

DOSSIER UE 901-22 - Projet Télédétection Approfondissement

MASTER 2 GÉOMATIQUE

« Sciences Géomatiques en environnement et Aménagement » (SIGMA)

Kanto Andrianarivony - Alizée Germain - Chloé Morgat - Anne-Sophie Pinna

Table des matières

Introduction.....	3
Notice d'utilisation des codes.....	4
■ Pré-traitement des images (<code>pre_traitement.py</code>)	4
■ Analyse des échantillons (<code>sample_analysis.py</code>)	5
■ Classification (<code>classification.py</code>)	5
■ Bibliothèque de fonctions (<code>my_function.py</code>).....	6
Description de la méthode	7
■ Pré-traitement des images (<code>pre_traitement.py</code>).....	7
■ Analyse des échantillons (<code>sample_analysis.py</code>).....	10
■ Classification (<code>classification.py</code>)	11
Résultats	14
■ Analyse des échantillons (20%).....	14
<i>Nombre de polygones par classe</i>	14
<i>Nombre de pixel par classe</i>	15
<i>Graphique de la signature temporelle de la moyenne et l'écart type du NDVI par classe</i>	16
■ Analyse des cartes des essences.....	18
Discussions	22
■ Validation du modèle de niveau 1.....	22
■ Validation du modèle de niveau 2.....	22
■ Validation du modèle de niveau 3.....	23
■ Comparaison des validations des modèles de niveau 1	24
■ Comparaison des validations des modèles de niveau 2	25
Conclusion	26

Introduction

Notre cas d'étude se focalise sur la commune de Toulouse et ses alentours (*Figure 1.*). L'objectif est de déterminer si l'inventaire forestier de la base de données IGN (*BDForêt_V2.0*) peut être utilisée comme source d'échantillons de référence pour réaliser une classification supervisée de séries temporelles d'images Sentinel 2. En effet, la BDForêt constitue le référentiel géographique de composition et de distribution spatiale des essences forestières sur le territoire national. Toutefois, c'est une base de données peu mise à jour (la dernière remonte à 2016) et ne permet pas de cartographier les essences forestières à une échelle intra-peuplement.

