

F

2025



# TÉLÉDÉTECTION APPROFONDISSEMENT

**901\_21**

NAVARRO LEO  
BIOU ROMAIN  
SALA MATHIEU

## **CONTEXTE :**

La BD Forêt® version 2.0 est une base de données géographique de référence dédiée à l'espace forestier et aux milieux semi-naturels. Elle offre une description détaillée des formations végétales forestières et naturelles en se basant sur des caractéristiques telles que la densité de couvert, la composition des peuplements et l'essence dominante, pour des zones d'au moins 5 000 m<sup>2</sup> (0,5 hectare). Élaborée par photo-interprétation d'images en infrarouge couleur issues de la BD ORTHO®, elle constitue un outil essentiel pour les acteurs des secteurs de la forêt-bois, de l'environnement, de l'aménagement du territoire et du développement durable. Ses applications sont variées, allant de la gestion des ressources et de la prévention des risques à l'évaluation environnementale et à la description des paysages et de la biodiversité.

Cependant, cette base, bien que précieuse, présente certaines limites : elle n'est pas mise à jour régulièrement et ne permet pas une cartographie des essences forestières à une échelle intra-peuplement. C'est dans ce contexte que s'inscrit votre projet, qui s'appuie sur des compétences avancées en télédétection pour surmonter ces contraintes.

## **OBJECTIFS :**

Ce projet vise à explorer le potentiel de la BD Forêt® version 2.0 comme source de données de référence pour effectuer une classification supervisée à partir de séries temporelles d'images Sentinel-2.

Les étapes principales incluront :

- Préparation des données : téléchargement et prétraitement des images Sentinel-2, puis extraction et préparation des échantillons de référence basés sur la BD Forêt.
- Réalisation d'une classification supervisée à l'échelle du pixel, suivie d'une agrégation de l'information à l'échelle des peuplements définis dans la BD Forêt.
- Évaluation de la qualité des cartes produites, tant à l'échelle des pixels qu'à celle des peuplements.

Ce travail permettra de mieux évaluer les possibilités de combiner des données satellitaires avec des référentiels existants pour produire des cartographies forestières plus précises et à jour.

## **A. DESCRIPTION DE LA MÉTHODE**

### **1. Téléchargement de la données :**

Pour notre projet, nous avons utilisé une série temporelle d'images Sentinel-2. Voici les étapes que nous avons suivies pour les télécharger :

Nous nous sommes rendus sur le site <https://catalogue.theia-land.fr/> et nous avons sélectionné les images en respectant les critères suivants :

- **Période** : entre janvier 2022 et février 2023.
- **Zone** : la tuile T31TCJ, correspondant à Toulouse et ses environs, dont l'emprise vecteur nous a été fournie.
- **Couverture nuageuse** : inférieure à 15 %, en vérifiant cette information dans les propriétés des images sur le catalogue.
- **Niveau de traitement** : images en réflectance (niveau 2A).

Nous avons téléchargé six images correspondant aux périodes suivantes :

- Une image prise en hiver.
- Quatre images réparties entre l'automne et le printemps (deux pour chaque saison).
- Une image prise en été.

Les données reste des données nécessaires, comprenant la BD Forêt, l'emprise des images Sentinel-2 (S2) et l'emprise de la zone d'étude, sont déjà disponibles dans votre environnement Onyxia, dans le dossier data.

## 2. Construire un masque à partir de la BD forêt

### Étapes réalisées :

- Importation des bibliothèques et modules : Nous importons les bibliothèques nécessaires, telles que geopandas pour la gestion des données spatiales, et ajoutons des chemins personnalisés pour accéder aux scripts et fonctions externes.
- Lecture des fichiers vectoriels : Nous chargeons les fichiers shapefile de la BD Forêt et de l'emprise de l'étude à l'aide de geopandas.
- Filtrage des classes à exclure : Nous créons une colonne binaire pour exclure certains types de végétation, en leur attribuant la valeur 0, et en conservant les autres types avec la valeur 1.
- Sauvegarde des données filtrées : Nous sauvegardons les données vectorielles filtrées dans un nouveau fichier shapefile pour utilisation ultérieure.
- Paramétrage de la rasterisation : Nous définissons les paramètres nécessaires pour rasteriser les données, tels que le fichier de sortie, la résolution spatiale, le type de données et le format.
- Rasterisation des polygones filtrés : Nous utilisons la fonction rasterize pour convertir les polygones filtrés en un fichier raster au format GeoTIFF avec les paramètres définis.

### Organisation des fichiers :

- Le fichier de sortie : **masque\_foret.tif**, a été enregistré dans le dossier results/data/img\_pretraites.
- Le script Python principal utilisé a été enregistré sous le nom **build\_mask.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

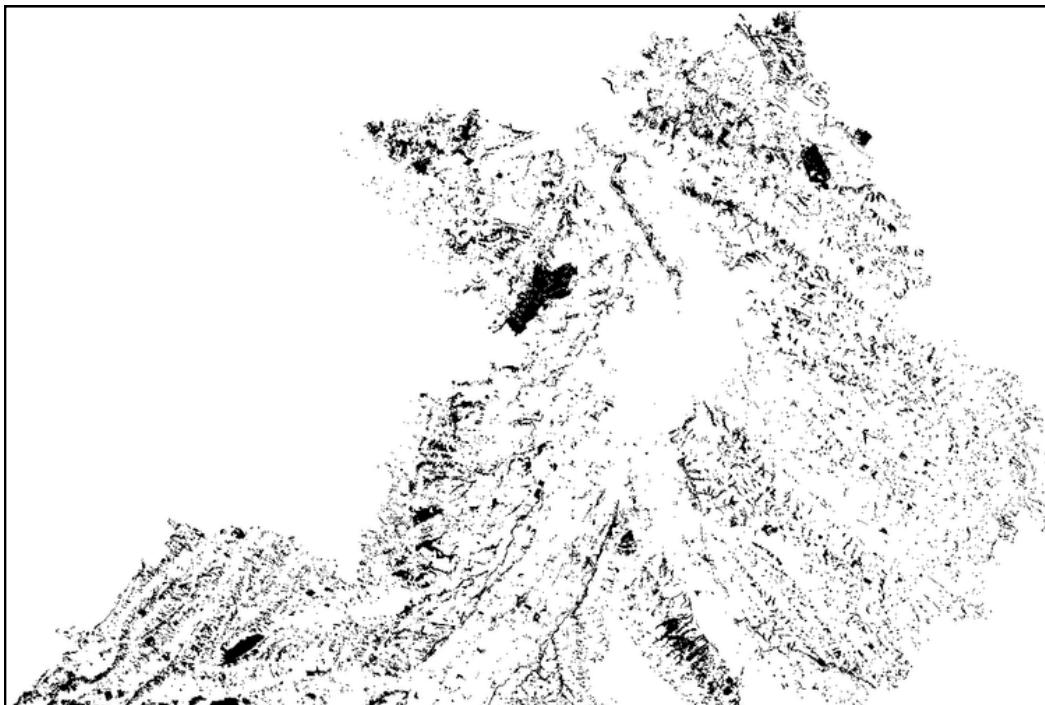


Figure 1: Masque forêt

Sur la figure ci-dessus, nous pouvons observer notre masque de forêt au format .tiff, un fichier de 80 Mo qui met en évidence les zones à classer, correspondant aux polygones de la BD Forêt. Cependant, les polygones de type "Lande" et "Formation Herbacée" ne sont pas inclus, car ils ne font pas partie de la forêt. Il convient également de ne pas prendre en compte les classes de "Forêt ouverte" ni la classe "Forêt fermée sans couvert arboré" dans l'analyse.

### 3. Pré-traitement des images :

#### Étapes réalisées :

- Initialisation des chemins et des paramètres : Nous avons d'abord défini les chemins nécessaires pour accéder aux fichiers raster, aux emprises géographiques et aux masques, ainsi que les paramètres d'output pour le stockage des résultats. Les dossiers de sortie ont été créés si nécessaire pour organiser les fichiers générés.
- Découpage des rasters : Ensuite, nous avons parcouru tous les fichiers raster présents dans le dossier des images et les avons découpés selon l'emprise géographique définie dans le fichier emprise\_etude.shp. Le découpage a été réalisé en appelant la fonction clip\_raster, avec une résolution spatiale de 10 mètres. Chaque raster découpé a été enregistré dans un dossier spécifique.
- Application du masque forêt : Pour chaque raster découpé, nous avons appliqué le masque de la forêt en utilisant la fonction apply\_mask. Cela permet de ne conserver que les pixels correspondant à la forêt, en excluant les zones hors forêt. Les rasters masqués ont été sauvegardés dans un dossier dédié.
- Concaténation des bandes spectrales : Les fichiers raster masqués ont ensuite été triés et concaténés en un seul fichier multibandes, grâce à la fonction concat\_bands. Le tri a été effectué en fonction de la date et de la bande spectrale pour garantir une organisation correcte des données. Le résultat final a été enregistré sous le nom **Serie\_temp\_S2\_allbands.tif**.
- Calcul du NDVI : Pour analyser la végétation, nous avons calculé l'indice de végétation NDVI à partir des bandes optiques (B8 et B4) des images Sentinel-2. La fonction apply\_mask a de nouveau été utilisée pour appliquer le masque sur les rasters, puis la fonction calculate\_ndvi a été utilisée pour générer les images NDVI. Ces rasters ont ensuite été enregistrés dans un dossier spécifique.
- Création du fichier NDVI final : Les rasters NDVI générés ont été triés et concaténés en un seul fichier **Serie\_temp\_S2\_ndvi.tif** grâce à la fonction concat\_bands. Ce fichier final contient les indices de végétation pour chaque image de la série temporelle, prêts à être utilisés pour les analyses suivantes.

#### Organisation des fichiers :

- Le fichier de sortie : **Serie\_temp\_S2\_ndvi.tif** / **Serie\_temp\_S2\_allbands.tif**, ont été enregistré dans le dossier results/data/img\_pretraitees.
- Le script Python principal utilisé a été enregistré sous le nom **pre\_traitement.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

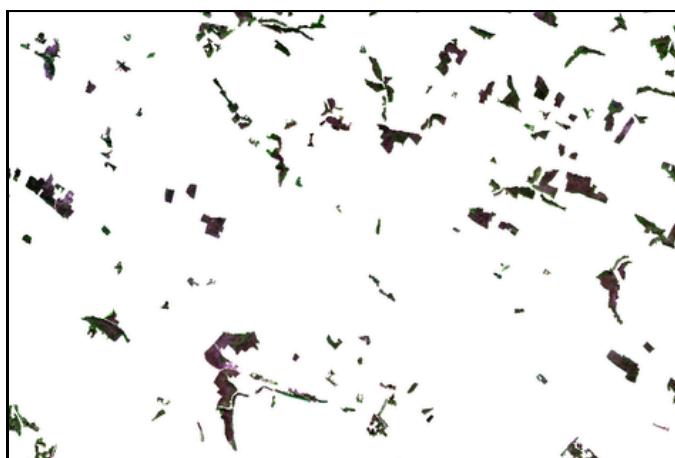


Figure 2 : Image prétraitée all\_bands

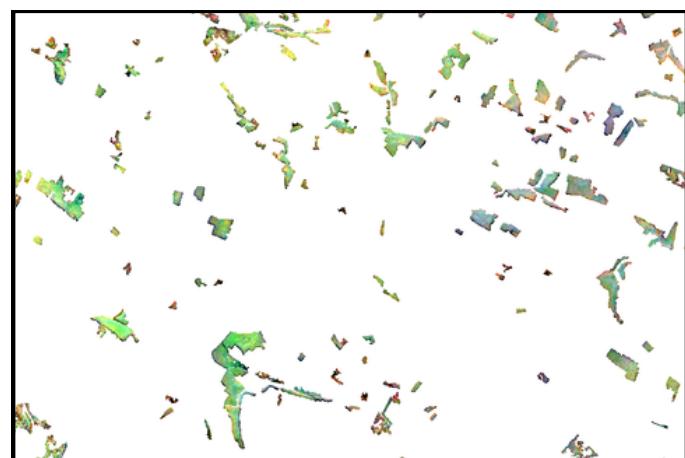


Figure 3 : Image prétraitée NDVI

Sur la figure ci-dessus, nous pouvons observer l'image prétraitée all\_bands au format .tiff, un fichier de 9 Go contenant les 10 bandes, pour les 6 dates, avec l'ensemble des paramètres demandés dans les consignes.

Sur la figure ci-dessus, nous pouvons observer l'image prétraitée NDVI au format .tiff, un fichier de 1,7 Go contenant la série temporelle de ndvi, constituée de chaque date à disposition, avec l'ensemble des paramètres demandés dans les consignes.

#### **4. Sélection des échantillons :**

##### **Étapes réalisées :**

- Importation des bibliothèques et des modules nécessaires : Nous avons importé les bibliothèques nécessaires, telles que geopandas, et ajouté les chemins pour accéder aux fonctions personnalisées à partir de notre propre script.
- Définition des chemins des fichiers de données : Nous avons défini les chemins des fichiers nécessaires, incluant la BD Forêt, l'emprise d'étude et le répertoire de sortie où nous enregistrerons les résultats.
- Lecture des données vectorielles : Nous avons chargé la BD Forêt et l'emprise d'étude à l'aide de geopandas.read\_file() pour préparer les données pour le traitement.
- Création des dictionnaires de correspondance pour les codes et les noms : Nous avons créé les dictionnaires code\_mapping et name\_mapping pour faire correspondre les codes de classe aux valeurs et aux noms des végétations dans la BD Forêt.
- Classification des données géospatiales : Nous avons classé les polygones de la BD Forêt en utilisant la fonction classify\_geodataframe, en appliquant les correspondances définies dans les dictionnaires de codes et de noms.
- Découpe de la BD Forêt selon l'emprise d'étude : Après avoir réussi la classification, nous avons découpé le GeoDataFrame classé selon l'emprise d'étude avec la fonction filter\_and\_clip\_geodata.

