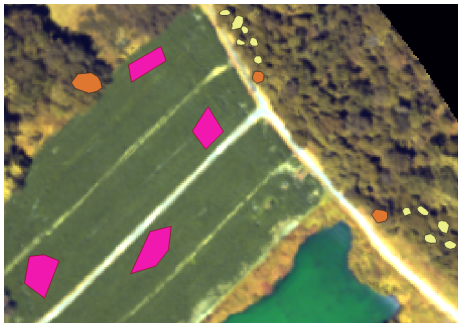


Figure 3.5: Visualisation of masks on QGIS

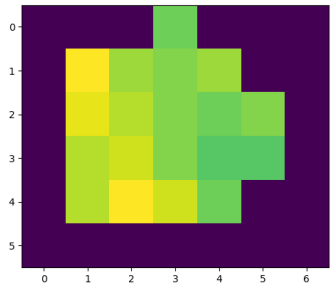


Figure 3.6: Extraction of a polygon using rasterio on Python

3.3.3. Predict rejection of a known class. To evaluate the rejection performance of our algorithms we build a new dataset, in which we delete one tree species and we perform the rejection method on this new dataset. If the rejection method is efficient we should observe that the pixels that were previously classified in the considered tree species should be rejected. We use two datasets to carry out these tests:

- two datasets built with the algorithms RLR and SVM in which the *Platanus* tree species has been deleted
- two datasets built with the algorithms RLR and SVM in which the *Reynoiutria* tree species has been deleted

These tree species were chosen because their spectra are those that are the furthest from the other species. We take the example of the dataset without the Platanus specie. First, we perform the rejection method

on the new dataset as usual, except that we now have nine classes. Then, in the former dataset, in which *Platanus* was not deleted, we find the coordinates of Platanus pixels that were part of the former training set. The reason is that, those pixels were selected by the **0.5 rule** or the **0.5 and difference rule** and were considered well classified. Next, we verify whether or not at these coordinates, the rejection method has rejected the pixels. We compute the **True Rejection Rate**:

$$TRR = \frac{\#(\text{rejected pixels} \cap \text{platanus pixels})}{\#\text{platanus pixels}} \tag{3.8}$$

3.3.4. Performance evaluation criteria. We aim to have an overview of the different methods used to quantify the quality of a rejection method. There is no method of assessment commonly used to assess the performance of a classification method with a reject option. We also describe the main indicators and recommendations about the choices of indicators which ensure the validity of the rejection method used. A good indicator should take into account the accuracy of the rejection and the ability of the model not to reject too much.

According to [Condessa et al., 2017], a good performance measure must follow these three properties:
1) to be a function of rejection rate
2) enable a comparison of rejection methods using the same or different rejection rates.
3) to be maximum (and respectively minimum) for a rejection algorithm with the highest (and respectively lowest) reliability.

Indicators are sorted into three types [Condessa et al., 2017]: measure of the accuracy of classification of non rejected samples, measure which deals with the quality of the rejection and measures of global efficiency in terms of accuracy and rejection. The most common indicators are also presented by [Guichard et al., 2010]. The following indicators are adapted to our problem to measure of quality of rejection :

TRR(true rejection rate) $= \frac{\text{true rejected}}{\text{items to be rejected}}$

FRR(false rejection rate) $= \frac{\text{false rejected}}{\text{items not to be rejected}}$

NRA(non rejected accuracy)$= \frac{\text{well-classified} \cap \text{non-rejected items}}{\text{items not to be rejected}}$

4. Results.

4.1. Results of K-means clustering.

4.1.1. Performance of K-means clustering on the training data. First, we evaluate the performance of K-means on the training data. As we consider this data as being labelled, we can compute the purity metric (equation 4.1). This external metric maps the clusters given by K-means to the real labels. First, a confusion table is built. The table has two entries, on the columns there are the real labels (i.e., the plant species) and on the rows the clusters given by K-means. For each tree species, the table counts the number of pixels that have been classified in the different K-means clusters. Then, we map each species with the cluster in which the maximum number of pixels has been allocated. For example, the confusion Table 4.4 was obtained using the dataset **RLR l1**. The clustering performs well since each species corresponds to a unique cluster.

We evaluate the performance of K-means to allocate each species to a unique cluster, that is to say, for each column T^j of the confusion table T we compute the following ratio:

$$P_j = \frac{max(T^j)}{\sum_{i=1}^{n} T_i^j}, \text{ where } n \text{ is the number of clusters} \tag{4.1}$$

Finally, we compute the purity metric on the whole result:

$$purity = \frac{1}{N} \sum_{j=1}^{n} max(T^j), \text{ where } N \text{ is the total number of pixels in the training set} \tag{4.2}$$

The purity coefficient belongs to $[0, 1]$, the closer it is to 1, the more accurate the clustering.

We determine the optimal rule for each dataset to build its training set. We vary the value of the threshold that we use to build the training set with the **rule05diff** and we compute the **purity metric**. We recall that if the difference between the two highest probabilities in the vector of a pixel is lower than the given threshold, then the pixel will not be part of the training set because there might be a confusion between two classes. So, the higher the threshold is, the better the purity is but also the smaller the obtained training set is. Thus, it is important to optimize this threshold. Ideally, we want K-means clustering to map each species to a unique cluster, which corresponds to a purity metric of 1. However, we also wish for the

Table 4.1: Confusion Table of K-means clustering on the dataset **RLR L1**

	Real species									
clusters	1	2	3	4	7	9	13	14	15	16
0	17	0	0	0	0	0	336	181710	0	0
1	0	0	0	0	0	0	0	0	0	244589
2	17	35478	30	0	5	22	135	35	0	1
3	24	0	185082	0	0	3	50	6	3	27
4	82	0	36	103861	4	20	0	16	0	1
5	89	0	99	0	143055	7	17	134	3	14
6	2	0	0	0	0	68662	0	1	0	12
7	6	0	0	0	0	0	218359	0	0	37
8	105248	0	0	0	0	0	0	0	0	28
9	43	0	0	0	0	0	1	7	26668	2

threshold to be as small as possible, so that the training set contains as many correctly classified pixels as possible. So, the rule is considered optimal when the chosen threshold gives a purity equal to 1 and is as small as possible.

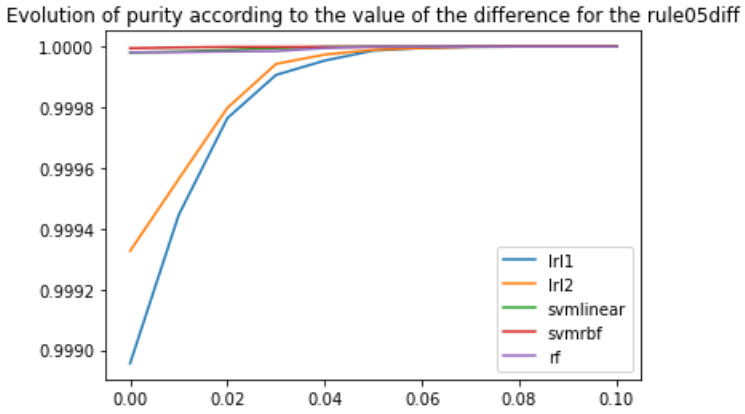


Figure 4.1: Variation of purity metric according to the value of the threshold

We observe that for K-means clustering, the optimal rules to build our training sets are:

- **the 0.5 rule with a difference of 0.08** for the datasets **RLR1** and **RLR2**
- **the 0.5 rule with a difference of 0.05** for the dataset **SVM linear** and **RF**
- **the 0.5 rule with a difference of 0.07** for the dataset **SVM rbf**

Later, we run all our test with K-means on the datasets built with the above rules.

4.1.2. Performance of K-means clustering on the test data. Now that we have built the optimal training dataset, we evaluate the rejection accuracy of K-means. We analyse visually the map for points of interests, and use the two methods previously introduced with the labelled polygons and the missing class.

Visual assessment on the map

For each dataset, we plot the map obtained after clustering and rejection (figure 4.2) and we take a closer look at some specific areas.