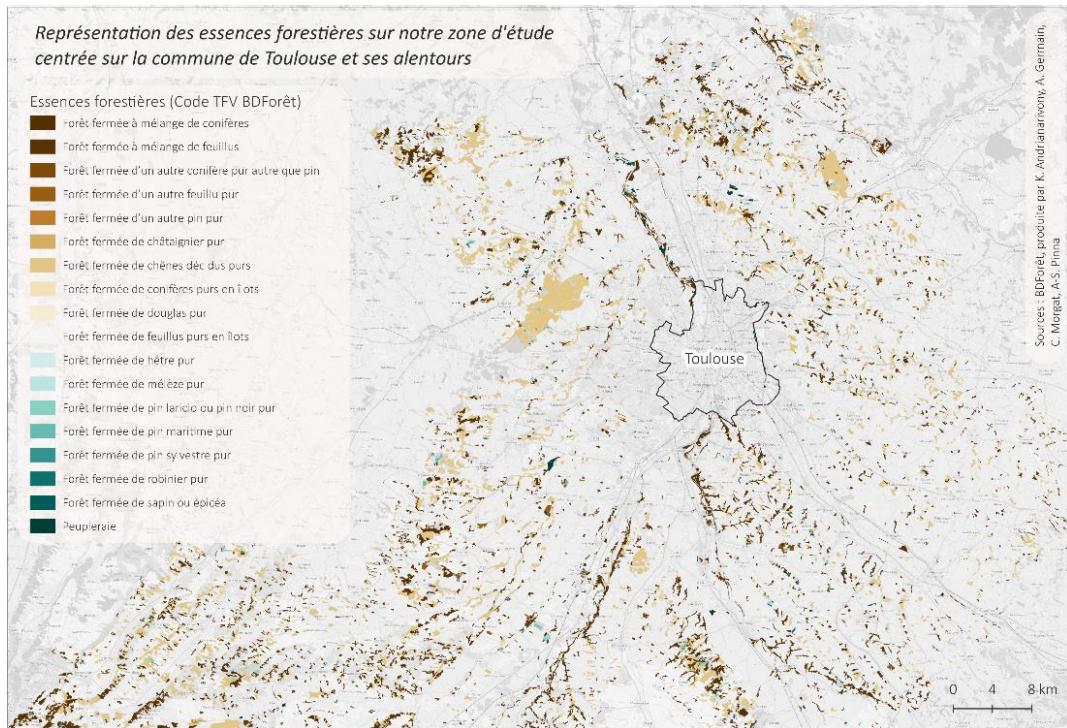


Figure 1. Localisation de notre zone d'étude pour notre cas d'application, et représentation des essences forestières (*BDForêt_V2.0*) sur cet espace

Pour répondre à cet objectif, plusieurs classifications supervisées seront réalisées pour différents niveaux de nomenclature d'essences forestières, dont sera évaluée la qualité des cartes produites à travers une validation stratifiée qui prend en compte l'appartenance des pixels à un polygone et qui sera répétée 30 fois avec 5 folds.

Dans le cas d'application exposé dans ce rapport, nous avons utilisé seulement 20% des échantillons de la BDForêt pour entraîner et valider nos modèles. Cette décision a pour seul but de permettre la réduction du temps de traitement qui est trop important (plus de 20 heures par modèle) lorsque la totalité des échantillons sont utilisés. Pour construire ce sous-échantillon, nous avons pris 20% de toutes les classes basées sur le niveau 3 de la reclassification de la BDForêt.
À noter que nos modèles fonctionnent parfaitement sur la totalité des échantillons, si l'utilisateur souhaite le tester, il peut remplacer la variable `Sample_BD_foret_T31TCJ` au début des scripts `sample_analysis.py` et `classification.py` par '`Sample_BD_foret_T31TCJ.shp`' (actuellement elle est égale à '`Sample_BD_foret_T31TCJ_split20.shp`').

Notice d'utilisation des codes

Assurez-vous que toutes les données nécessaires sont présentes dans le dossier spécifié (`./datas`)
En cas de problèmes ou de questions, veuillez-vous référer à la documentation de la fonction `my_function.py` ou contacter les développeurs.
Il est nécessaire de lancer les scripts dans un ordre précis, les uns étant dépendants des autres. Il faut lancer le script `pre_traitement.py` puis le script `sample_analysis.py` et finir avec le script `classification.py`.

- **Pré-traitement des images** (`pre_traitement.py`)

Nom du script : `pre_traitement.py`

Introduction :

Ce script a pour finalité d'obtenir deux images :

- ‘Serie_temp_S2_allbands.tif’ : une série temporelle multi-bandes ;
- ‘Serie_temp_S2_ndvi.tif’ : une série temporelle NDVI.

Pour ce faire, un pré-traitement est réalisé sur les images Sentinel 2 de niveau 2A THEIA et se décompose en plusieurs étapes :

- une reprojection selon le CRS de son choix ;
- un découpage selon l'emprise vecteur de son choix & un rééchantillonnage selon une bande Sentinel 2 de son choix (encodage ‘uint8’)
- le calcul du NDVI selon les bandes et les dates de son choix (encodage ‘uint8’)
- une concaténation selon les dates sélectionnées et un ordre de bande prédéfini (encodage ‘uint8’/‘float32’)
- l'application d'un masque (.tif) aux valeurs binaires (0 ou 1, zéro étant la valeur qui sera masquée) à l'image de son choix, et encodée en ‘uint8’/‘float32’.