##### **Organisation des fichiers :**

- Le fichier de sortie : **Sample\_BD\_foret\_T31TCJ.shp**, a été enregistré dans le dossier results/data/sample.
- Le script Python principal utilisé a été enregistré sous le nom **sample\_curation.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

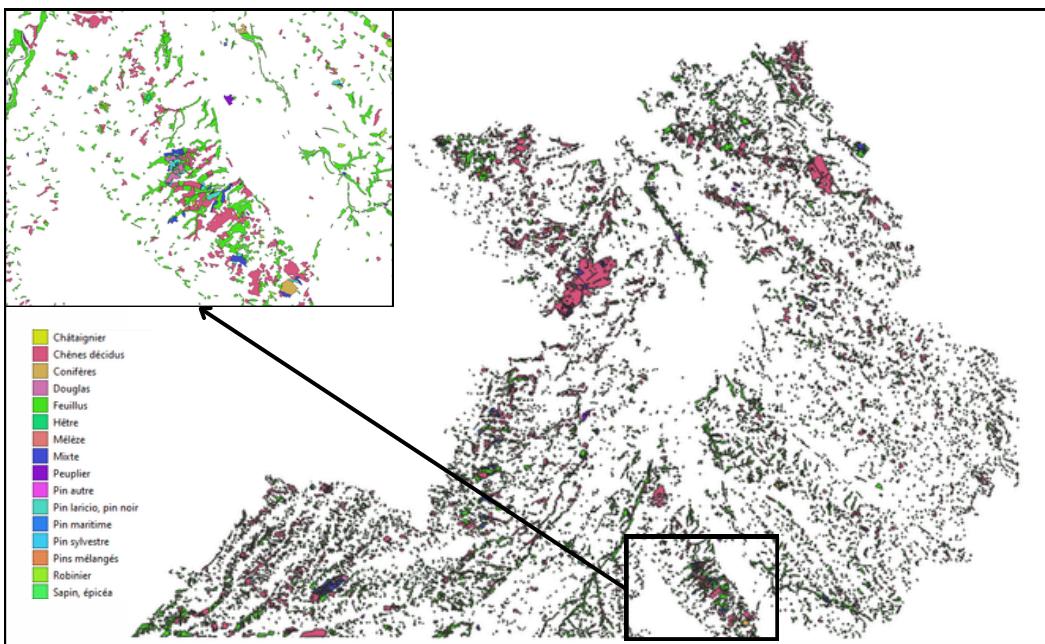


Figure 4 : Échantillonnage

Cette image représente une cartographie des peuplements forestiers, où une classification par couleur distingue les différentes espèces d'arbres ou types de végétation. La répartition apparaît relativement hétérogène, avec une prédominance de zones en vert, tandis que d'autres secteurs présentent une mosaïque de classes représentées par les couleurs rose, bleu et jaune.

## 5. Analyse des échantillons

### 5.1 Nombre d'échantillons

#### Étapes réalisées :

- Chargement des données : Nous avons chargé le fichier shapefile contenant les polygones de la BD Forêt avec GeoPandas, ce qui nous a permis d'accéder aux données nécessaires pour l'analyse.
- Comptage des polygones par classe : Nous avons calculé le nombre de polygones par classe en utilisant une fonction dédiée, en nous basant sur les codes de classe pour effectuer le comptage.
- Création des diagrammes en bâtons : Nous avons généré un diagramme en bâtons pour visualiser le nombre de polygones par classe, avec des couleurs adaptées et des valeurs ajoutées sur chaque barre pour faciliter la lecture.
- Comptage des pixels par classe : Nous avons comptabilisé le nombre de pixels par classe de la même manière que pour les polygones, afin de comparer la surface occupée par chaque classe.
- Calcul de la surface des polygones en pixels : Nous avons converti la surface de chaque polygone en pixels, en prenant en compte la résolution spatiale des images Sentinel-2 utilisées.
- Préparation des données pour le "Violin Plot" : Nous avons extrait et organisé les données relatives au nombre de pixels par polygone pour créer un "violin plot", ce qui nous a permis d'observer la distribution des valeurs.
- Création et personnalisation du "Violin Plot" : Nous avons créé et personnalisé un "violin plot" pour afficher la distribution des pixels par polygone pour chaque classe, en appliquant une échelle logarithmique sur l'axe des ordonnées.

#### Organisation des fichiers :

- Le fichier de sortie : `diag_baton_nb_poly_by_class.png` / `diag_baton_nb_pix_by_class.png` / `violin_plot_nb_pix_by_poly_by_class.png`, ont été enregistré dans le dossier `results/figure`.
- Le script Python principal utilisé a été enregistré sous le nom `sample_analysis_nb_sample.py`. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé `my_function.py`.

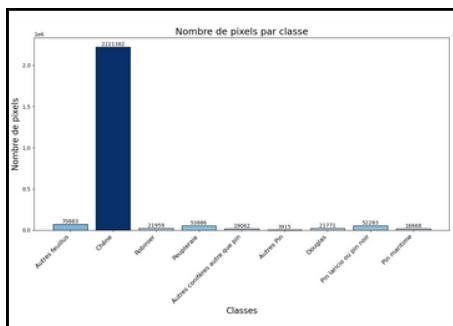


Figure 5 :  
`diag_baton_nb_pix_by_class`

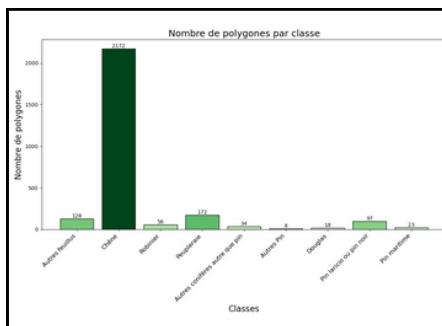


Figure 6 :  
`diag_baton_nb_poly_by_class`

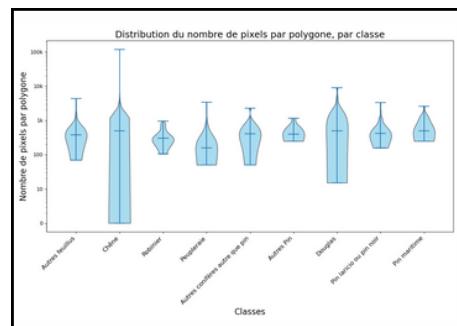


Figure 7 :  
`violin_plot_nb_pix_by_poly_by_class`

Sur le graphique ci-dessus, nous pouvons observer le nombre de pixel par classe.

Sur le graphique ci-dessus, nous pouvons observer le nombre de polygones par classe.

Sur le graphique ci-dessus, nous pouvons observer le nombre de pixels par polygone, par classe.

Grâce aux trois graphiques ci-dessus, nous pouvons clairement observer que la grande majorité des entités sont classées dans la catégorie "chêne", que ce soit en termes de pixels ou de polygones.

## 5.2 Phénologie des peuplements purs

### Étapes réalisées :

- Préparation de l'environnement : Nous avons importé les bibliothèques nécessaires et configuré le logging pour suivre l'avancement du traitement, tout en créant le dossier de sortie pour enregistrer les résultats.
- Définition des classes et initialisation des statistiques : Nous avons sélectionné les classes pertinentes de la BD Forêt et initialisé un dictionnaire pour stocker la moyenne et l'écart type du NDVI pour chaque classe.
- Traitements des bandes NDVI : Pour chaque bande temporelle du raster NDVI, nous avons extrait la bande correspondante et créé des rasters temporaires pour chaque période.
- Création des masques pour chaque classe : Nous avons créé des masques pour chaque classe forestière à partir du shapefile de la BD Forêt, puis utilisé ces masques pour découper les rasters NDVI par classe.
- Calcul des statistiques : Nous avons extrait les statistiques (moyenne et écart type) des rasters découpés en utilisant gdalinfo et avons sauvégarde ces résultats dans un dictionnaire.
- Nettoyage des fichiers temporaires : Après chaque itération, nous avons supprimé les fichiers temporaires (masques et rasters découpés) pour libérer de l'espace disque.
- Création du graphique des signatures temporelles : Nous avons créé un graphique représentant l'évolution du NDVI pour chaque classe, en affichant les courbes de moyenne et les zones d'incertitude (écart type), puis avons sauvégarde le graphique dans le dossier de sortie.
- Finalisation et sauvegarde des résultats : Nous avons sauvégarde les résultats, y compris le graphique des signatures temporelles, dans le dossier spécifié, tout en journalisant chaque étape du processus pour assurer la traçabilité des actions.

### Organisation des fichiers :

- Le fichier de sortie : **temp\_mean\_ndvi.png**, a été enregistré dans le dossier results/figure.
- Le script Python principal utilisé a été enregistré sous le nom **sample\_analysis\_temp\_signature.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

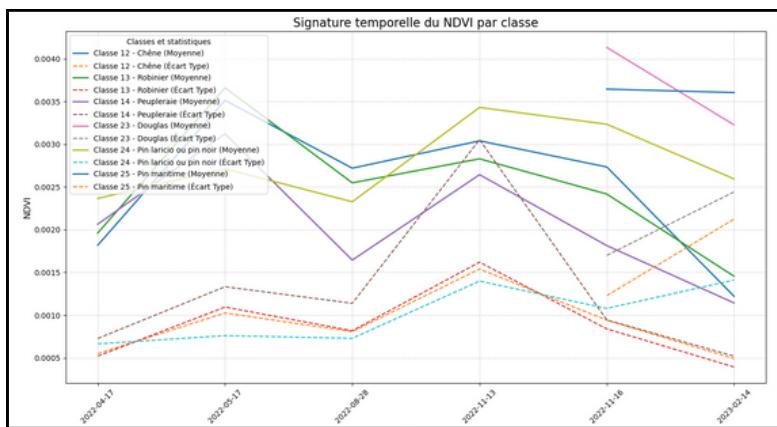


Figure 8 : temp\_mean\_ndvi

Sur la figure ci-dessus, nous pouvons observer la signature temporelle de la moyenne et de l'écart-type du NDVI par classe sur les 6 dates sélectionnées.

### Variabilité saisonnière :

Nous observons des fluctuations du NDVI, avec généralement une augmentation au printemps et en été (d'avril à novembre), suivie d'une diminution en hiver (février).

Comme nous pouvons le constater en haut à droite du graphique, il y a deux classes moyennes (23 et 25) qui n'apparaissent que durant la période de novembre à février.

### 5.3 Analyse de la variabilité spectrale de la BD forêt

#### Étapes réalisées :

- Chargement des données géographiques : Nous avons chargé les données vectorielles de la BD Forêt et le raster NDVI, en récupérant leurs propriétés spatiales nécessaires pour les traitements.
- Filtrage des classes pertinentes : Nous avons sélectionné uniquement les polygones correspondant aux classes d'intérêt, en les regroupant en classes bleues et rouges selon leur nomenclature.
- Calcul des distances moyennes au centroïde par classe : Nous avons calculé la distance moyenne entre le centroïde des polygones de chaque classe et les pixels NDVI correspondants, en tenant compte des zones valides du raster.
- Création du diagramme en bâtons : Nous avons visualisé les distances moyennes par classe à l'aide d'un diagramme en bâtons, différenciant les classes par couleur, et sauvegardé le graphique dans le dossier des résultats.
- Analyse des distances à l'échelle des polygones : Nous avons mesuré la distance moyenne entre le centroïde de chaque polygone individuel et les pixels NDVI associés, et compilé les résultats dans un tableau par classe.
- Création du violin plot : Nous avons représenté la distribution des distances moyennes par classe avec un violin plot, permettant de visualiser les variations internes, avant de sauvegarder le graphique final.
- Organisation des scripts et résultats : Nous avons centralisé nos scripts dans des fichiers dédiés et sauvegardé les graphiques et fichiers générés dans les dossiers structurés du projet.