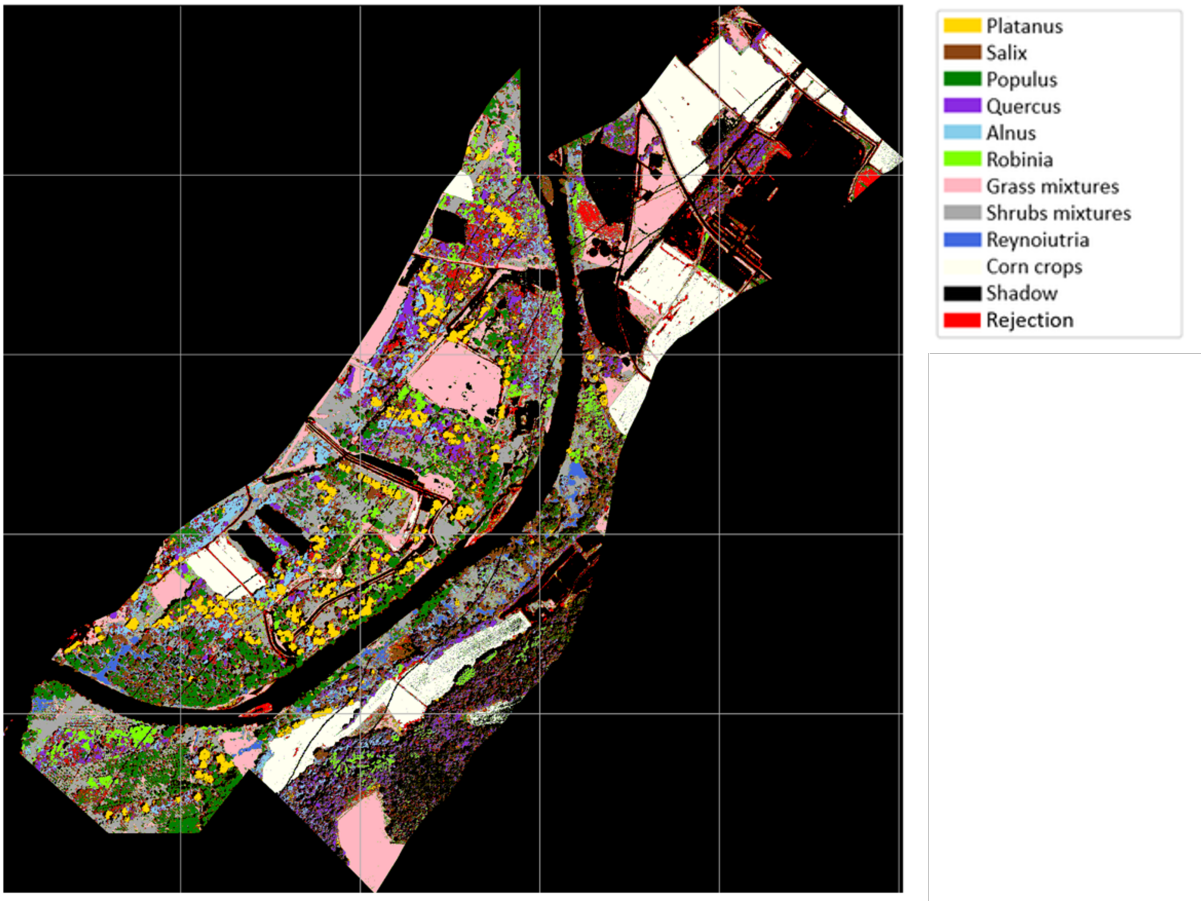


Figure 4.2: Map after clustering and rejection with K-means on the dataset RLR l1

Table 4.2 summarizes the areas of interest rejected by K-means in the different datasets.

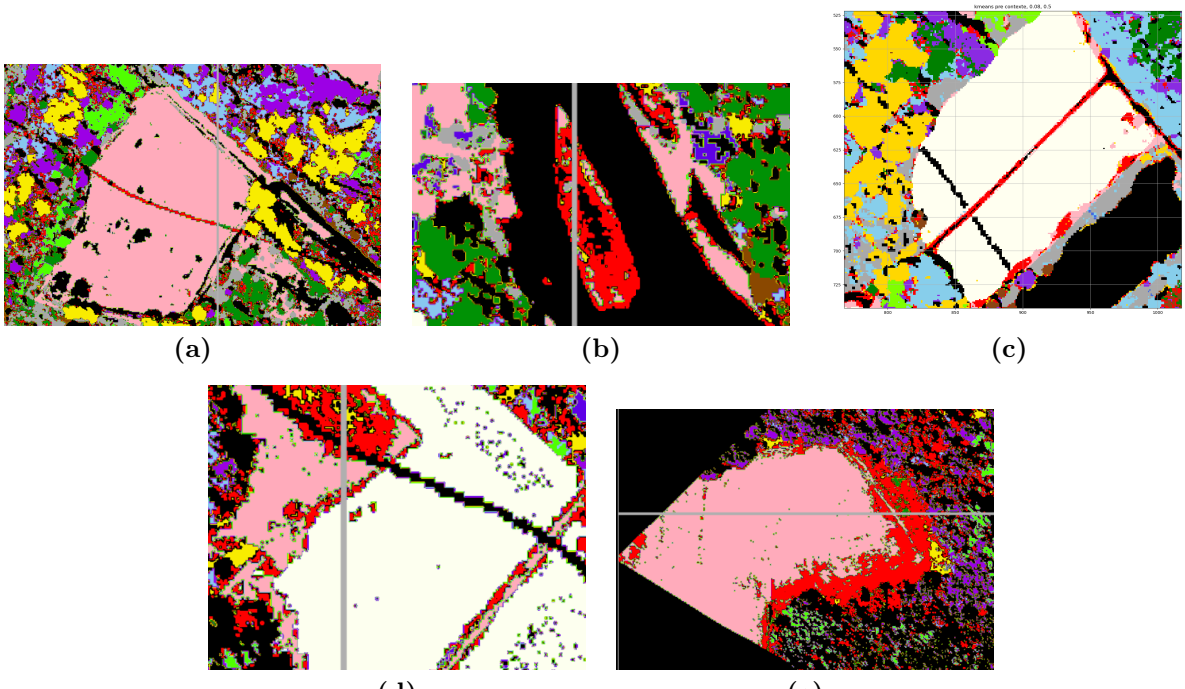
	RLR l1	RLR l2	SVM linear	SVM RBF	RF
Electricity line	×	✓	×	×	×
Fallow land	✓	partly	✓	✓	✓
Dirt road	✓	partly	✓	✓	partly
Borders between species	✓	partly	✓	✓	✓
Island	✓	partly	×	✓	partly

Table 4.2: Visual assessment of the rejected elements for K-means algorithm

These areas are displayed in Figure 4.3 (these results have been extracted from different datasets), the rejected pixels appear in red. We notice that some of the elements have not been rejected on all of the datasets. This is explained by the fact that they were classified differently by the initial algorithm (Figure 4.4). For example, we notice that the island has been rejected for the datasets RLR l1 and SVM RBF, but only partly or not at all for the others. Yet, we observe that for these two datasets, the pixels on the island are classified as *Platanus*, while on the three others they are classified as grass mixtures. In reality, the plants on the island do not belong to any of the species that we want to classify, but they spectrum is very close to that of grass mixtures. Thus, K-means is not able to reject them in the case where they are classified as grass mixtures. We observe something similar for the electricity line, which is never detected except for the dataset RLR l2. Yet, in this dataset, the pixels of the electricity line have been initially classified as Robinia, while for the other datasets, they have been classified as Corn crops, like the rest of the field. Then, we notice that the dirt road is only partly rejected for the algorithms RLR l2 and RF. On the three other datasets, this road appears as a mixture of many different species, so it is clear that the pixels have been misclassified. On the dataset RLR l2, we observe the same thing; however, we notice that the dirt road appears thinner than on the other datasets, which may have impacted K-means' capacity to reject it. Finally, on the dataset

RF, the dirt road is classified as grass mixtures, which seems coherent, we can expect the vegetation on this road to be similar to grass. After the rejection step, we can see that K-means did not reject the road, but it rejected a part of the limit between the road and the field, which was initially classified as a mixture of many different species. So, K-means proves to be quite efficient at rejecting borders between different species. This can also be observed by looking at the complete map (Figure 4.2).

To conclude this visual assessment, the results obtained with the rejection method are strongly impacted by the results obtained on the initial dataset with the five initial classification algorithms.



(a) **(b)** **(c)**

(d) **(e)**

Figure 4.3: **(a)** Electricity line **(b)** Island **(c)** Dirt road crossing a field **(d)** Borders between two species **(e)** Fallow land