Prérequis :

- Environnement Python avec les dépendances nécessaires (`sys, os, collections, osgeo, rasterio, numpy, fnmatch, shutil`).
- Le masque (raster) doit avoir la même emprise que les images Sentinel 2 traitées ;
- Le fichier `./datas` doit contenir l'emprise d'étude vectorielle shapefile et le masque forêt raster.
- Les images Sentinel 2 doivent être préalablement téléchargées et placées dans le fichier `./datas/Sentinel_2`

Etapes d'utilisation :

- Configuration de l'environnement : assurez-vous d'avoir Python installé avec les bibliothèques nécessaires. Installez toute bibliothèque manquante avec `pip install {nom_de_la_bibli}` Si nécessaire.
- Configuration du dossier : mettez à jour la variable `{my_folder}` avec le chemin d'accès au dossier contenant l'arborescence téléchargée. Exemple : `my_folder = ('C:/Users/Desktop/SIGMA/Projet_teledesc')`
- Exécution du script : exécutez le script dans un environnement Python.

Notes :

En sortie, quatre dossiers intermédiaires sont générés à l'intérieur du répertoire ‘`./results`’ puis supprimés en fin de script pour ne conserver que les séries temporelles finales :

- ‘`/S2_reproj`’ : images reprojetées ;
- ‘`/S2_clip_resample`’ : images découpées et rééchantillonées ;
- ‘`/S2_ndvi`’ : NDVI ;

- '/S2_concat' : concaténation des séries temporelles finales avec le masque forêt appliqué.
- Assurez-vous que la bibliothèque *my_function* contient les fonctions nécessaires pour une exécution réussie. Ce script suppose que vos données géospatiales respectent les exigences et les formats mentionnés dans le script.

- Analyse des échantillons (*sample_analysis.py*)

Nom du script : *sample_analysis.py*

Introduction :

Ce script effectue une analyse des échantillons sélectionnés en utilisant des données shapefiles et rasters. Il inclut des fonctions pour créer des diagrammes en bâtons du nombre de polygones (*fct.plot_bar_polygone*) et du nombre de pixels par classe (*fct.plot_bar_pixel*) ainsi que des graphiques de signatures temporelles moyennes et d'écart type du NDVI par classe (*fct.plot_graphs*).

Prérequis :

- Environnement Python avec les dépendances nécessaires (*sys, os, matplotlib, gdal, geopandas*) ;
- Le fichier *./datas* doit contenir le shapefile des échantillons.
- Le fichier *./results* doit contenir l'image '*Serie_temp_S2_ndvi.tif*' produit par le script *pre_traitement.py*.

Etapes d'utilisation :

- Configuration de l'environnement : assurez-vous d'avoir Python installé avec les bibliothèques nécessaires. Installez toute bibliothèque manquante avec *pip install {nom_de_la_bibli}* Si nécessaire.
- Configuration du dossier : mettez à jour la variable *{my_folder}* avec le chemin d'accès au dossier contenant l'arborescence téléchargée. Exemple : *my_folder = 'C:/Users/Desktop/SIGMA/Projet_teledesc'*
- Exécution du script : exécutez le script dans un environnement Python.
- Personnalisation : ajustez les étiquettes (labels_lvl1, labels_lvl2, labels_lvl3) et les schémas de couleurs selon vos préférences.

Notes :

Les diagrammes de l'analyse des échantillons seront sauvegardés dans le dossier '*./results*'.

Assurez-vous que la bibliothèque *my_function* contient les fonctions nécessaires pour une exécution réussie. Ce script suppose que vos données géospatiales respectent les exigences et les formats mentionnés dans le script.

- Classification (*classification.py*)

Nom du script : *classification.py*

Introduction :

Ce script effectue des classifications d'essences forestières avec la fonction *fct.classification* basées sur des échantillons de la BD Forêt reclassés selon trois niveaux de nomenclature et à partir de données spatiales et temporelles. Il utilise un modèle de machine learning nommé *RandomForestClassifier* de la bibliothèque *sklearn.ensemble*. Le script effectue également la validation de ce modèle en produisant un graphique de qualités et une matrice de confusion avec la fonction *fct.validation*.