#### Organisation des fichiers :

- Le fichier de sortie : **diag\_baton\_dist\_centroide\_classe.png** / **violin\_plot\_dist\_centroide\_by\_poly\_by\_class.png**, ont été enregistré dans le dossier results/figure.
- Le script Python principal utilisé a été enregistré sous le nom **sample\_analysis\_spectral\_variability.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

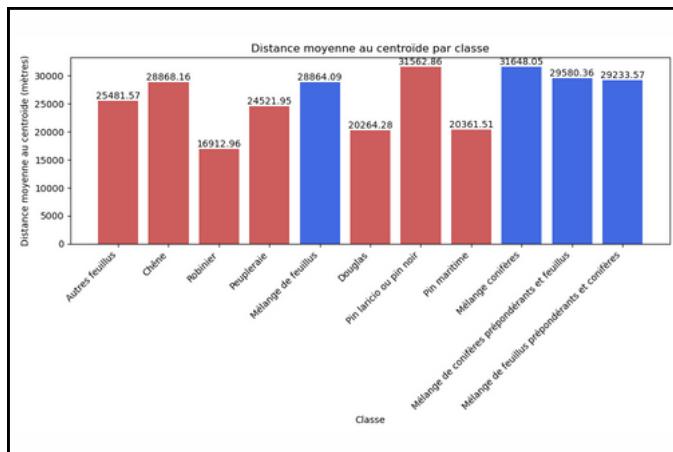


Figure 9 :  
diag\_baton\_dist\_centroide\_classe

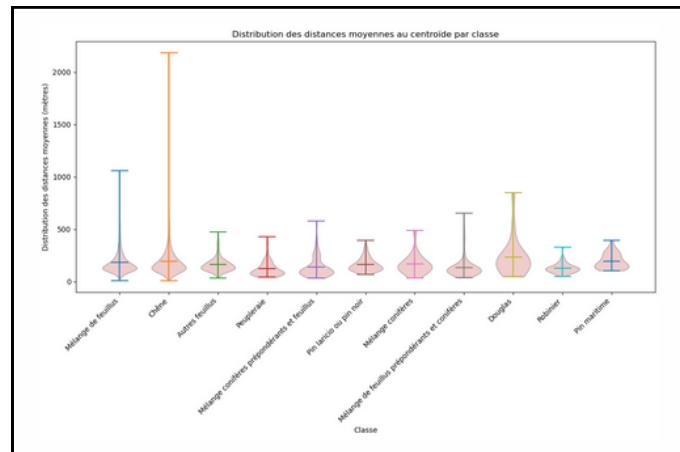


Figure 10 :  
violin\_plot\_dist\_centroide\_by\_poly\_by\_class

Sur la figure ci-dessus, nous pouvons observer la distance moyenne au centroïde pour chaque essence d'arbre présente dans le projet, pour les classes 1 (en rouge) et 2 (en bleu).

Sur la figure ci-dessus, nous pouvons observer la distance moyenne des pixels au centroïde de leur polygone pour chaque essence d'arbre présente dans le projet.

## 6. Production des essences forestières à l'échelle du pixel

### 6.1 Choix du classifieur et sa paramétrisation / 6.2 Stratégie de validation / 6.3 Production des cartes finales

#### **Étapes réalisées :**

- Lecture et filtrage des données : Nous avons lu les données d'échantillons forestiers et conservé uniquement les classes pertinentes pour la classification.
- Rastérisation des échantillons : Nous avons converti les données vectorielles filtrées en raster en utilisant une emprise d'étude et une résolution spatiale de 10 mètres.
- Extraction des échantillons d'apprentissage : Nous avons récupéré les valeurs des pixels et leurs classes associées à partir des images rastérisées et prétraitées.
- Mise en place de la validation croisée : Nous avons défini une validation croisée stratifiée à 5 plis pour évaluer la performance du modèle.
- Entraînement du modèle de classification : Nous avons entraîné un modèle Random Forest en utilisant une partie des échantillons et en ajustant ses hyperparamètres.
- Évaluation du modèle : Nous avons testé le modèle sur les données restantes et calculé des métriques de classification, telles que l'accuracy et la matrice de confusion.
- Moyennage des résultats : Nous avons agrégé les performances des différentes itérations pour obtenir une évaluation globale du modèle.
- Visualisation et sauvegarde des résultats : Nous avons généré et enregistré des graphiques de qualité de classification et des matrices de confusion.
- Application du modèle sur l'image complète : Nous avons appliqué le modèle entraîné sur l'ensemble de l'image prétraitée pour produire une carte de classification des essences forestières.

#### **Organisation des fichiers :**

- Le fichier de sortie : **carte\_essences\_ecelle\_pixel.tif**, a été enregistré dans le dossier results/data/classif.
- Le script Python principal utilisé a été enregistré sous le nom **classification\_pixel.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

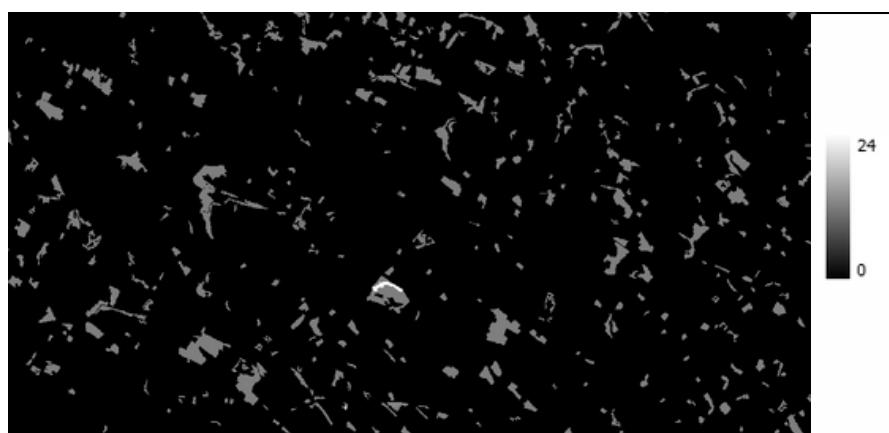


Figure 11 : Essences forestières à l'échelle du pixel

Cette image représente une classification des essences forestières à l'échelle du pixel, avec une gamme de valeurs allant de 0 à 24. Les zones en noir correspondent à la nodata, tandis que les tons de gris indiquent différentes essences forestières.

## 6.4 Production d'une carte à l'échelle des peuplements

### Objectifs :

Classer les zones d'intérêt en appliquant plusieurs filtres et paramètres afin d'obtenir une classification précise selon le schéma disponible si dessous.

### Étapes réalisées :

- Nous avons chargé les fichiers d'entrée, incluant un raster de classification et un shapefile contenant des échantillons de forêts.
- Nous avons lu le shapefile avec geopandas et extrait la correspondance entre les codes et les noms des essences forestières.
- Nous avons calculé la surface de chaque polygone en hectares pour affiner l'analyse des données.
- Nous avons effectué une analyse zonale en croisant les polygones du shapefile avec le raster afin d'extraire des statistiques sur l'occupation du sol.
- Nous avons attribué une classe prédite à chaque polygone en utilisant la fonction `classify_polygon`.
- Nous avons supprimé la colonne de surface et réécrit le shapefile mis à jour avec la nouvelle colonne `coderedit`.
- Nous avons généré une matrice de confusion pour comparer les valeurs réelles et les valeurs prédites, en excluant les données invalides.
- Nous avons affiché et sauvegardé la matrice de confusion sous forme d'image pour évaluer les performances du modèle de classification

### Organisation des fichiers :

- Le fichier de sortie : **Sample\_BD\_foret\_T31TCJ.shp**, a été enregistré dans le dossier `results/data/classif`.
- Le script Python principal utilisé a été enregistré sous le nom **classification\_stand.py**. Les autres fonctions auxiliaires ont été regroupées dans un fichier séparé, nommé **my\_function.py**.

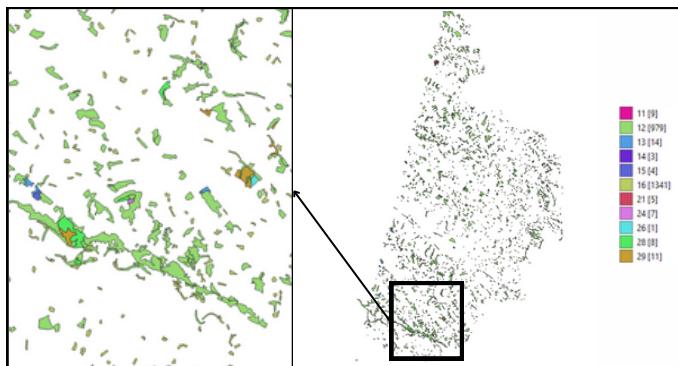


Figure 12 : Classification des polygones

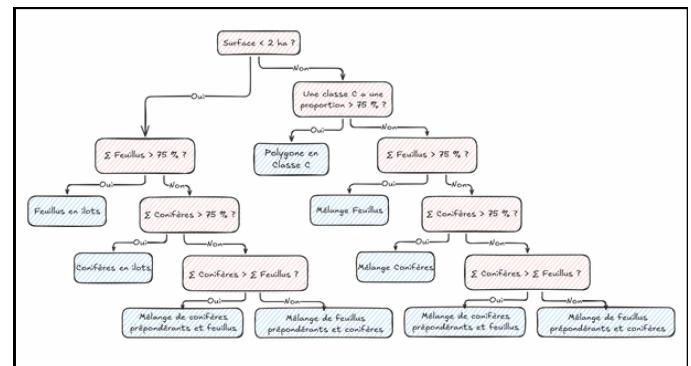


Figure 13 : Diagramme des consignes

Cette image illustre la classification de nos polygones en fonction des essences forestières. Leur répartition est organisée selon le diagramme de flux présenté ci-dessus dans l'énoncé du projet.

## 7. Analyse des résultats

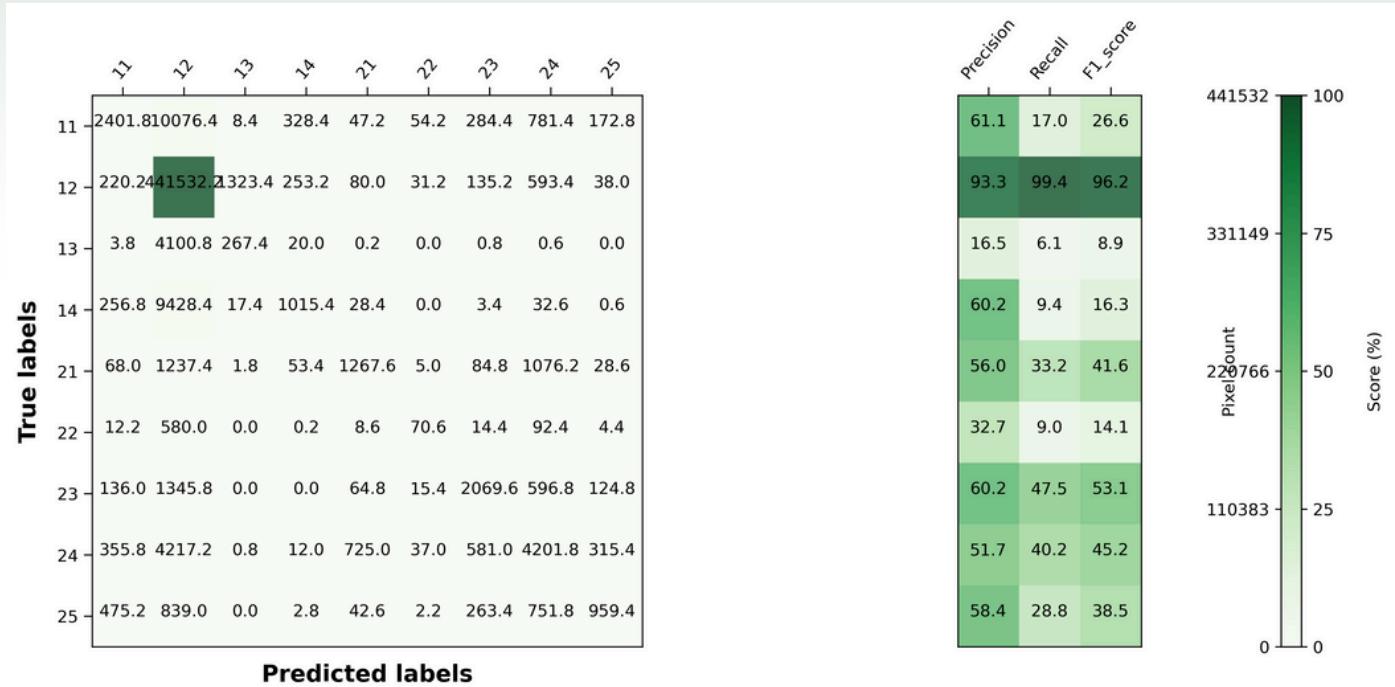


Figure 14 : Matrice de confusion

### 1. Qualité globale du modèle de classification

- L'évaluation de la qualité globale du modèle repose sur les valeurs de précision, rappel et F1-score affichées sur la matrice de droite.
- Certaines classes présentent des scores élevés (exemple : classe "12" avec une précision de 93.3% et un rappel de 99.4%, ce qui indique qu'elle est bien détectée).
- Cependant, d'autres classes ont des scores beaucoup plus faibles (exemple : classe "13" avec un rappel de 6.1% et un F1-score de 8.9%), ce qui signifie que cette classe est rarement identifiée correctement.
- L'hétérogénéité des scores F1 montre que certaines classes sont bien reconnues, tandis que d'autres posent problème.

### 2. Qualité des classes

- Classes bien classées :
  - La classe "12" a un excellent score de classification avec une précision de 93.3% et un rappel de 99.4%. Cela signifie que presque tous les pixels de cette classe sont correctement détectés, avec très peu de confusion.
  - La classe "23" affiche un F1-score relativement élevé de 53.1%, ce qui suggère une reconnaissance modérée.
- Classes mal classées :
  - La classe "13" est mal détectée avec un faible rappel (6.1%) et un F1-score de 8.9%. Elle est souvent mal classée, indiquant une forte confusion avec d'autres classes.
  - La classe "22" a aussi des performances médiocres (précision de 32.7%, rappel de 9.0%, F1-score de 14.1%), ce qui signifie que la majorité des pixels de cette classe sont mal attribués.

### Conclusion

- Le modèle fonctionne bien pour certaines classes (notamment la classe "12"), mais d'autres sont nettement moins bien classées, ce qui impacte la performance globale.
- Les confusions observées peuvent être logiques si certaines classes sont visuellement ou spectralement proches.
- Pour améliorer le modèle, il faudrait peut-être rééquilibrer les classes, affiner les caractéristiques utilisées pour la classification ou appliquer une post-correction basée sur des connaissances a priori.