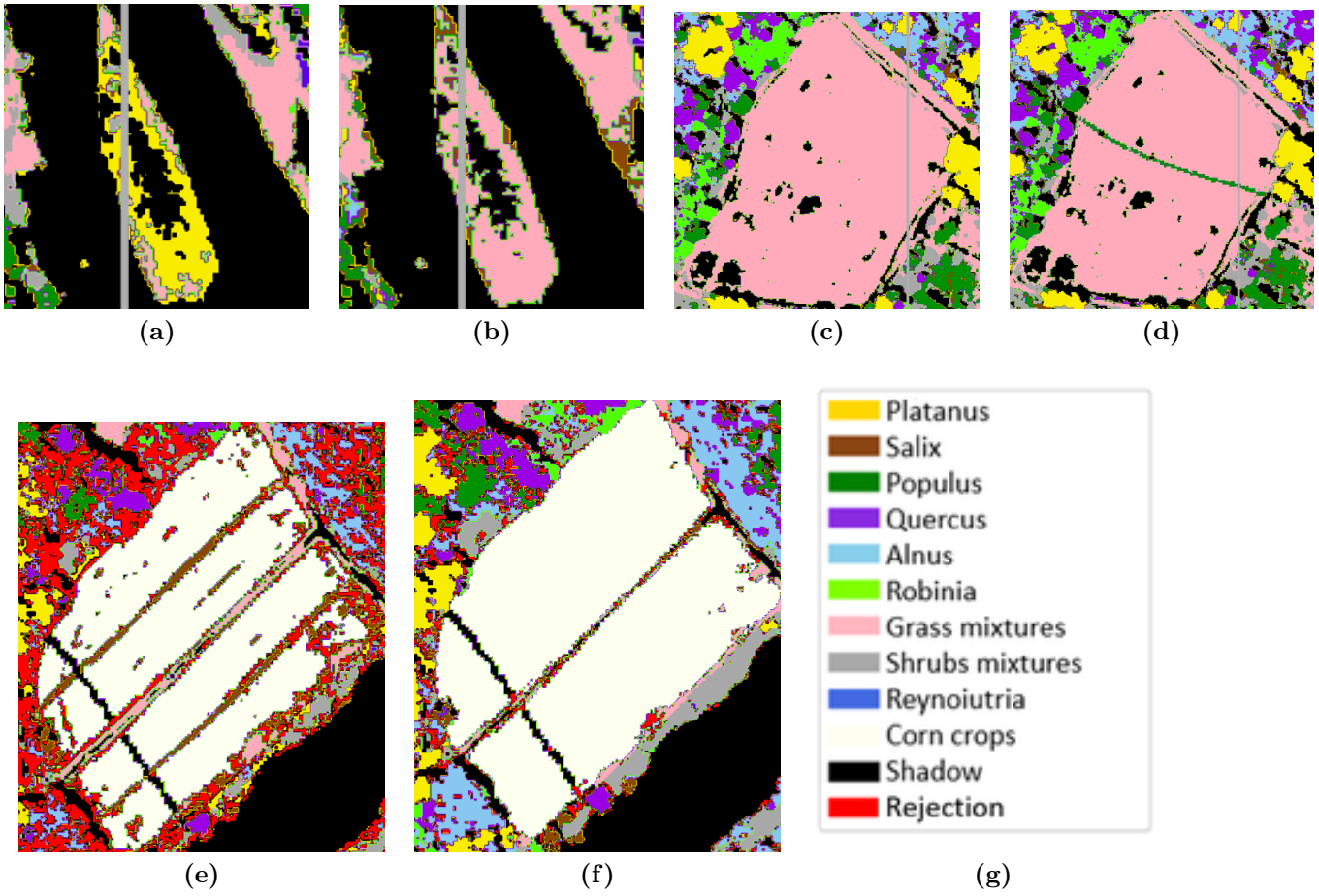


Figure 4.4: **(a)** Island RLR l1 **(b)** Island RF **(c)** Electricity line RLR l1 **(d)** Electricity line RLR l2 **(e)** Dirt road RLR l1 **(f)** Dirt road RLR l2 **(g)** Legend

The labelled polygons

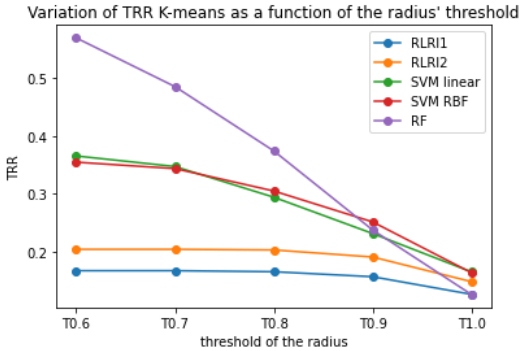


Figure 4.5: Variation of True Rejected Rate (K-means)

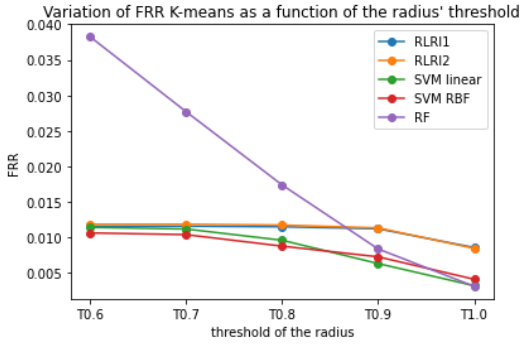


Figure 4.6: Variation of False Rejected Rate (K-means)

Table 4.3: Non Rejected Accuracy (NRA) (K-means)

dataset	RLR l1	RLR l2	SVM linear	SVM RBF	RF
NRA	0.94	0.94	0.94	0.94	0.90

Ideally, we should obtain a high **True Rejected Rate**, a low **False Rejected Rate** and a high **Non Rejected Accuracy (NRA)**. We vary the value of the radius' threshold and observe the results obtained for those indicators.

First, we can see that **NRA** is always quite high; it is about 94% for the datasets RLR l1, RLR l2, SVM linear and SVM RBF and 90% for RF (Table 4.3). NRA is not impacted by the radius' treshold. NRA represents the ratio of pixels that should not be rejected and that have been well-classified against the total number of pixels which should not be rejected. This indicator confirms that the classification performed by K-means is quite efficient. We can consider that the rules we used to build the training set are quite good.

Moreover, **FRR** represents the ratio of pixels that have been incorrectly rejected, against the total number of pixels that should not be rejected. Thus, this indicator allows us to control whether we reject too much or not enough. FRR decreases when the radius' threshold increases (Figure 4.6). The wider the radius, the less we reject and thus the risk of wrongly rejecting decreases as well. FRR is always quite low. For the datasets RLR l1, RLR l2, SVM linear and SVM RBF, FRR is about 0.5% to 1%. We can conclude that for those algorithms, increasing the threshold of the radius does not have small impact on FRR. For the dataset RF, we observe that FRR is more impacted by the radius' threshold, it varies between 0.5% and 4%. For this dataset it may be more important to calibrate the threshold correctly.

Then, **TRR** represents the ratio of pixels that have been correctly rejected, against the total number of pixels that should be rejected. The result varies between the different datasets and according to the value of the radius' threshold (Figure 4.5). We obtain similar results for RLR l1 and RLR l2 for which TRR is about 20% and does not vary with the radius' threshold. We also obtain similar results for SVM linear and SVM RBF for which TRR is more impacted by the threshold and varies from 20% to 35%. Finally, the dataset RF is the most impacted by the threshold. We obtain the best result for a threshold $T = 0.6$ for which $TRR = 57\%$. However, for $T = 0.6$, FRR is also maximal, so it means that we reject more but not necessarily that we reject better. As we can see, TRR decreases when the threshold increases because we reject less we are less likely to reject pixels which should be classified.

We take into account the context to see if the results can be improved. We test on the datasets **RLR l1** and **SVM RBF** with a radius' threshold equal to 0.8. We can see that NRA (Figure 4.9) improves, but mostly stays around 94%. FRR (figure 4.8) also improves, it is divided by 1.5 for the dataset **RLR l1** and by 2 for the dataset **SVM RBF**. These are the best results, they are obtained with the second type of context. Moreover, with the second type of context, TRR (Figure 4.7) decreases by 2% to 5%. Thus, taking the context into account only reduces the false rejection rate but does not improve the true rejection rate. For example, comparing the results on the field without context (Figure 4.10) and with context (Figure 4.11), we can see that using the context has erased some of the rejected points in the field. Regarding their location, it is reasonable to assume that these points are grass mixtures like the rest of the field. To conclude, using the context makes the image smoother.

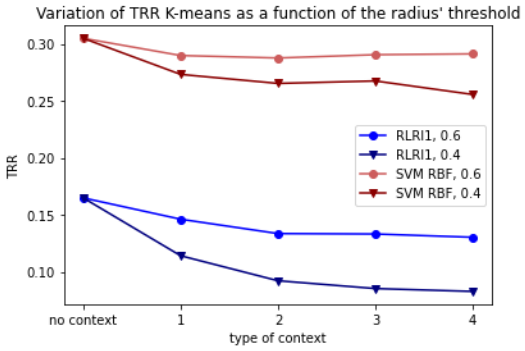


Figure 4.7: Variation of True Rejected Rate with context (K-means)

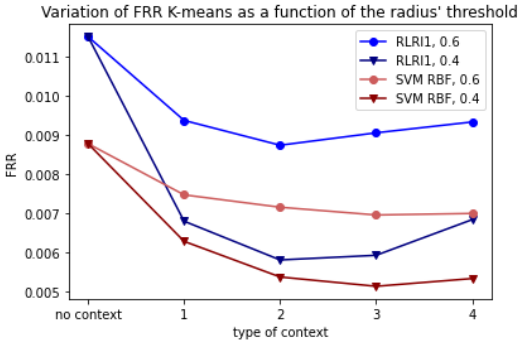


Figure 4.8: Variation of False Rejected Rate with context (K-means)

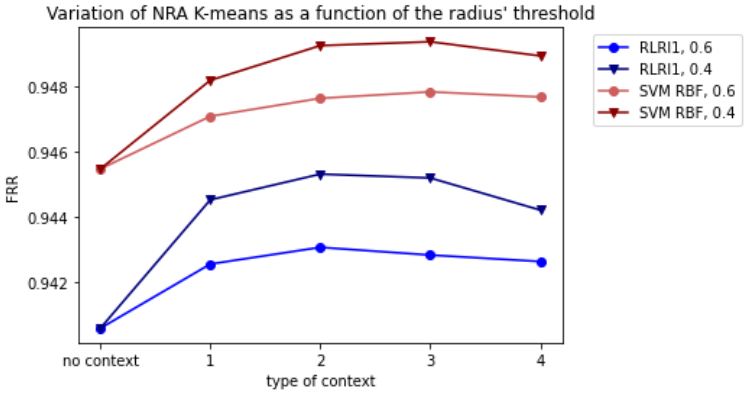


Figure 4.9: Variation of Non Rejected Accuracy with context (K-means)

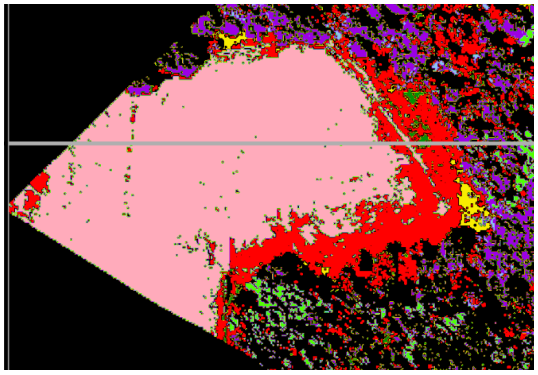


Figure 4.10: Rejection without context (K-means)

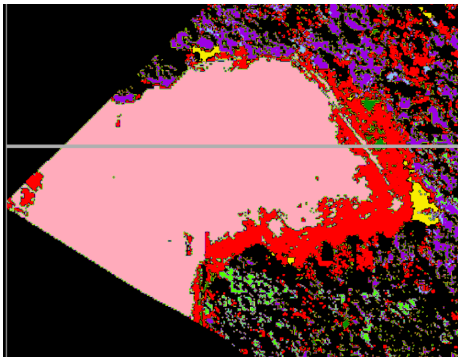


Figure 4.11: Rejection with context (K-means)

To conclude, the K-means algorithm obtains good results regarding NRA and FRR, which means that applying this rejection method does not impact negatively the classification previously carried by the initial five algorithms. TRR varies from 20% to 57% on the different datasets, so we improve the classification as we reject pixels which were initially classified whereas they are outliers. However, we highlight the fact that the data labelled by the polygons are specific trees from different species on the map. They do not take into account the rejection of borders between tree species so only one of the two types of rejection is evaluated with this method. Moreover, only a few trees are labelled and most of them are species that should not be rejected. So, this method of evaluating the rejection performance has its limitations.