Prérequis :

- Environnement Python avec les dépendances nécessaires (*sys, os, numpy, gdal, pandas, geopandas, matplotlib, sklearn, museotoolbox*) ;
- Le fichier *./datas* doit contenir le shapefile des échantillons.
- Le fichier *./results* doit contenir l'image '*Serie_temp_S2_allbands.tif*' produit par le script *pre_traitement.py* et les trois images par niveau *sample_BD_foret_T31TCJ_1vIX.tif* produit par le script *sample_analysis.py*.

Etapes d'utilisation :

- Configuration de l'environnement : assurez-vous d'avoir Python installé avec les bibliothèques nécessaires. Installez toute bibliothèque manquante avec *pip install {nom_de_la_bibli}*. Si nécessaire.
- Configuration du dossier : mettez à jour la variable *{my_folder}* avec le chemin d'accès au dossier contenant l'arborescence téléchargée. Exemple : *my_folder = 'C:/Users/Desktop/SIGMA/Projet_teledec'*
- Exécution du script : exécutez le script dans un environnement Python.

Notes :

Les cartes classifiées et les images png de validation seront sauvegardées dans le dossier '*./results*'.

Assurez-vous que la bibliothèque *my_function* contient les fonctions nécessaires pour une exécution réussie. Ce script suppose que vos données géospatiales respectent les exigences et les formats mentionnés dans le script.

- **Bibliothèque de fonctions (*my_function.py*)**

Le script *my_function.py* regroupe toutes les fonctions nécessaires à l'exécution des trois précédents scripts :

- *sample_analysis.py* ;
- *pre-traitement.py* ;
- *classification.py*.

Le fichier du script est placé dans le dossier *./scripts* et est appelé grâce à la bibliothèque *sys* qui le reconnaît comme un fichier contenant des librairies à mobiliser. Le fichier est importé dans les trois scripts qui l'utilisent tel que *import my_function as fct*. Pour faire appel à une de ses fonctions, le suffixe est *fct*. est placé devant la fonction appelée.

Description de la méthode

- Pré-traitement des images (`pre_traitement.py`)