## 8. Discussion

### 8.1. Pourquoi ces confusions ?

Plusieurs raisons peuvent expliquer les confusions observées :

- a) Similarités spectrales ou contextuelles entre classes

Certaines classes peuvent partager des signatures spectrales proches, ce qui complique leur distinction par le modèle. Par exemple :

- La forte confusion entre la classe "13" et "12" peut indiquer qu'elles ont des caractéristiques similaires en termes de texture, couleur ou localisation.
- La confusion entre "24", "23" et "25" suggère que ces classes se ressemblent, peut-être en raison d'un gradient de transition entre elles.

- b) Biais dans la répartition des données d'entraînement

- Si certaines classes sont sous-représentées dans l'échantillon d'apprentissage, le modèle peut avoir du mal à les reconnaître correctement. Par exemple, la classe "13" a un rappel très faible (6.1%), ce qui pourrait signifier que peu de pixels de cette classe étaient présents dans l'ensemble d'entraînement, ou qu'ils étaient trop variés pour être bien appris.

- c) Erreurs d'annotation des données d'entraînement

Si les données de référence contiennent des erreurs ou des ambiguïtés dans l'étiquetage des pixels, le modèle va apprendre sur des bases incorrectes. Ceci peut expliquer certaines confusions non intuitives.

### 8.2. Ces résultats sont-ils satisfaisants ?

La réponse dépend de l'objectif de la classification et du contexte d'utilisation.

**Points positifs :**

- Certaines classes, comme "12", sont très bien détectées (précision et rappel élevés).
- Des classes majeures restent exploitables malgré certaines confusions.

**Points préoccupants :**

- La mauvaise reconnaissance de certaines classes importantes ("13", "22", "14") peut limiter l'usage des résultats dans certaines applications.
- Les confusions entre classes proches peuvent réduire la fiabilité des analyses basées sur cette classification.

Si l'objectif est une analyse globale avec des classes larges, la classification reste exploitable. Si l'enjeu est de discriminer finement entre certaines classes, ces résultats posent problème.

### 8.3. Limites méthodologiques et des données :

- Déséquilibre des classes

Le fait que certaines classes aient des scores très faibles confirme qu'elles sont sous-représentées dans les données d'entraînement. Une amélioration pourrait être apportée par un rééquilibrage des données ou une augmentation artificielle des exemples des classes minoritaires.

- Utilisation d'autres sources de données plus complète.

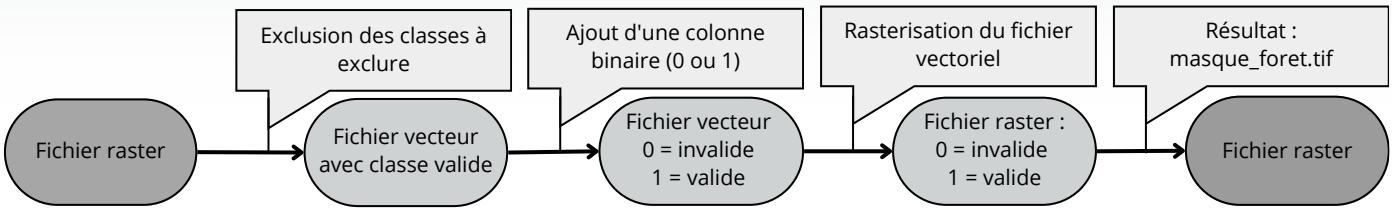
### 8.4. Interprétation des résultats :

- Si l'objectif est la cartographie générale, alors la classification peut être considérée comme raisonnablement exploitable, malgré certaines confusions.
- Si une distinction fine entre certaines classes est cruciale, ces résultats doivent être améliorés avant d'être utilisés.
- Dans un contexte de suivi temporel, ces erreurs pourraient être amplifiées, rendant les analyses évolutives plus incertaines.

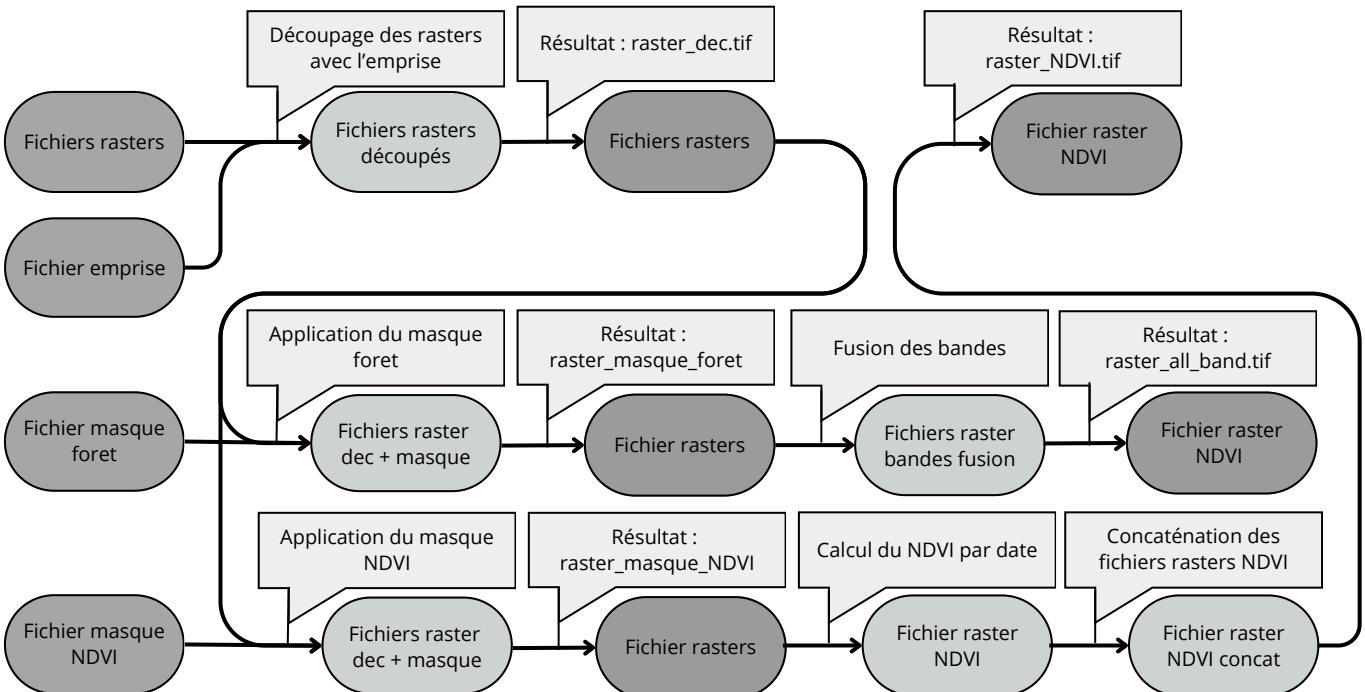
## ANNEXES ET DIAGRAMMES DE FLUX

### Diagramme de flux

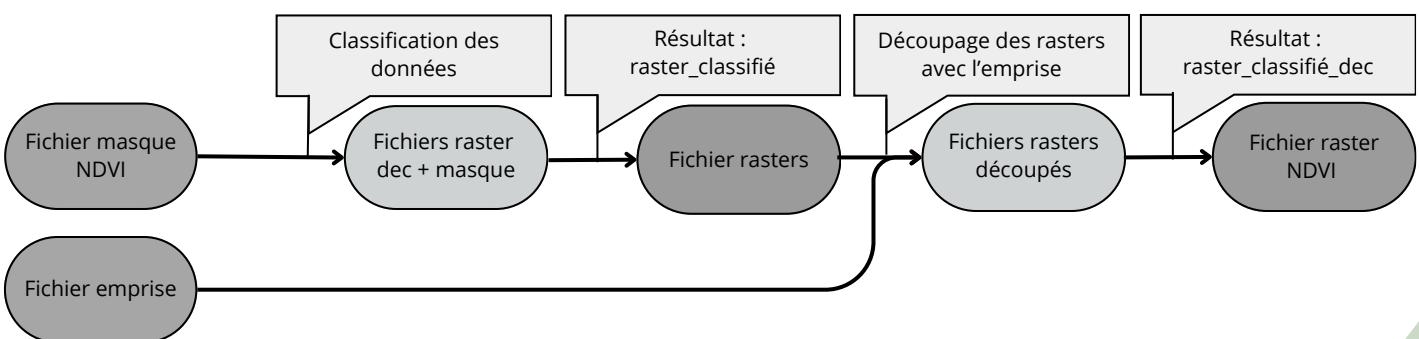
#### Masques :



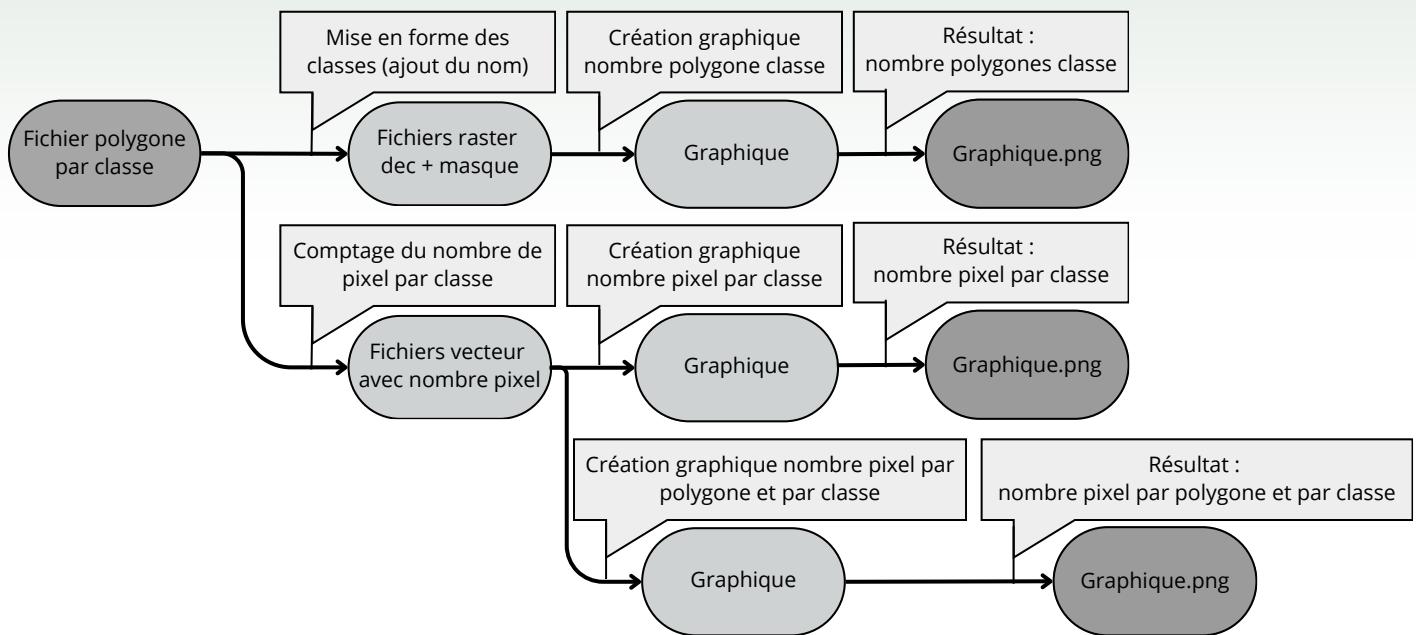
#### Prétraitements :



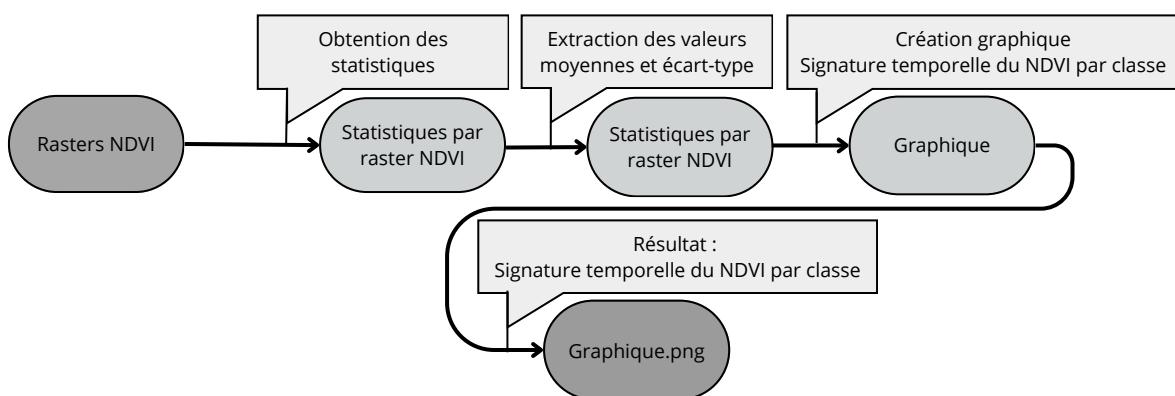
#### Sélection des échantillons :