The missing class

Table 4.4: True rejection rate on the datasets with a missing class

radius' threshold	0.6	0.7	0.8	0.9	1
RLR l1 without Platanus	0.042	0.042	0.042	0.042	0.031
SVM RBF without Platanus	0.451	0.451	0.435	0.385	0.321
RLR l1 without Reynoiutria	0.183	0.183	0.183	0.183	0.157
SVM RBF without Reynoiutria	0.157	0.157	0.155	0.136	0.107

The true rejection rates obtained on Platanus and Reynoiutria species are quite inhomogeneous, as they vary from 3% to 45%. Again, we observe that the smaller the radius' threshold, the better the true rejection rate. Tests are carried out using the same optimal training set as in the tests with labelled polygons so we can still control the fact that we should not reject too much (FRR being low).

4.2. Results of SVM clustering. 1) Assessment of the performance of the SVM algorithm on the polygon masks taking or not into account the context

Table 4.5: Polygon assessment of the SVM methods

	OVO				OVR			
	TRR	FRR	NRA	STRR	TRR	FRR	NRA	STRR
Median								
SVM2	0.1795	0.0091	0.9446	0.2096	0.1946	0.0044	0.9474	0.3
RLR l2	0.095	0.0079	0.943	0.075	0.1094	0.0049	0.9446	0.0949
RF	0.3001	0.0278	0.927	0.3394	0.3476	0.0129	0.9382	0.4412
RLR l1	0.0636	0.0069	0.9432	0.0447	0.0993	0.0051	0.9444	0.0807
SVM1	0.1971	0.0082	0.9474	0.2640	0.1976	0.003	0.9502	0.3245
3rd quartile								
SVM2	0.2688	0.0103	0.9438	0.3263	0.2410	0.0053	0.9471	0.3192
RLR l2	0.1421	0.0104	0.9416	0.1112	0.1529	0.0082	0.9433	0.1342
RF	0.4602	0.0399	0.9159	0.5307	0.453	0.033	0.9212	0.5921
RLR l1	0.1144	0.0099	0.9418	0.0868	0.1223	0.0063	0.9439	0.0912
SVM1	0.2749	0.0096	0.9463	0.3974	0.2321	0.0069	0.9483	0.3412

TRR(True Rejection Rate), FRR(False Rejection Rate) NRA(Non Rejected Rccuracy) STRR(Specific True Rejection Rate)

Without context

Table 4.5 depicts the overall performance of rejection and classification according to each of the five datasets. First, **NRA** is greater than 90% for all the datasets and algorithms, which proves that our decision rule and classification without considering the reject is efficient. **FRR** highest value is about 4% and for most of the algorithms, FRR is lower than 1%; our algorithms do not incorrectly reject species. On the contrary, our algorithms have a lack of power of rejection on pixels which should be rejected, as the greatest **TRR** doesn't exceed 46% with a mean of 21.2%. However, as mentioned previously in section 4.1 and according to our visual assessment, we notice that our rejection algorithms are more likely to reject boundaries, and this is not taken into account in the calculation of this rate. Moreover, there are only 28086 pixels which represents 1.8% of our non-shadowed pixels of the datasets. These areas correspond to specific and gathered parts of the datasets, which can explain these results. We compute the **STRR** which corresponds to the rejection rate among the classes whose spectrum is the furthest from the other species' spectra. We can see that the STRR is higher than the FRR which shows that our algorithms are more likely to reject these specific species.

With context on SVMRBF and RLR l1 datasets

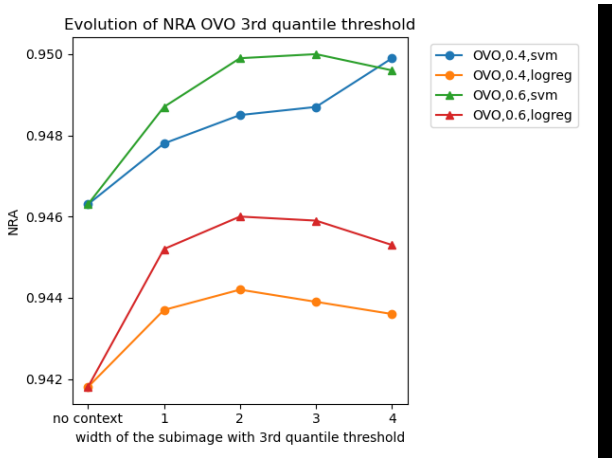


Figure 4.12: NRA OVO rejection threshold based on the 3rd quartile

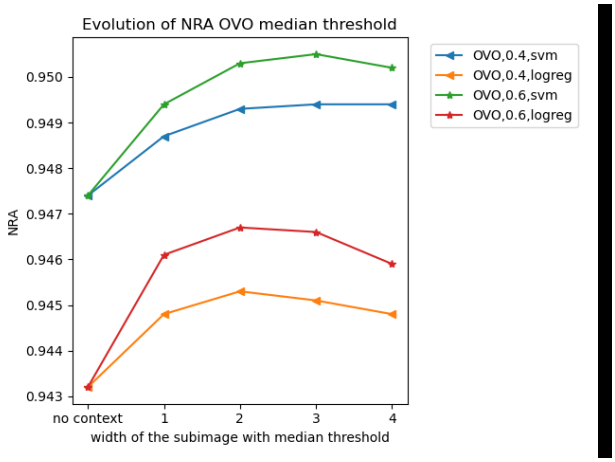


Figure 4.13: NRA OVO rejection threshold on the median

Figure 4.12 and 4.13 display the variation of NRA as a function of the width of the sub-image. A plateau is reached when the context option is the second one, as the considered sub-image is too large, and taking into account the context doesn't improve the non-rejected accuracy. With SVM algorithms the optimal width of the image is 2, which corresponds to 25 pixels. The rate of well-classified pixels is almost the same between

the two rejection thresholds. OVR-SVM has a higher accuracy than OVO whereas this algorithm is supposed to have a higher bias.

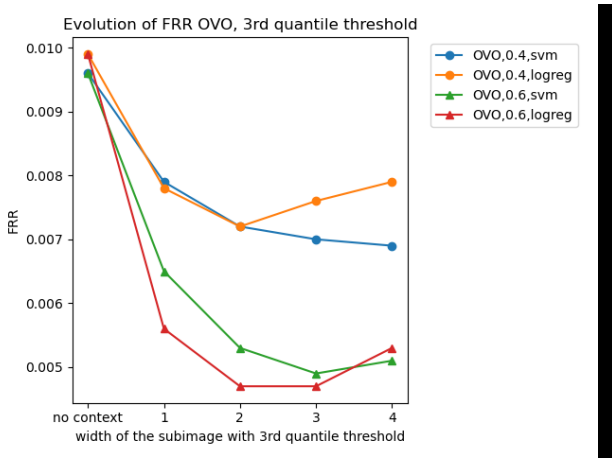


Figure 4.14: FRR OVO rejection threshold based on the 3rd quartile

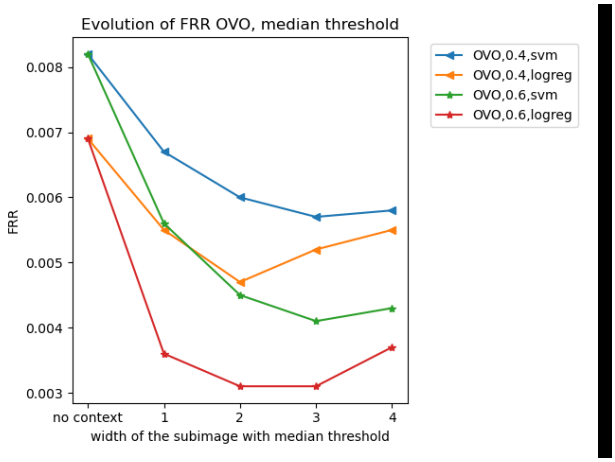


Figure 4.15: FRR OVO rejection threshold on the median

Figure 4.16 and 4.17 report the variation of FRR according the width of the sub-image used in the algorithm of the context. For each algorithm, we observe that FRR decreases when the threshold increases, which shows the influence of our context algorithm on the dataset. The dataset is being smoothed as taking the context into account leads to classifying pixels which were previously rejected whereas they were not supposed to.