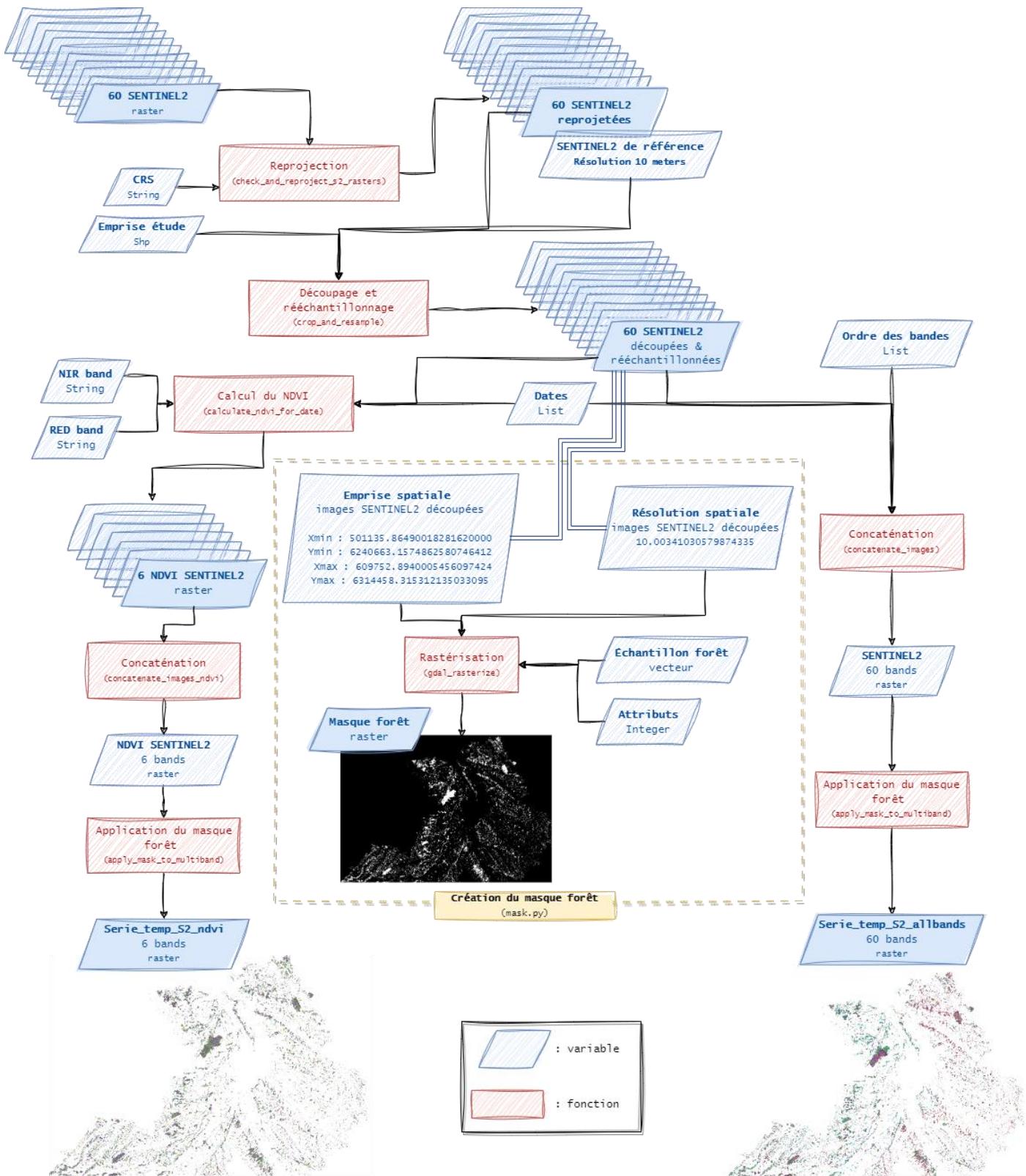


Figure 2. Diagramme de flux : pré-traitement

Dans notre cas d'application, utilisez les images Sentinel 2 correspondants à la tuile T31TCJ, ayant une couverture nuageuse inférieure à 15%, en réflectance (niveau 2A THEIA), à savoir les dates suivantes :

- 01/03/2021
- 31/03/2021
- 05/04/2021
- 15/04/2021
- 19/07/2021
- 31/12/2021

Placez ces images téléchargées dans le dossier `./datas/Sentinel_2`.

Ce script génère plusieurs dossiers et résultats intermédiaires qui sont supprimés à la fin du script (voir la partie précédente '*Notice d'utilisation des codes*') pour n'obtenir en sortie que les deux images reprojetées en EPSG : 2154, à 10 mètres de résolution spatiale, découpées selon une emprise d'étude et un masque forêt :

- '`Serie_temp_S2_allbands.tif`' : une image de 60 bandes (10 bandes par date) ;
- '`Serie_temp_S2_ndvi.tif`' : une image de 6 bandes (NDVI des 6 dates).

Le script commence par sélectionner les images de types *Flat Reflectance* (suffixe FRE) parmi les dossiers téléchargés des images Sentinel 2, et s'ensuit les traitements expliqués par le diagramme de flux ([Figure 2.](#)) et détaillés ci-dessous :

- Reprojection des images Sentinel 2 :

La fonction `check_and_reproject_s2_rasters` permet une reprojection les images Sentinel 2 sélectionnées en EPSG : 2154.