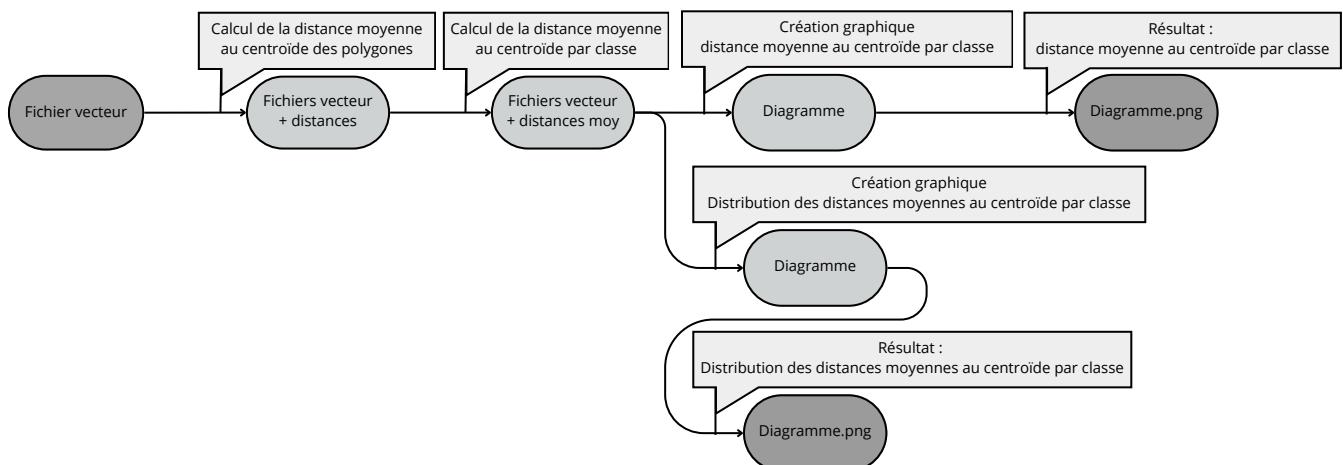
## Nombre d'échantillons :