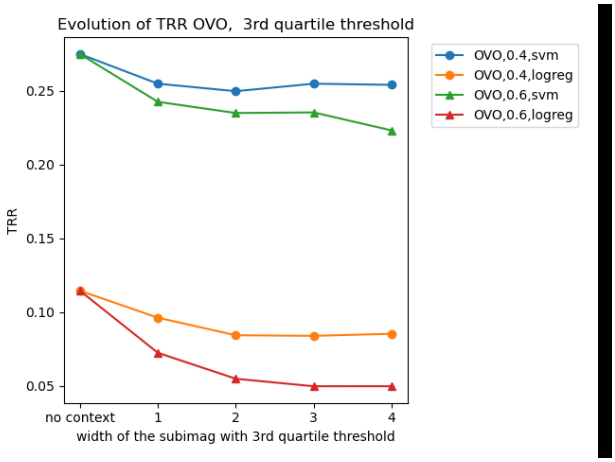


Figure 4.16: TRR OVO rejection threshold based on the 3rd quartile

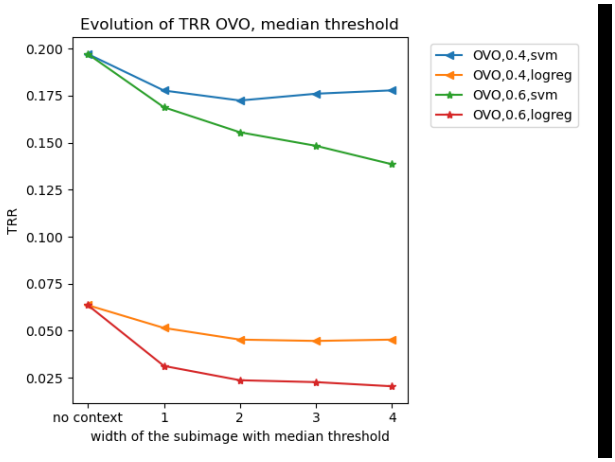


Figure 4.17: TRR OVO rejection threshold on the median

2) Assessment of the performance of the SVM algorithms on the polygon masks on datasets with a missing class (Platanus and Reynoitruia)

Table 4.6 provides the TRR, FRR, NRA and STRR for the models which have been trained with the dataset in which we have removed a plant species (Platanus or Reynoiutria). We want these species to be rejected. The TRR, FRR, STRR and NRA are computed on the pixels labelled by the polygons instead of the entire dataset, otherwise we do not have any way of quantifying the accuracy of the rejection. We can see that FRR is low NRA is high which underlines the accuracy of our algorithms. However, TRR and STRR, which corresponds to the rate of Platanus (or Reynoiutria) which are well-rejected do not exceed 28% for TRR and 36% for STRR, which is lower than the results with the K-means algorithm.

4.3. Results of DBSCAN clustering. The main problem of the DBSCAN algorithm is that it automatically crashes the kernel when applied to the large database. To counter this problem, we divide the large

Table 4.6: Assessment of the performance of SVM methods with polygons on the datasets with a missing class

	OVO							
	Median				**3rd quartile**			
	TPR	FRR	NRA	specific rejection	TPR	FRR	NRA	specific rejection
SVM1 without Platanus	0.16	0.01	0.90	0.20	0.24	0.014	0.90	0.3084
RLR1 without Platanus	0.037	0.016	0.89	0.0048	0.052	0.0312	0.88	0.0093
SVM1 without Reynoiutria	0.0758	0.0067	0.93	0.065	0.1157	0.0098	0.93	0.08
RLR1 without Reynoiutria	0.13	0.00786	0.93	0.24	0.1929	0.014	0.92	0.35

image into several sub-images and apply the algorithm to each of them. However, this technique gives poor results. This can be explained by the choice of the *min_sample* parameter which depends on the considered database. Depending on the image we choose, the recognized clusters are not the same. For example, on some portions *Platanus* may be more present than *Quercus* (so they are considered as rejected because they are a minority) and vice versa. As an example, an image showing the results of DBSCAN in the large database is presented in the appendix C.1. We can notice that a large part of the pixels is rejected. Thus, we consider in this section the results obtained on the small dataset extracted from the complete image. The choice of epsilon is made with the help of Figure 3.2 and remains the same for all our tests. We choose the *min_sample* parameter empirically, and make sure that we obtain the same number of clusters as the number of species in our original database. Once the parameter is set, we must choose the metric. Table 4.7 shows that the proportion of rejected pixels varies significantly depending on the chosen metric. For example, for the Cosine and Correlation metrics, only 0.2% of pixels are rejected, which does not seem important enough. This can be explained by the fact that these metrics measure the degree of similarity between the probability vectors. However, the spectra of the considered species are generally very similar. The Cosine and Correlation metrics therefore consider that the distance between the probability vectors is very small, which explains why so few pixels are rejected. Concerning the other metrics, we obtain more or less the same results. Thus, the best way to select the most accurate metric is to analyze TRR, FRR and NRA scores.

The scores for each of the selected metrics are presented in Table 4.8. First, we notice that TRR is always zero. This is due to the fact that the number of real rejected pixels is low on the small dataset.

Also, we can visually evaluate our results. For example, on the different images in Figure 4.18 representing the rejected pixels for each of the metrics, we can see that globally the pixels located on the borders between two species are well rejected. However, we find a great number of rejected pixels everywhere in a random way.

Table 4.7: Percentage of rejected pixels for each metric used on the small dataset

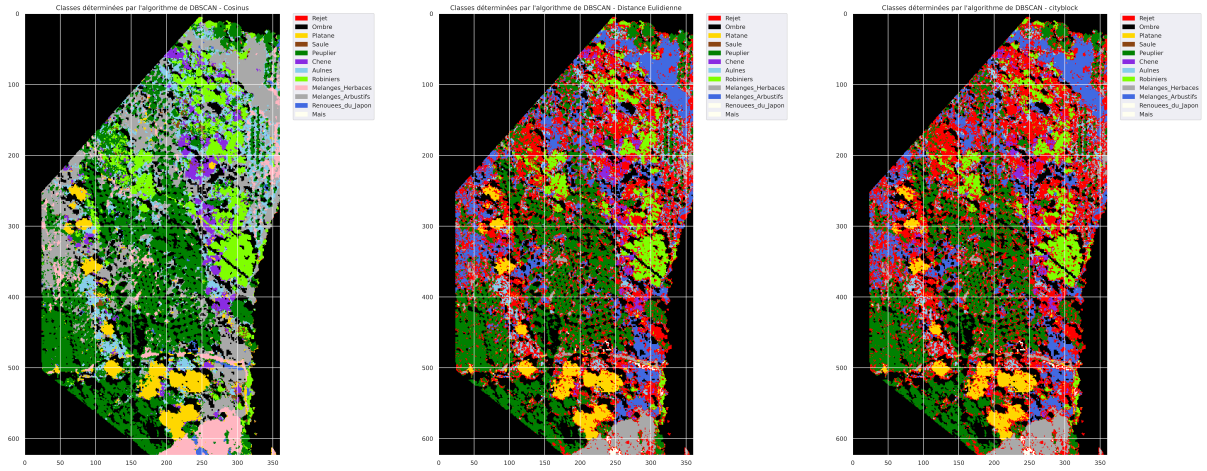
Metric	Euc.	Cos.	Correlation	Cityblock	Braycurtis	Manhattan	Minkowski
Percentage	32%	0.2%	0.2%	40%	27%	37%	29%

Table 4.8: FRR, NRA for the small dataset for each metric

Metric	Euc.	Cityblock	Braycurtis	Manhattan	Minkowski	Euc without Platanus
FRR	0.24	0.29	0.20	0.29	0.21	0.13
NRA	0.03	0.03	0.03	0.03	0.03	0.42

We can observe in Tables 4.7 and 4.8 rather poor results concerning the DBSCAN method applied on the small database. However, we have to keep in mind that as we only use the small database, there are very few polygons present on this portion of the image. The TRR score is not displayed because it is worth 0 in any case. Indeed, on the small dataset there are only 23 pixels that we have to reject since they belong to other species but DBSCAN did not. However, it is not relevant to judge the performance of the algorithm based on the rejection of only 23 pixels. Contrary to the previous methods, here FRR is quite high: DBSCAN wrongly rejects 20% of the pixels. Visually this is noticeable when looking at Figure 4.18: many parts of the image are red, except for the Cosine metric where very few pixels have been rejected and we can nearly not see the rejected red points on the image.

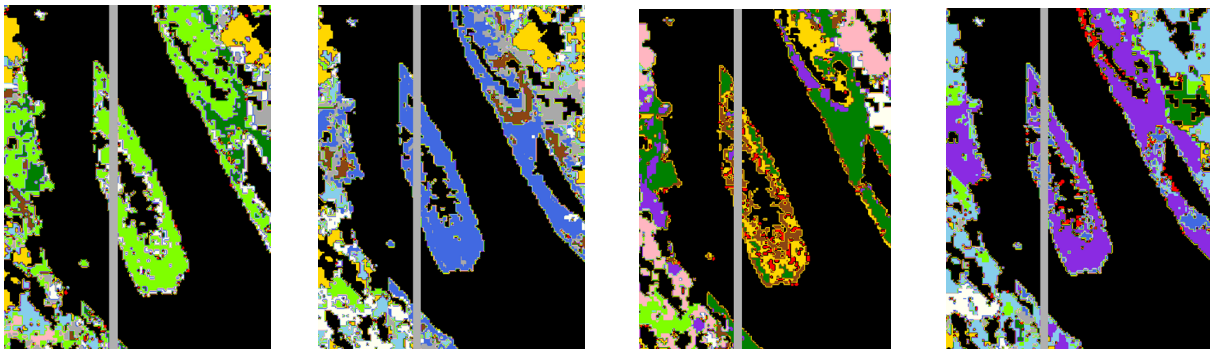
However, by removing *Platanus*, we can improve our results considerably. In particular, for NRA, we manage to boost the score from almost 3% to 40%. This can be explained by the fact that *Platanus* is a species whose spectrum is quite different from the others so DBSCAN can distinguish them more easily. Moreover the STRR score (as defined earlier) equals 0.22 which indicates that DBSCAN does not misclassify 22% of the *Platanus*.