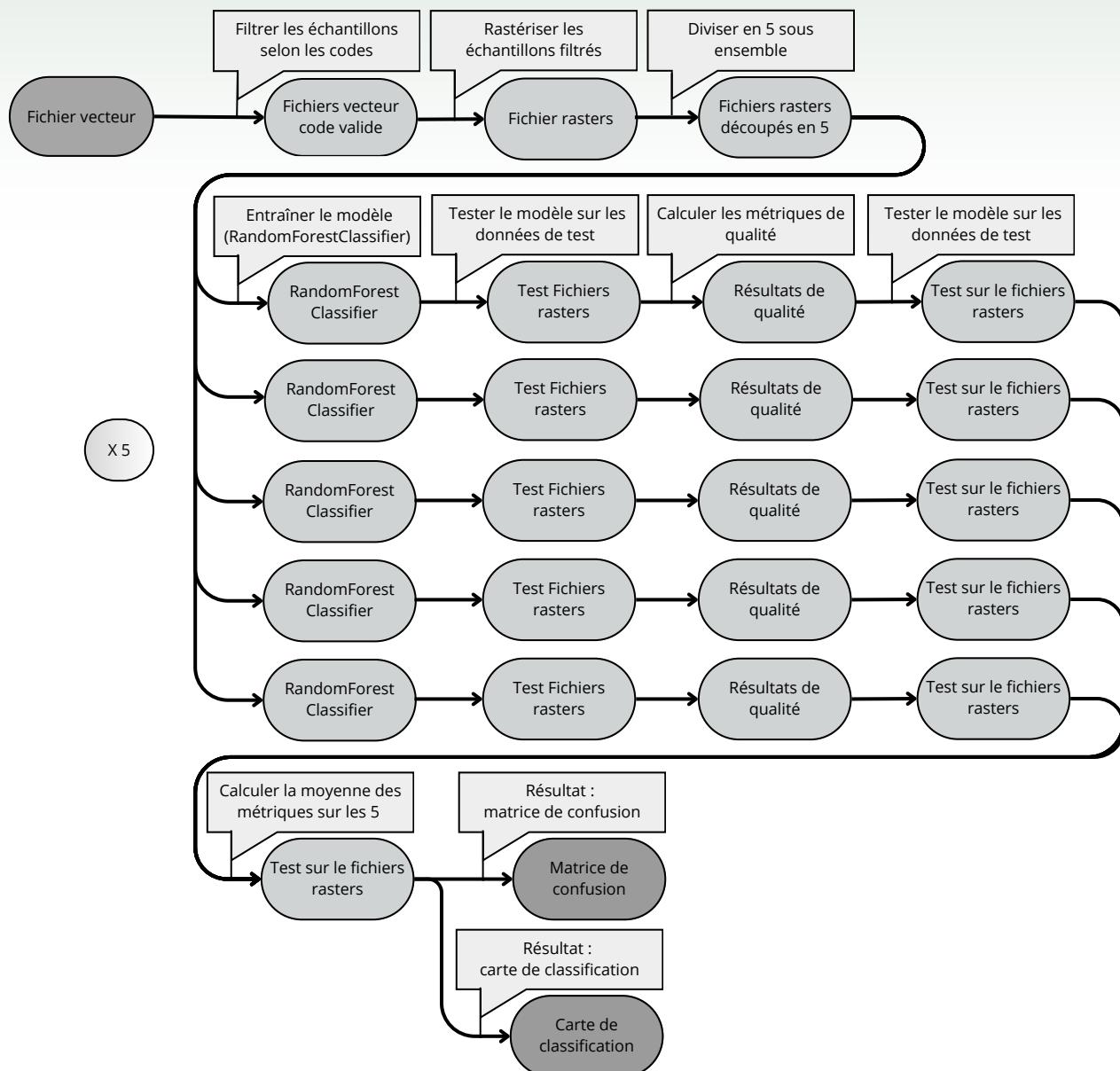
## Phénologie des peuplements purs



## Analyse de la variabilité spectrale de la BD forêt

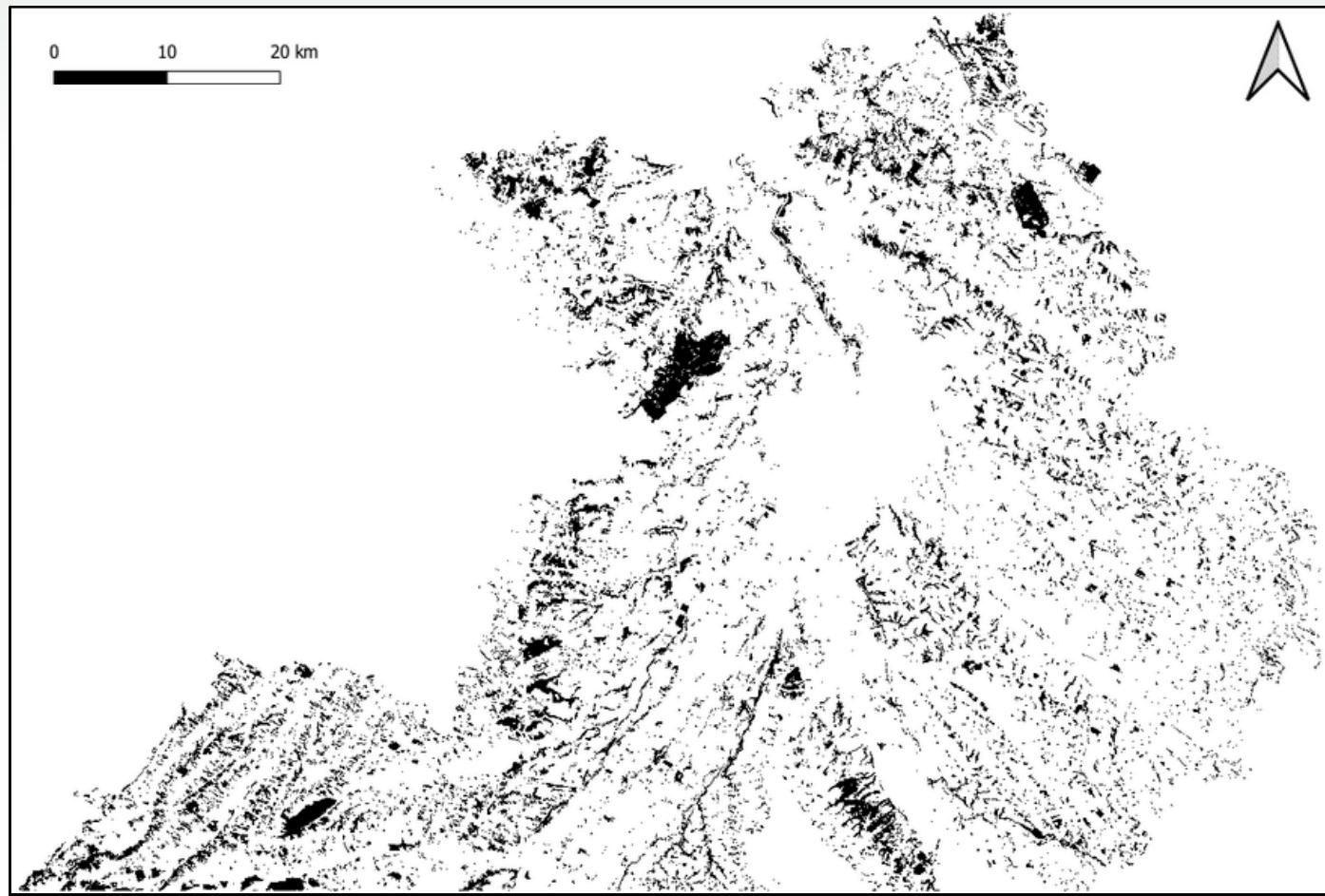


## Essences forestières à l'échelle du pixels / peuplements :



## Annexes

**Figure 1 : Masque forêt**



**Figure 2 : Image prétraitée all\_bands**

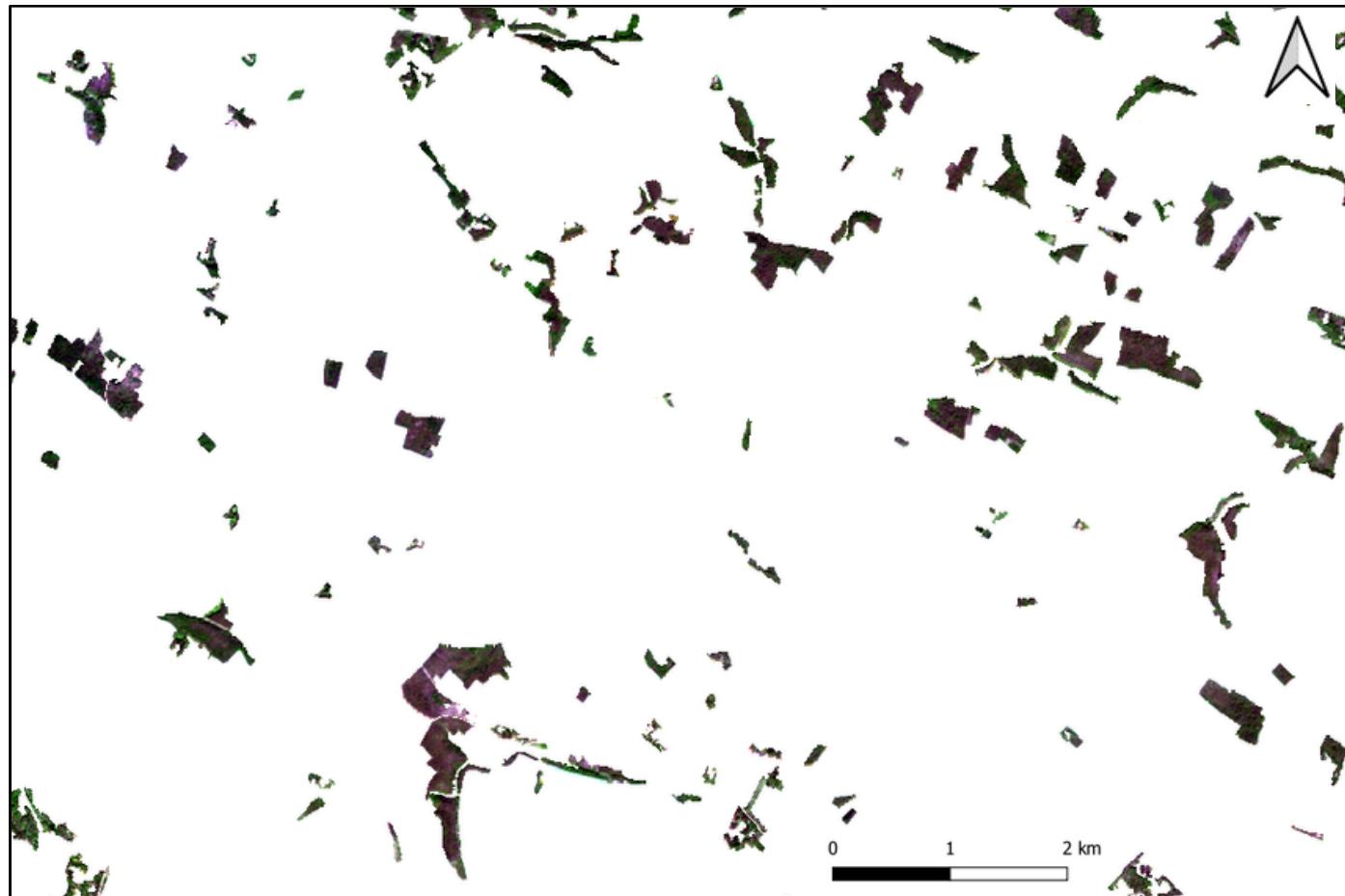


Figure 3 : Image prétraitée NDVI

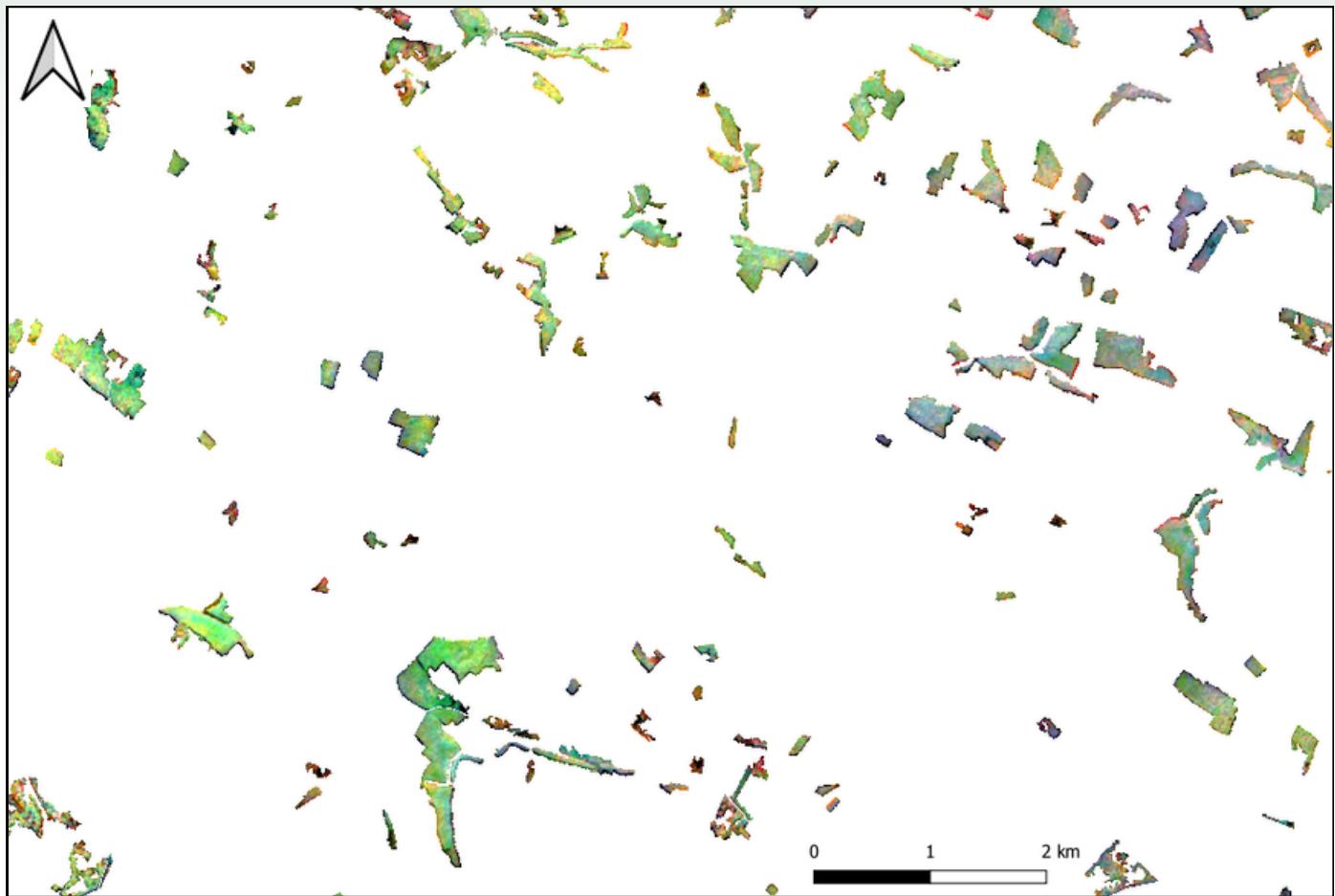