(a) DBSCAN - Cosine metric (b) DBSCAN - Euclidean metric (c) DBSCAN - Cityblock metric

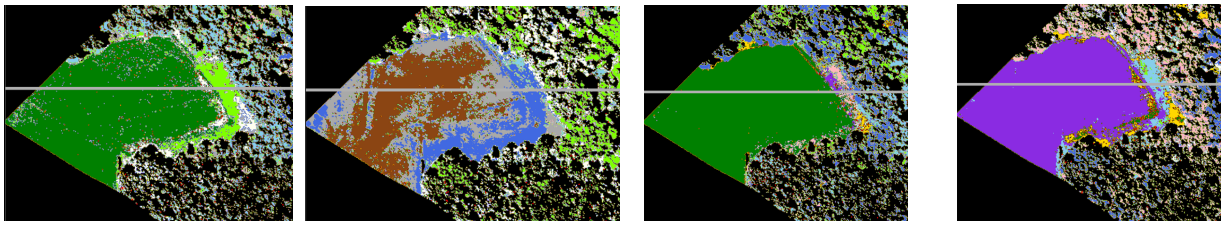
Figure 4.18: Rejection on the small dataset by DBSCAN using different metrics

4.4. Results of rejection with GMM. We run the method presented in section 3.2.4 using the GMM and K-means clustering algorithms. First, we simply perform a visual evaluation to know if this algorithm rejects the areas of interest such as the electricity line, the fallow land, the island, the dirt road, e.g.. We first notice that a single loop of this algorithm takes a long time to run on our computers, even after parallelizing the code. Then, we observe that the algorithm does not reject the areas of interest at all, regardless of the rejection threshold $T > 0$ used. The results are presented in Figures 4.19 and 4.20.



(a) GMM - RLR l1 (b) GMM - SVMrbf (c) K-means - RLR l1 (d) K-means - SVMRBF

Figure 4.19: Rejection of the island for T = 0.50 with RLR l1 and SVMrbf datasets for GMM and K-means clustering algorithms



(a) GMM - RLR l1 (b) GMM - SVMrbf (c) K-means - RLR l1 (d) K-means - SVMrbf

Figure 4.20: Rejection of the fallow land for T = 0.50 with RLR l1 and SVMRBF datasets for GMM and K-means clustering algorithms

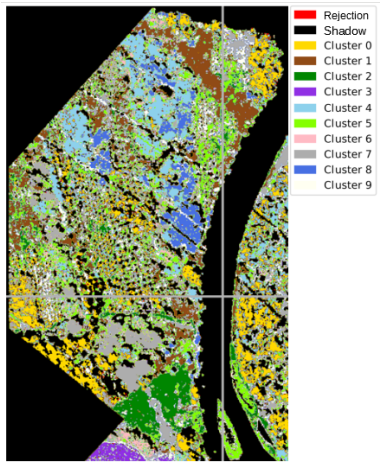


Figure 4.21: Rejection on the dataset RLR1 after applying GMM clustering

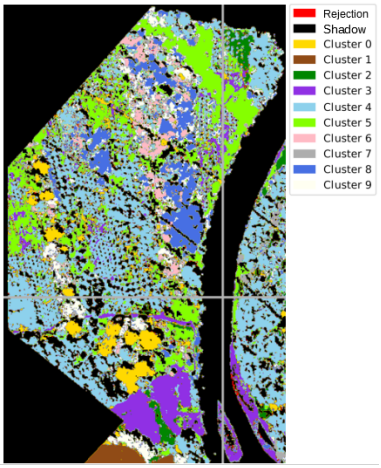


Figure 4.22: Rejection on the dataset SVMRBF after applying K-means clustering

In general, this algorithm gives a low and inconsistent rejection in the whole satellite image, whatever the parameters' values as we can see in Figures 4.21 and 4.22. We also notice that the GMM algorithm gives a worse clustering than K-means. Given the poor results of this method, we do not test the method further on polygons or on a missing class.
These results can be explained by the fact that we are classifying probability vectors, which is different from content-style feature vectors constructed in the paper [Deng et al., 2019] to describe both the style (texture, color, brushwork, etc.) and content (object, scene, etc.) features of a painting image. This can also be explained by the fact that we apply the algorithm on all non-shadow pixels instead of selecting the well classified pixels with the rules stated in the section 3.1.1.

5. Conclusion and perspectives. We have introduced four post classification rejection algorithms which aim to reject pixels that are considered outliers. We can distinguish two types of outliers: pixels at the border between two species and pixels that do not belong to any of the classified classes. The clustering methods that have been tested are DBSCAN, GMM, K-means and SVM. K-means and SVM algorithms require two steps. First, the training set is built, it contains almost-sure well-classified pixels. Second, the remaining pixels that are considered missclassified undergo a rejection method to be reclassified or rejected. The GMM-based algorithm also involves two steps. The first step is to apply a clustering algorithm such as K-means or GMM on the set of pixels that are not shadow pixels. The second step is to compute the probability that each pixel belongs to its assigned cluster using gaussian distributions and to reject when this probability is too low. Contrary to K-means, SVM and GMM algorithms, DBSCAN requires only one step. There is no data selection/preparation step since DBSCAN detects outliers on its own. However, the parameters of the algorithm must be chosen carefully.

The visual assessment of the results obtained with K-means and SVM seems quite good. We can observe the significant effect of the rejection process on the boundaries and some areas of interest. Rejection on the labelled polygons gives quite inhomogeneous results depending on the dataset. The true rejection rate does not exceed 57%, but it still means that we improve the classification as without any rejection option, those pixels would be classified as one of the considered species. Moreover, K-means and SVM wrongly reject very few pixels as the false rejection rate is always quite low, mostly around 1%, and they have a good accuracy in terms of classification, the NRA being about 94%.
The visual results for GMM are poor as the rejection is low and inconsistent especially in the areas of interest. This is why we do not test the method further on polygons or on a missing class. The results given by DBSCAN on the large database are poor but rather exploitable on the small dataset despite low results.

However, it should be noted that assessing rejection algorithms is a real problem as there are few labelled data points compared with the size of the dataset, labelling being a time-consuming task. What is more, among the points that are labelled, only a small minority should actually be rejected and the other are part of the classes we want to classify.

This work suggests further research to compare the choice of clustering algorithms with other algorithms involving Convolutional Neural Network (CNN) [Laroui et al., 2021] and rejection tasks integrated during the classification algorithm.

6. Github. To see the codes please go to this link: https://github.com/Eva427/Projet-classifiction-espces-v-g-tales.git

Acknowledgement.
We would first like to thank our tutors Mr Rollin Gimenez and Mrs Sophie Fabre for having given us their trust to carry out this research which is part of their research. Thank you for your valuable guidance throughout this research and your benevolence.

In addition, we would like to express our gratitude to Mrs Juliette Chevalier, Mr. Joris Guerin and Mr. Olivier Roustant for the time spent giving us precious advice and hints which allowed us to start this research properly.

Appendix

A. Pseudo-codes of the presented methods.

Algorithm A.1 Classification with context

• **Initialisation:**
Set the size m of the sub-image
Define s, the threshold (%) of presence of a class
• **Algorithm :**
for i in all non-classified pixels **do**
    Extract the ith sub image of size $(2 * m + 1)^2$ surrounding the ith non classified pixel
    if The majority vote class c of the sub-image is s% represented and c $\neq$ shadow class **then**
        $class_i \leftarrow c$
    end if
end for

A.1. Algorithm of classification with context.

Algorithm A.2 EM algorithm

Initialisation:
Set the number K of clusters
Randomly draw K centroids among the data points
while centroids are moving **do**
    Allocation update: Data points are re-allocated to their closest centroid.
    Centroids update: Centroids are re-computed as the mean of the new clusters obtained at the previous step.
end while

A.2. EM Algorithm.

Algorithm A.3 Rejection with K-means

• **Initialisation**:
- The matrix obtained by the clustering algorithm containing the cluster of each pixel. The clusters are noted $\mathcal{C}_k,\ \forall k \in \{1..K = 10\}$
- The cluster centres c_k
- The rejection threshold to be calibrated T
• **Cluster radius calculation:**
for each cluster $\mathcal{C}_k$ (based on classified pixels) **do**
    for each pixel x_i belonging to cluster $\mathcal{C}_k$ **do**
        - Compute $d(x_i, c_k),\ \forall i \in \{1..\#\mathcal{C}_k\}$ where $d(.,.)$ is a given distance
    end for
    - Compute the radius of cluster k: $R_k = \max_{i \in \{1..\#\mathcal{C}_k\}} d(x_i, c_k)$
end for
• **Reject option:**
for each unclassified pixel x_i **do**
    - Compute $d(x_i, c_k),\ \forall k \in \{1..K\}$ the distances from x_i to the centre c_k of each cluster.
    - Compute $k^* = \text{argmin}_{k \in \{1..K\}} d(x_i, c_k)$
    if $d(x_i, c_{k^*}) \geq T \times R_{k^*}$ **then**
        Reject x_i as outlier
    else
        x_i belongs to cluster $\mathcal{C}_{k^*}$
    end if
end for

A.3. Rejection algorithm with K-means.