Figure 4 : Échantillonnage

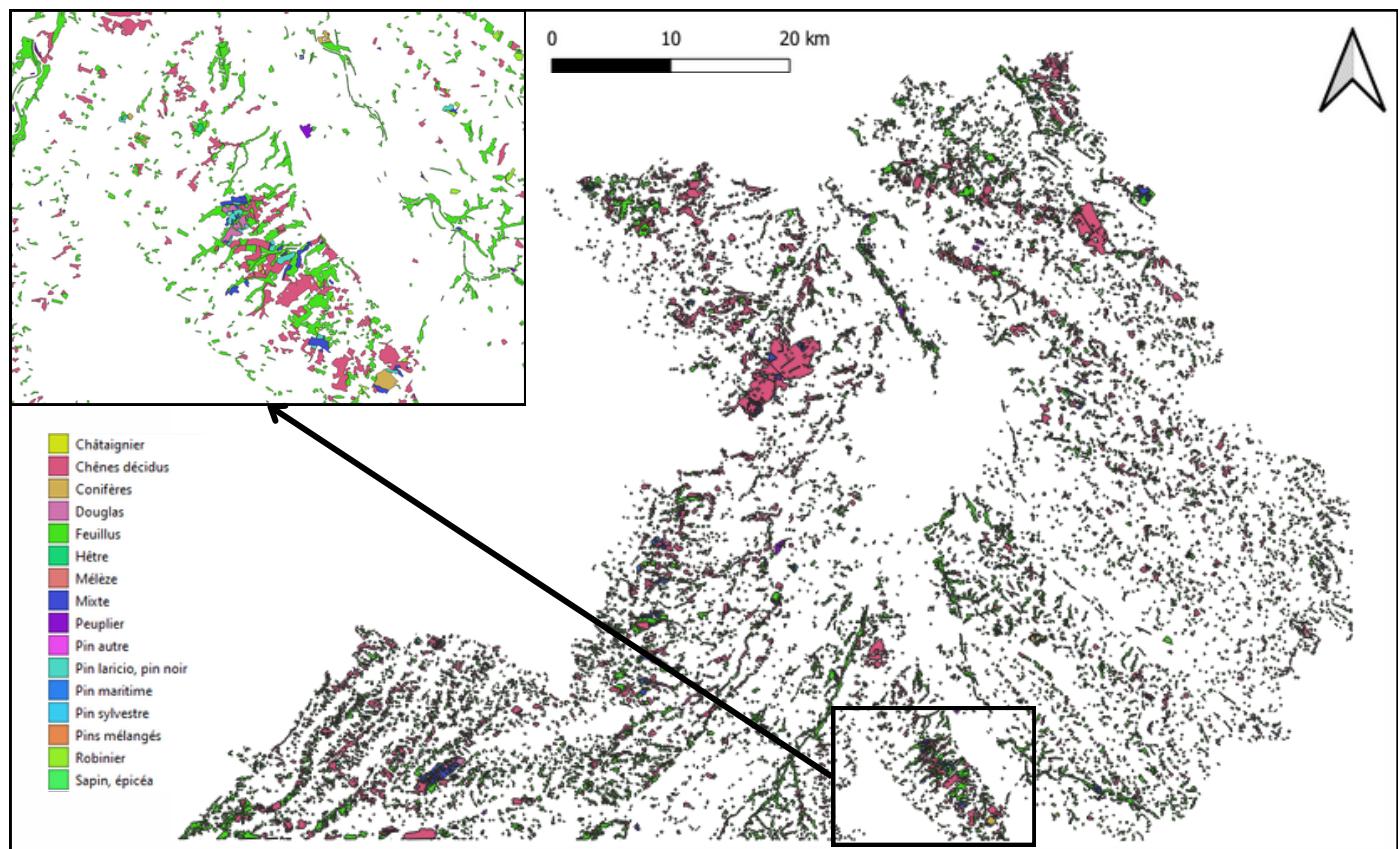


Figure 5 : diag\_baton\_nb\_pix\_by\_class

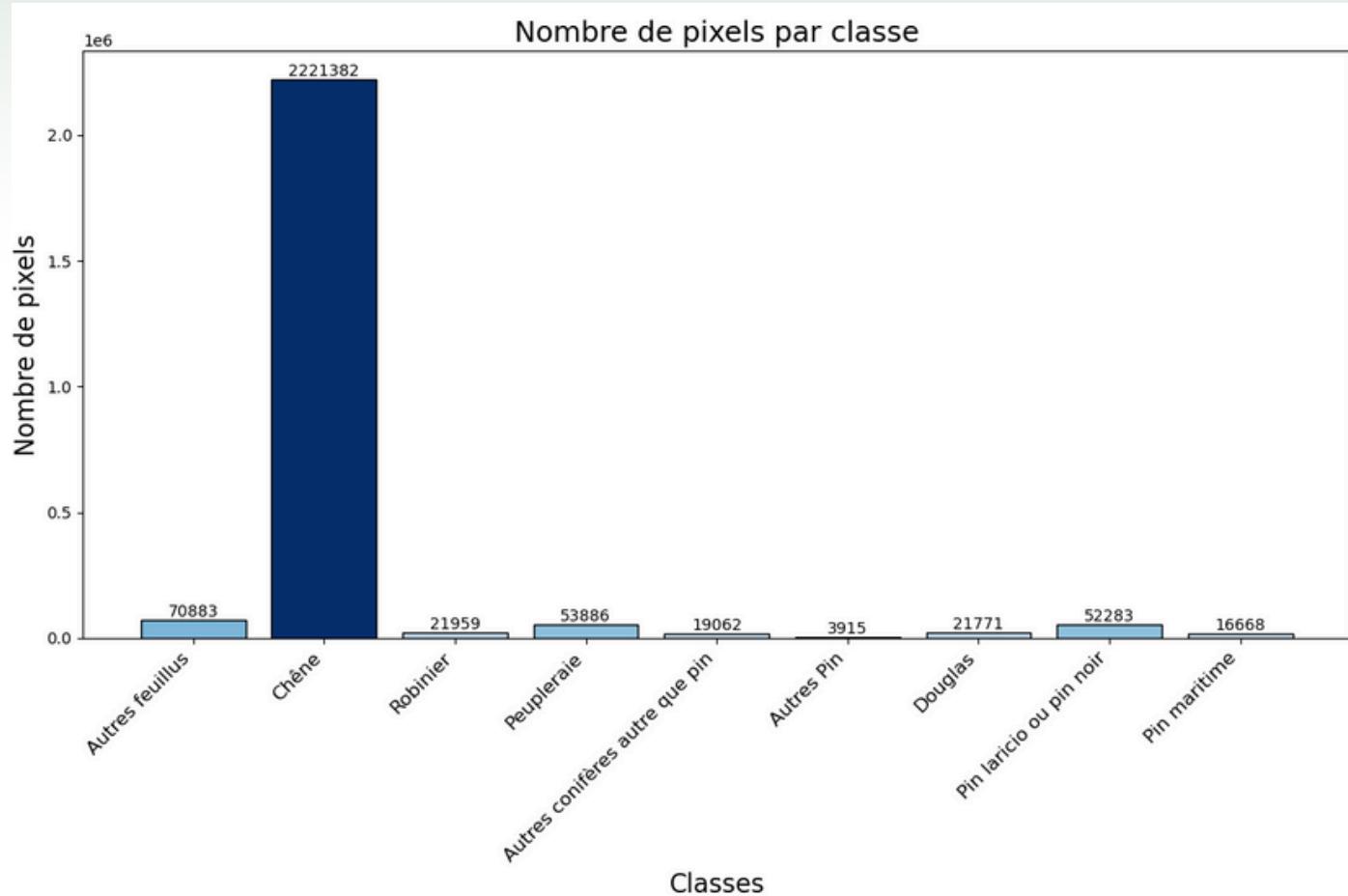


Figure 6 : diag\_baton\_nb\_poly\_by\_class

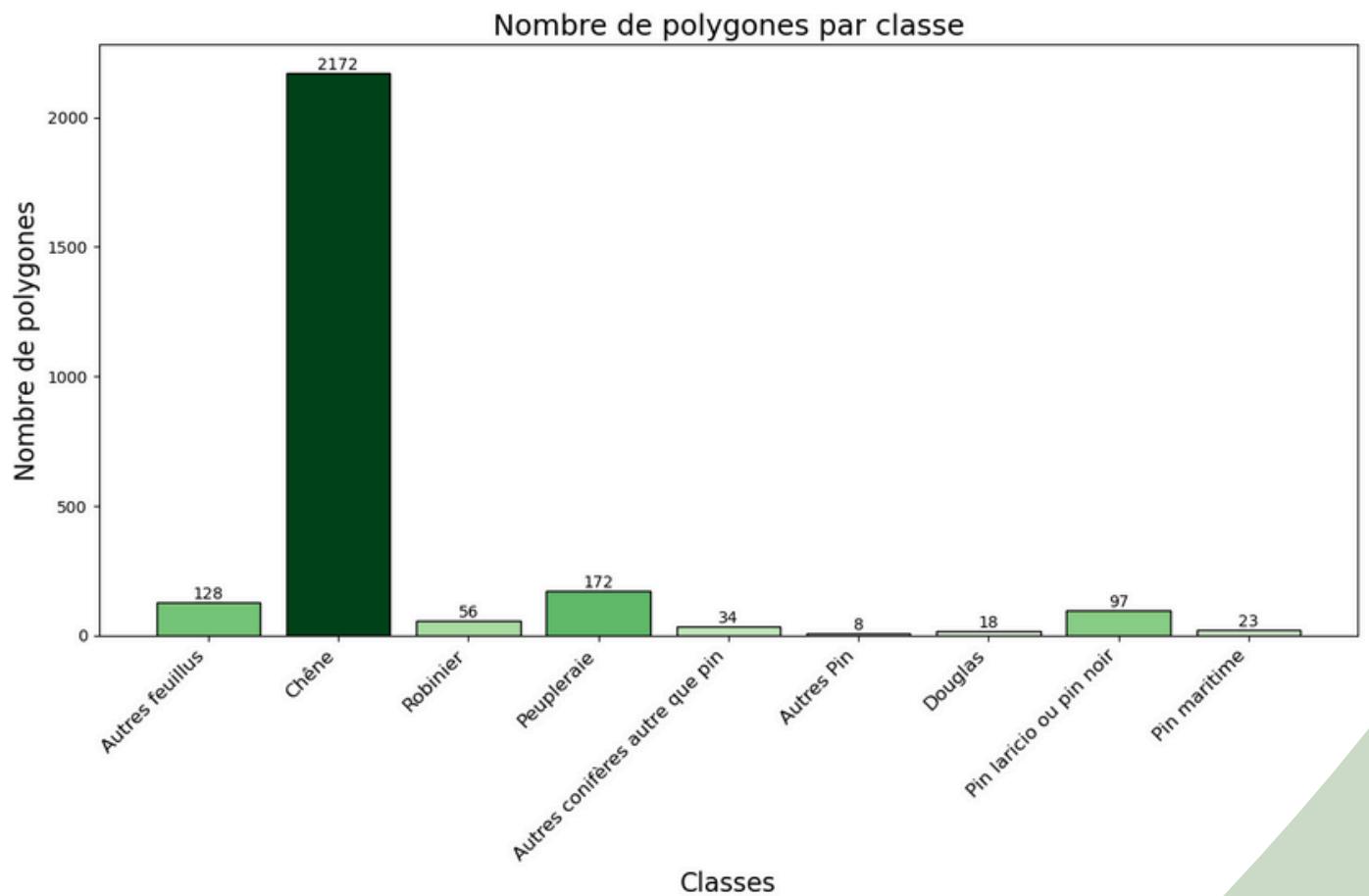


Figure 7 : violin\_plot\_nb\_pix\_by\_poly\_by\_class

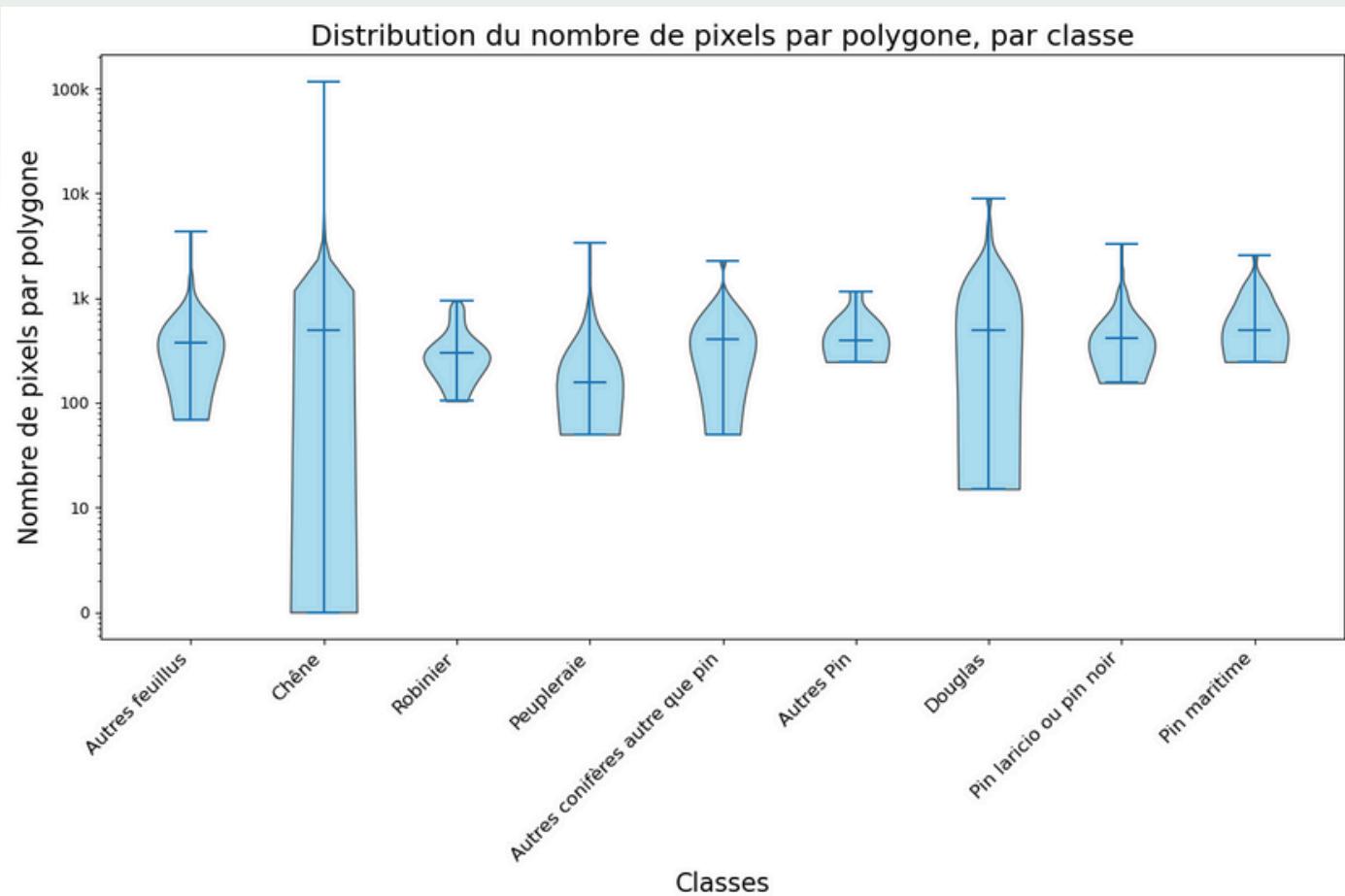


Figure 8 : temp\_mean\_ndvi

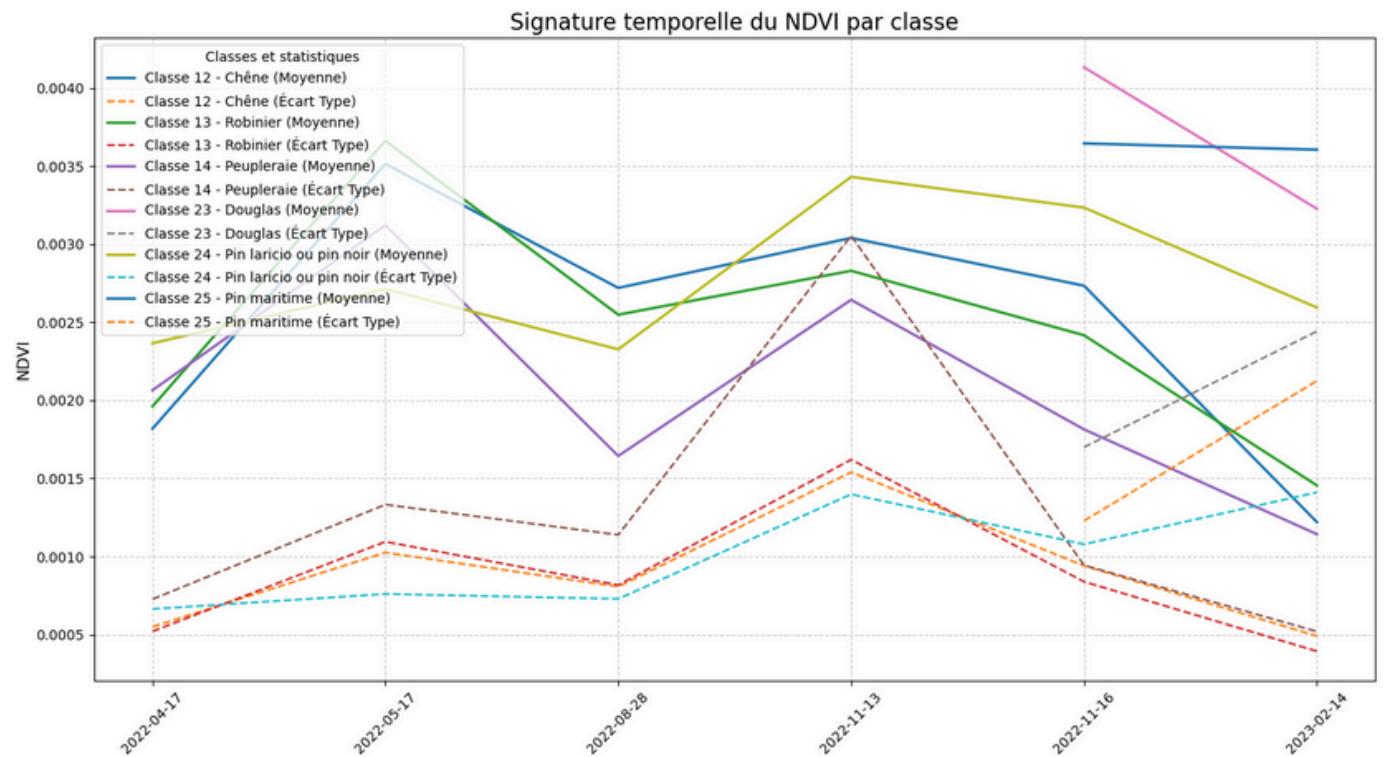


Figure 9 : diag\_baton\_dist\_centroide\_classe