Algorithm A.4 Rejection method with SVM

Train the calibrated OVO (or OVR) SVM classifier the training dataset (obtained with the dataset extraction rule)
Obtain the matrix of decision function ($10 * number_{pixels\ to\ be\ predicted}$
Sort in ascending order the vector of relative difference decision function to the highest decision function for each pixel, and keep the second lowest relative difference denoted diff
if Rejection rule = median **then**
    Rejected all pixels with a $diff_i$ lower than the median of diff
else if Rejection rule = third quartile **then**
    Rejected all pixels with a $diff_i$ lower than the third quartile of diff
end if

A.4. Rejection algorithm with SVM.

Algorithm A.5 DBSCAN algorithm

Initialisation:
Set the value of eps and $minPts$
for each point A **do**
    Compute its distance d from all the other points B_i
    if $d \leq eps$ **then**
        Mark B_i as a neighbor of A
    end if
    if A has $minPts$ neighbors **then**
        Mark A as a core point
    end if
end for
for each core point C **do**
    if C is not already assigned to a cluster **then**
        Create a new cluster ; recursively find all its neighboring points and assign them the same cluster as the core point
    end if
end for

A.5. DBSCAN Algorithm.

Algorithm A.6 Reject option using GMM

• **Initialisation:**
- The matrix obtained by the clustering algorithm containing the cluster of each pixel
- The rejection threshold to be calibrated T
• **Algorithm:**
while stopping criterion is not met **do**
  for M in number of clusters **do**
    for k in number of clusters **do**
      - Compute the mean μ_k of cluster k
      - Compute the covariance matrix Σ_k of cluster k
      - Compute $\mathbb{P}(k)$ for cluster k
    end for
    for each probability vector $u_i \in$ cluster M **do**
      - Compute $\mathbb{P}(u_i|M) =$ multivariate normal distribution of $(u_i, \mu_M, \Sigma_M) \in \mathbb{R}^D$ for each cluster M using μ_k and Σ_k calculated in the previous *For* loop.
      - Compute $\mathbb{P}(u_i) = \sum_{k \in clusters} \mathbb{P}(u_i|k)\mathbb{P}(k)$
      if $\mathbb{P}(u_i) \neq 0$ **then**
        $\mathbb{P}(M|u_i) = \frac{\mathbb{P}(u_i|M)}{\mathbb{P}(u_i)} \in \mathbb{R}^D$
      end if
      if $\mathbb{P}(M|u_i) < T$ **then**
        Reject u_i as outlier
      else
        u_i belongs to cluster M
      end if
    end for
  end for
end while

A.6. Rejection algorithm using GMM.

Table B.1: Percentage of rejected pixels for OVR SVM method

		0.75 quantile			median		
OVR		TRR	FRR	NRA	TRR	FRR	NRA
1) 2 dataset : logreg and svm,difference rule 0,1	OVR without context	0,2817	0,014	0,9406	0,1666	0,0094	0,9422
	context 1, 0.6	0,2608	0,0114	0,9434	0,1522	0,0075	0,9447
	context 2, 0,6	0,2565	0,0109	0,9439	0,1464	0,0072	0,9451
	context 3, 0.6	0,2573	0,011	0,944	0,1471	0,0075	0,9449
	context 4, 0,6	0,2601	0,0111	0,9437	0,1478	0,0075	0,9447
	context 1, 0.4	0,2724	0,0105	0,9447	0,1532	0,0068	0,9459
	context 2, 0,4	0,259	0,009127	0,9459	0,1414	0,0062	0,9464
	context 3, 0.4	0,2562	0,00928	0,9457	0,1385	0,0062	0,9464
	context 4, 0.4	0,2536	0,009	0,9457	0,136	0,0062	0,9462
2) 2 dataset:logreg and svm, 0.5 rule	OVR without context	0,2817	0,0141	0,9406	0,1666	0,0094	0,9423
	context 1, 0.6	0,2609	0,0115	0,9434	0,1522	0,0075	0,9448
	context 2, 0,6	0,2566	0,0109	0,9439	0,1464	0,0072	0,9451
	context 3, 0.6	0,2573	0,011	0,9439	0,1471	0,0075	0,9449
	context 4, 0,6	0,2602	0,0111	0,9437	0,1479	0,0075	0,9447
	context 1, 0.4	0,2724	0,0105	0,9448	0,1533	0,0069	0,9459
	context 2, 0,4	0,2591	0,0091	0,9459	0,1414	0,0062	0,9465
	context 3, 0.4	0,2562	0,0093	0,9458	0,1385	0,0062	0,9464
	context 4, 0.4	0,2537	0,0091	0,9457	0,136	0,0062	0,9462
3) 1 dataset:logreg 0,08 rule difference	OVR without context	0,1223	0,0063	0,9439	0,0993	0,0051	0,9444
	context 1, 0.6	0,107	0,0052	0,9452	0,0849	0,0042	0,9458
	context 2, 0,6	0,0953	0,00482	0,9456	0,0734	0,0039	0,9459
	context 3, 0.6	0,0957	0,005	0,9453	0,073	0,004	0,9456
	context 4, 0,6	0,0964	0,0051	0,9452	0,0759	0,0041	0,9455
	context 1, 0.4	0,0874	0,0043	0,956	0,0687	0,00359	0,9462
	context 2, 0,4	0,068	0,0034	0,9467	0,0474	0,0028	0,9468
	context 3, 0.4	0,0651	0,0032	0,9467	0,045	0,0026	0,9471
	context 4, 0.4	0,0597	0,0035	0,9462	0,0435	0,0026	0,9466
4) 1 dataset:svm 0,07 rule difference	OVR without context	0,2321	0,0069	0,9483	0,1976	0,003	0,9502
	context 1, 0.6	0,217	0,0058	0,9495	0,1846	0,0024	0,951
	context 2, 0,6	0,2097	0,0056	0,9497	0,0022	0,9513	0,9513
	context 3, 0.6	0,2119	0,0052	0,9499	0,1813	0,002	0,9512
	context 4, 0,6	0,2109	0,0052	0,9499	0,1803	0,0021	0,9513
	context 1, 0.4	0,2076	0,0053	0,9496	0,1784	0,0021	0,9512
	context 2, 0,4	0,199	0,0043	0,9504	0,1738	0,0015	0,9519
	context 3, 0.4	0,1957	0,004	0,9504	0,1677	0,0015	0,9516
	context 4, 0.4	0,1832	0,0039	0,9504	0,158	0,002	0,9514

Table B.2: OVO and OVR methods with context

OVO		0.75 quantile			median		
		TRR	FRR	NRA	TRR	FRR	NRA
3) 1dataset:logreg 0,08 rule difference	OVO without context	0,1144	0,0099	0,9418	0,0636	0,0069	0,9432
	context 1, 0.6	0,0961	0,0078	0,9437	0,0515	0,0055	0,9448
	context 2, 0,6	0,0842	0,0072	0,9442	0,0453	0,0047	0,9453
	context 3, 0.6	0,0838	0,0076	0,9439	0,0446	0,0052	0,9451
	context 4, 0,6	0,0852	0,0079	0,9436	0,0453	0,0055	0,9448
	context 1, 0.4	0,0723	0,0056	0,9452	0,0313	0,0036	0,9461
	context 2, 0,4	0,0547	0,0047	0,946	0,0237	0,0031	0,9467
	context 3, 0.4	0,0496	0,0047	0,9459	0,0227	0,0031	0,9466
	context 4, 0.4	0,04965	0,0053	0,9453	0,0205	0,0037	0,9459
4) 1dataset:svm 0,07 rule difference	OVO without context	0,2749	0,0096	0,9463	0,1971	0,0082	0,9474
	context 1, 0.6	0,2548	0,0079	0,9478	0,1777	0,0067	0,9487
	context 2, 0,6	0,2497	0,0072	0,9485	0,1724	0,006	0,9493
	context 3, 0.6	0,2548	0,007	0,9487	0,176	0,0057	0,9494
	context 4, 0,6	0,254	0,0069	0,9467	0,1778	0,0058	0,9494
	context 1, 0.4	0,2425	0,0065	0,9487	0,1688	0,0056	0,9494
	context 2, 0,4	0,2349	0,0053	0,9499	0,1555	0,0045	0,9503
	context 3, 0.4	0,2353	0,0049	0,95	0,1483	0,0041	0,9505
	context 4, 0.4	0,2231	0,0051	0,9496	0,1385	0,0043	0,9502

B. Table of SVM results. *context: height of the subimage, threshold of context*

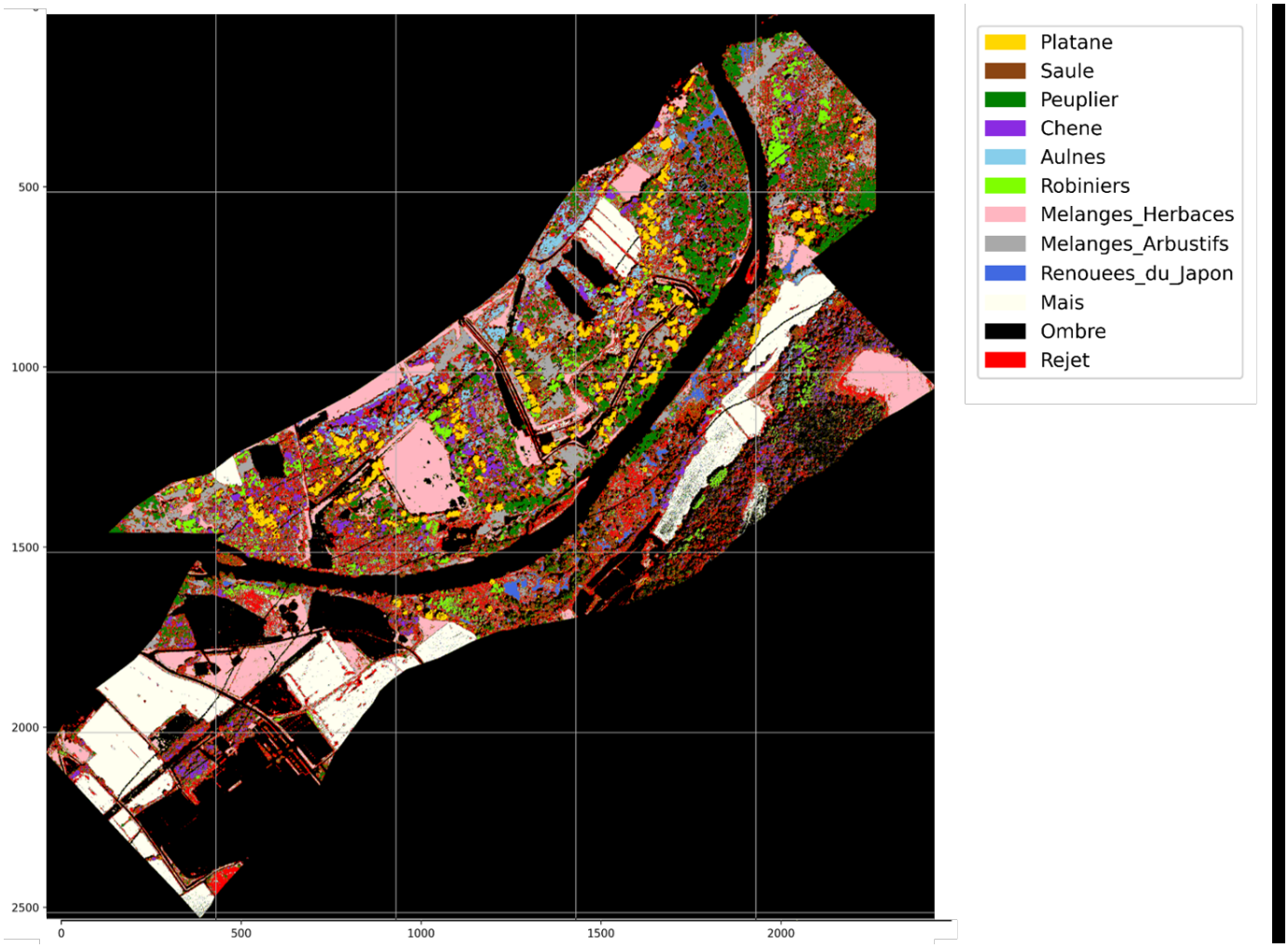


Figure B.1: Rejection by SVM_OVR_0.75 quartile 0.1 diff on svm/logreg

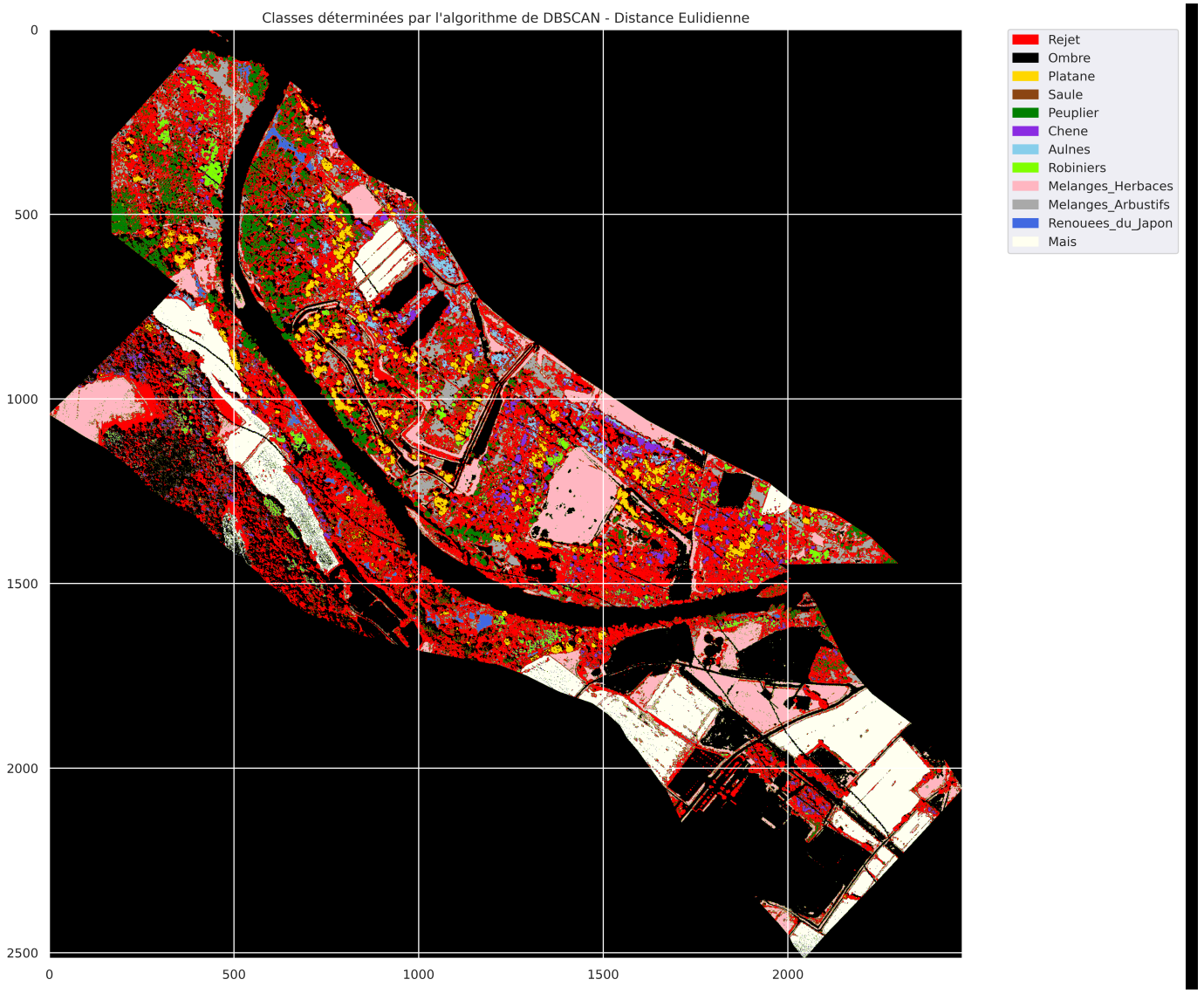


Figure C.1: Rejection by DBSCAN on the large dataset with the Euclidean distance

C. Results of DBSCAN on the large dataset.

[Ahmed and Mahmood, 2013] Ahmed, M. and Mahmood, A. N. (2013). A novel approach for outlier detection and clustering improvement. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pages 577–582.

[Besse, 2022] Besse, P. (2021-2022). *Sciences des Données 2 - Exploration Statistique Multidimensionnelle*.

[Chow, 1957] Chow, C. K. (1957). An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6(4):247–254.

[Condessa et al., 2015] Condessa, F., Bioucas-Dias, J., and Kovacevic, J. (2015). Supervised hyperspectral image classification with rejection. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2600–2603.

[Condessa et al., 2017] Condessa, F., Bioucas-Dias, J., and Kovačević, J. (2017). Performance measures for classification systems with rejection. *Pattern Recognition*, 63:437–450.

[Deng et al., 2019] Deng, Y., Tang, F., Dong, W., Wu, F., Deussen, O., and Xu, C. (2019). Selective clustering for representative paintings selection. *Multimedia Tools and Applications*, 78(14):19305–19323.

[Djeffal, 2012] Djeffal, A. (2012). *Utilisation des méthodes Support Vector Machine (SVM) dans l'analyse des bases de données*. PhD thesis.

[El-Yaniv and Wiener, 2010] El-Yaniv, R. and Wiener, Y. (2010). On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641.

[García-García et al., 2012] García-García, D., Santos-Rodríguez, R., and Parrado-Hernández, E. (2012). Classifier-based affinities for clustering sets of vectors. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6.

[Givens and Shortt, 1984] Givens, C. R. and Shortt, R. M. (1984). A class of wasserstein metrics for probability distributions. *Michigan Mathematical Journal*, 31(2):231–240.

[Grandvalet et al., 2008] Grandvalet, Y., Rakotomamonjy, A., Keshet, J., and Canu, S. (2008). Svm with a reject option.

[Guichard et al., 2010] Guichard, L., Toselli, A. H., and Coüasnon, B. (2010). Un nouveau système indépendant de rejet multi-seuils pour la reconnaissance de mots manuscrits. In *RFIA*, pages 471–478, Caen, France.

[Laroui et al., 2021] Laroui, S., Descombes, X., Vernay, A., Villiers, F., Villalba, F., and Debreuve, E. (2021). How to define a rejection class based on model learning? In *ICPR 2020 - 25th International Conference on Pattern Recognition*, Milan / Virtual, Italy.

[Lourenço, 2021] Lourenço, P. (2021). Biomass estimation using satellite-based data.

[Martin Ester, 1996] Martin Ester, Hans-Peter Kriegel, J. S. X. X. (1996). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*.

[Vo-Van and Pham-Gia, 2010] Vo-Van, T. and Pham-Gia, T. (2010). Clustering probability distributions. *Journal of Applied Statistics*, 37:1891–1910.