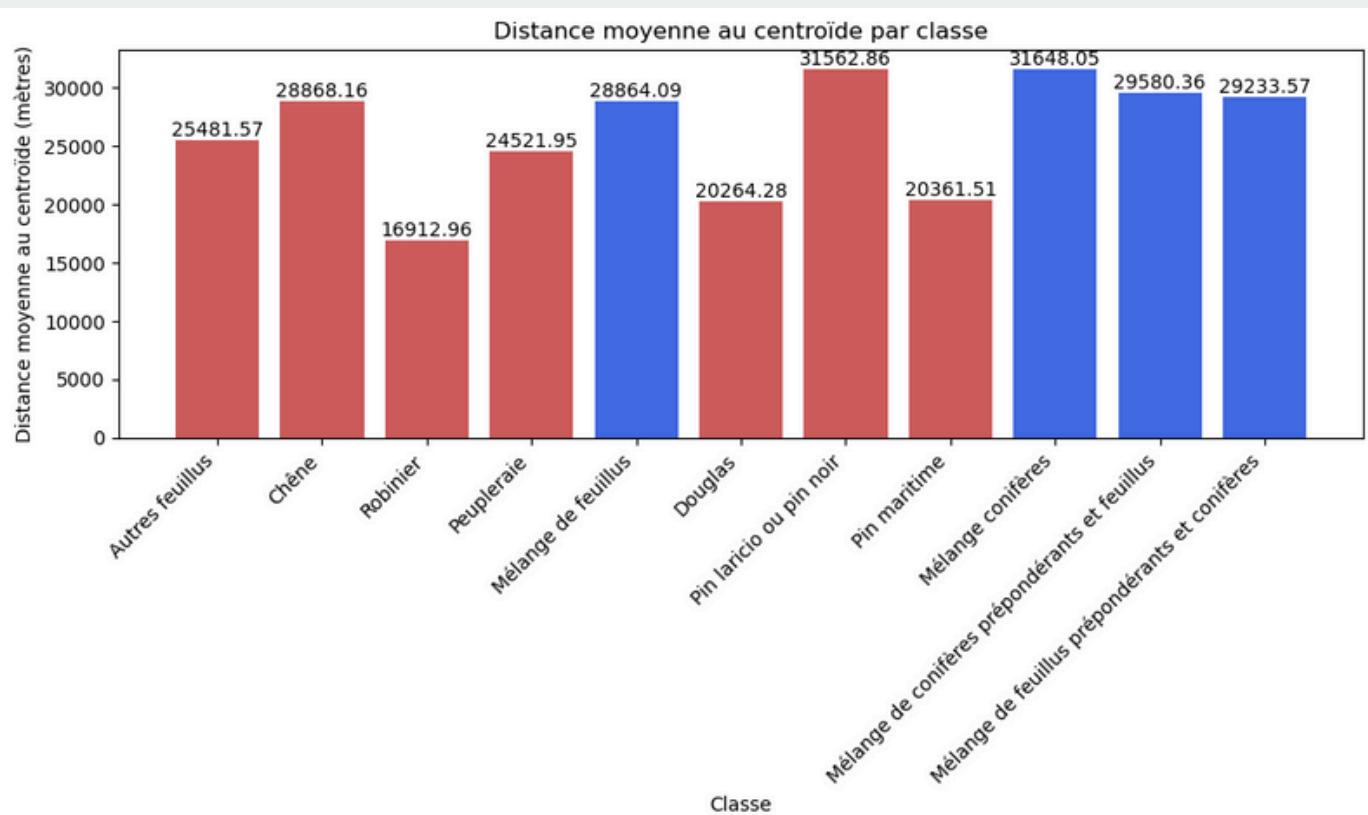


Figure 10 : violin\_plot\_dist\_centroide\_by\_poly\_by\_class

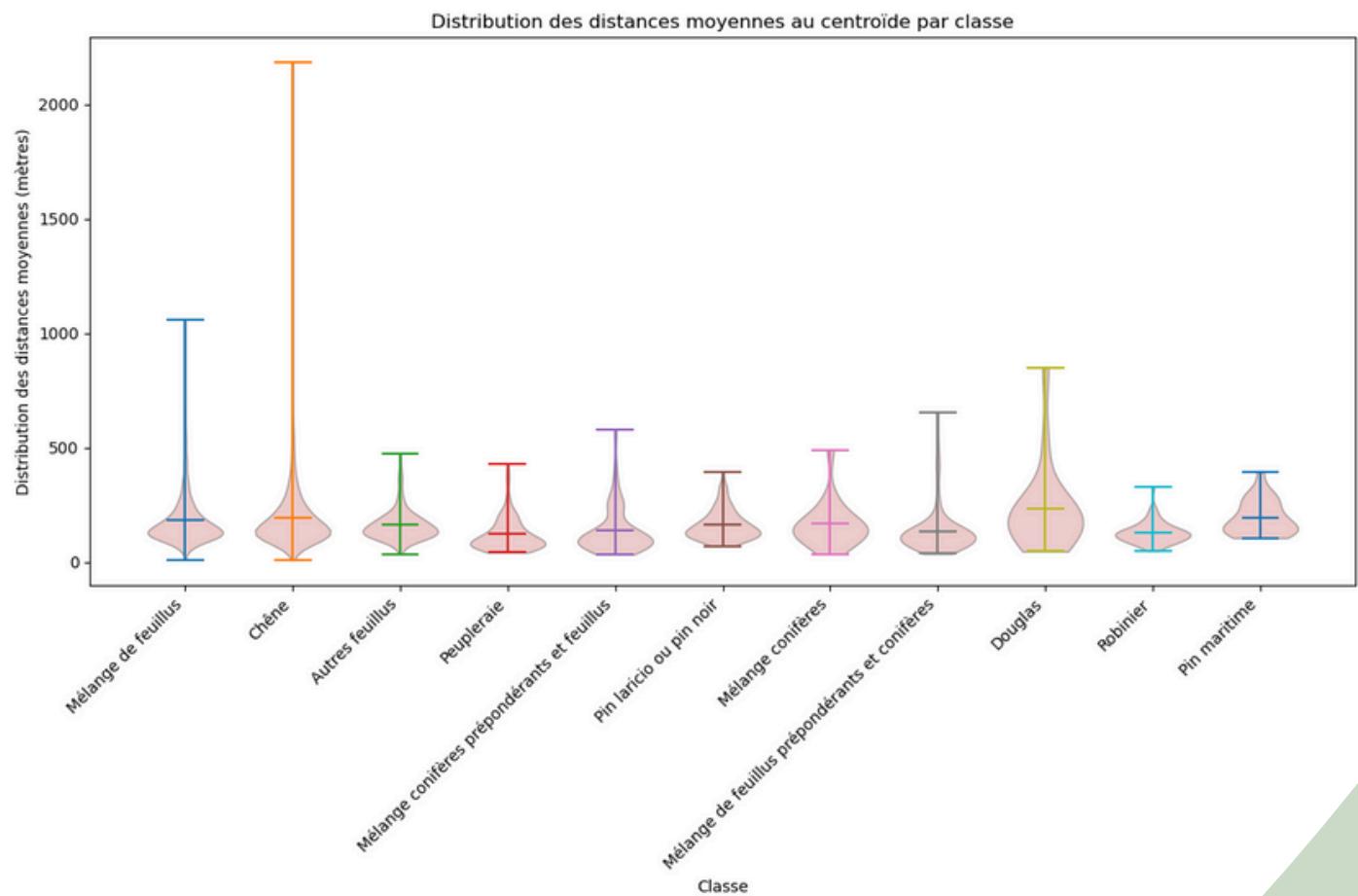


Figure 11 : Essences forestières à l'échelle du pixel

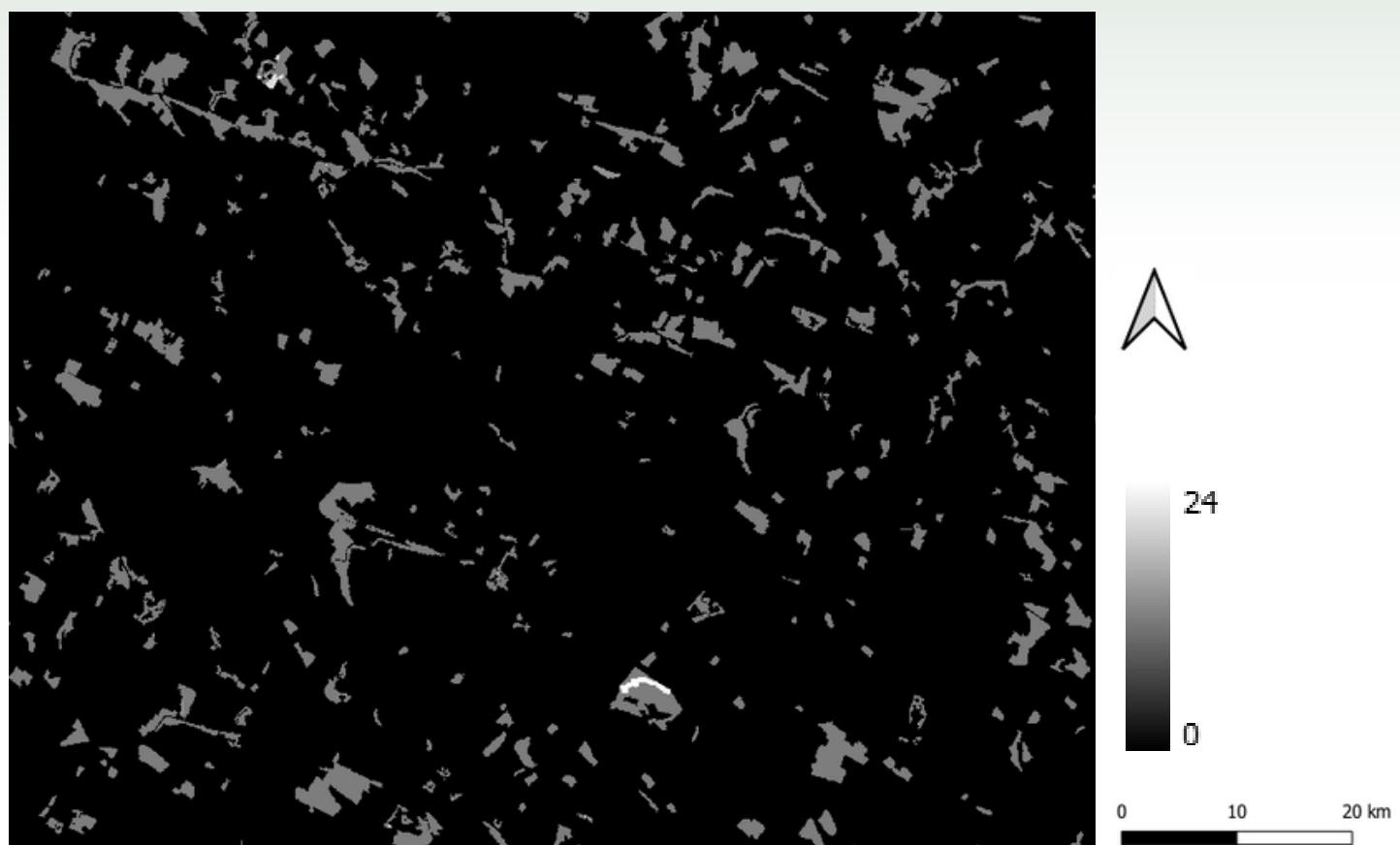
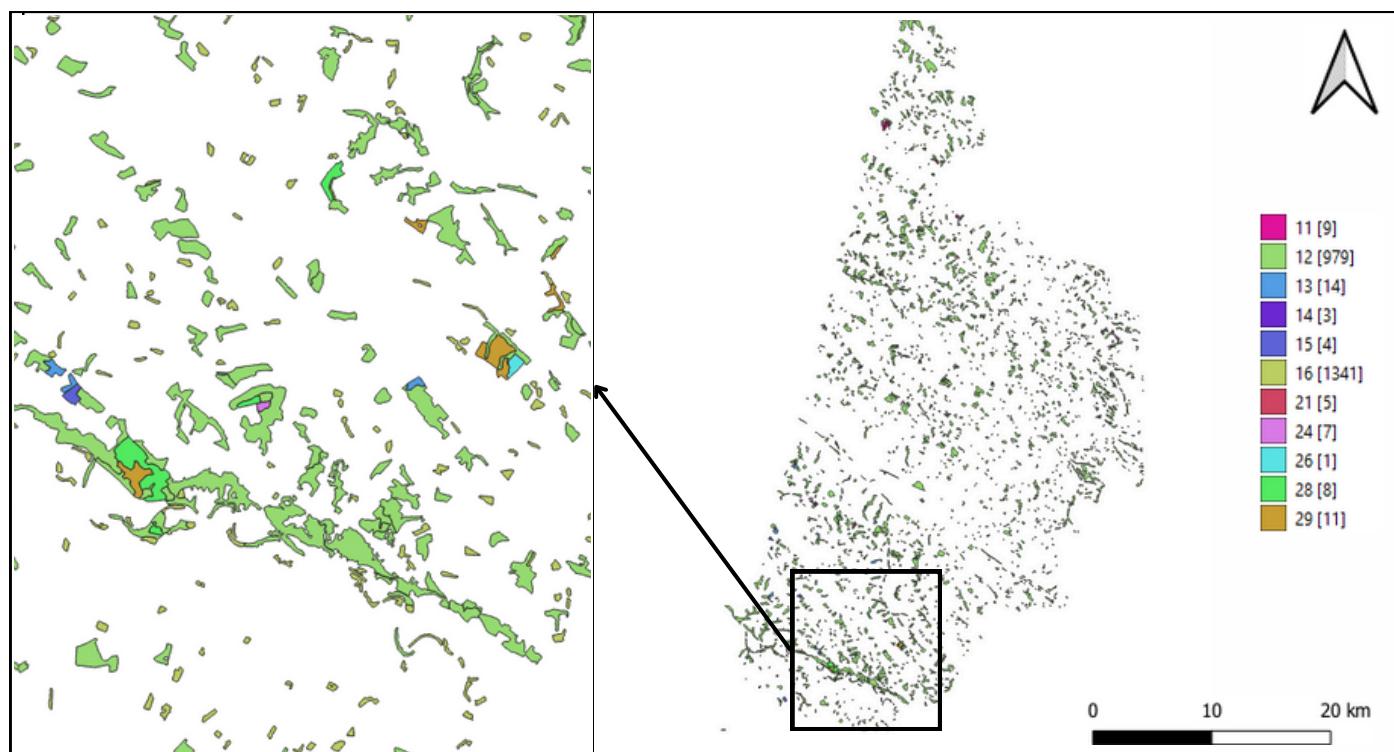
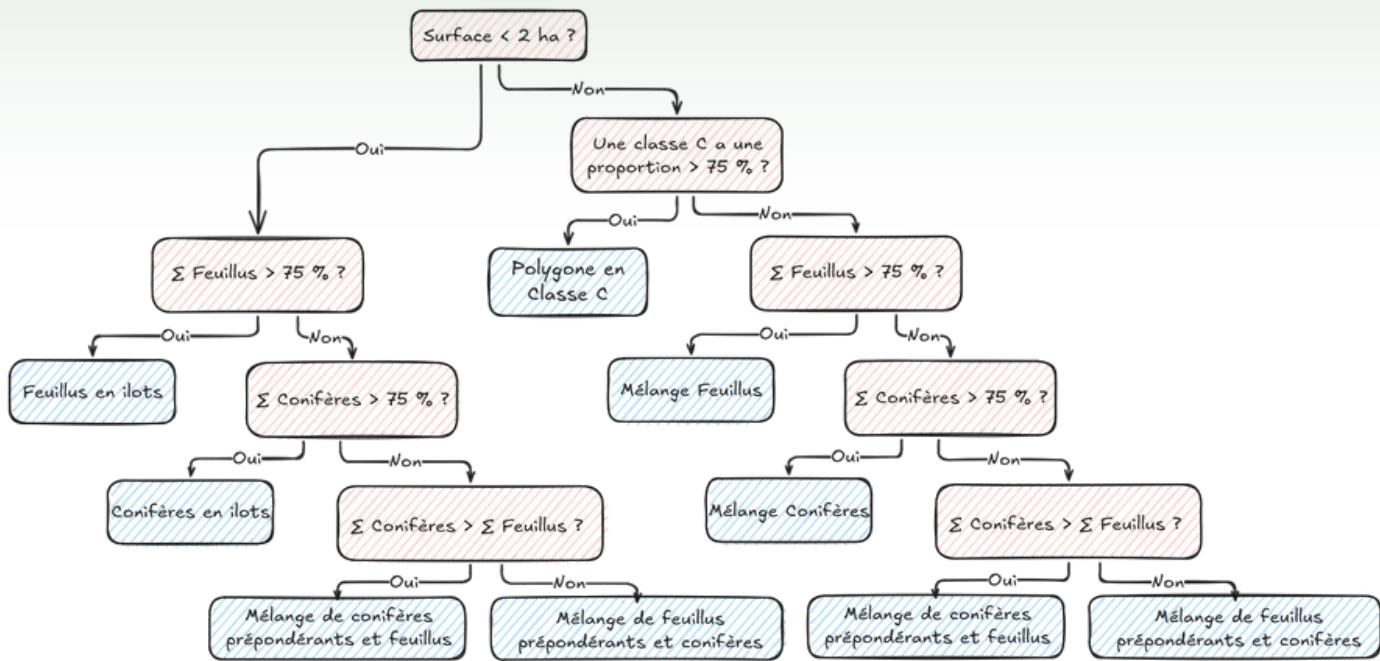


Figure 12 : Classification des polygones



**Figure 13 : Diagramme des consignes**



**Figure 14 : Matrice de confusion**