- Découpage et rééchantillonnage :

La fonction `crop_and_resample` permet le découpage des images selon l'emprise de notre zone d'étude (`emprise_etude.shp`). Le résultat de ce découpage est rééchantillonné selon une résolution à 10 mètres et un encodage '`uint8`', d'après une bande déjà échantillonnée à 10 mètres (voir [Figure 3.](#) ci-dessous). Dans le script, nous avons choisi l'image : `SENTINEL2A_20210405-105854-998_L2A_T31TCJ_C_V3-0_FRE_B2.tif` ([Figure 3.](#)).

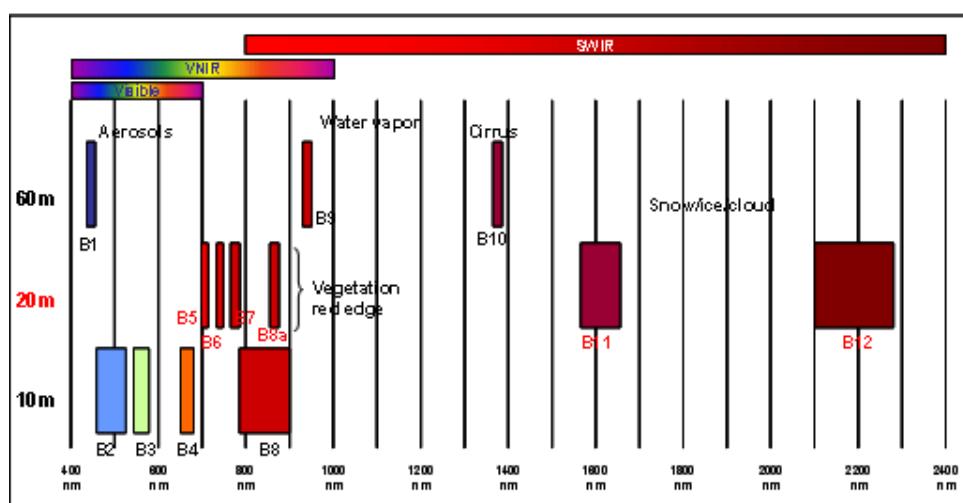


Figure 3. Représentation des différentes bandes spectrales S2 (cesbio.cnrs.fr)

- Concaténation des images :

La fonction `concatenate_images` permet de rassembler les 60 bandes par date croissante ([Tableau 1.](#)), encodée '`uint8`', afin d'obtenir une série temporelle multi bandes :

N°	Dates	Bandes
1	20210301	B2
2	20210301	B3
3	20210301	B4
4	20210301	B5
5	20210301	B6
6	20210301	B7
7	20210301	B8
8	20210301	B8A
9	20210301	B11
10	20210301	B12
...		
59	20211231	B11
60	20211231	B12

Tableau 1. Liste des bandes par dates de l'image concaténée

- Calcul du NDVI :

L'appel de la fonction `calculate_ndvi_for_date` permet de calculer le NDVI pour les 6 dates d'après les bandes B8 pour le Proche Infra-Rouge (NIR) et B4 pour le Rouge (revoir [Figure 3.](#)), encodée '`float32`'.

- Concaténation du NDVI :

Tout comme précédemment, la fonction `concatenate_images_ndvi` permet la concaténation des images NDVI par ordre croissant de date ([Tableau 2.](#)), encodé '`float32`', afin d'obtenir une série temporelle NDVI :

N°	NDVI_dates
1	20210301
2	20210331
3	20210405
4	20210415
5	20210719
6	20211231

Tableau 2. Liste des bandes par dates de l'image NDVI concaténée

- Application du masque forêt :

La dernière étape consiste à appliquer le masque forêt (`masque_foret.tif`) via la fonction `apply_mask_to_multiband` sur les 2 séries temporelles obtenues après concaténations afin d'obtenir nos rasters finaux :

- 'Serie_temp_S2_allbands.tif' encodée '`uint8`' ;
- 'Serie_temp_S2_ndvi.tif' encodée '`float32`'.

- Analyse des échantillons (`sample_analysis.py`)

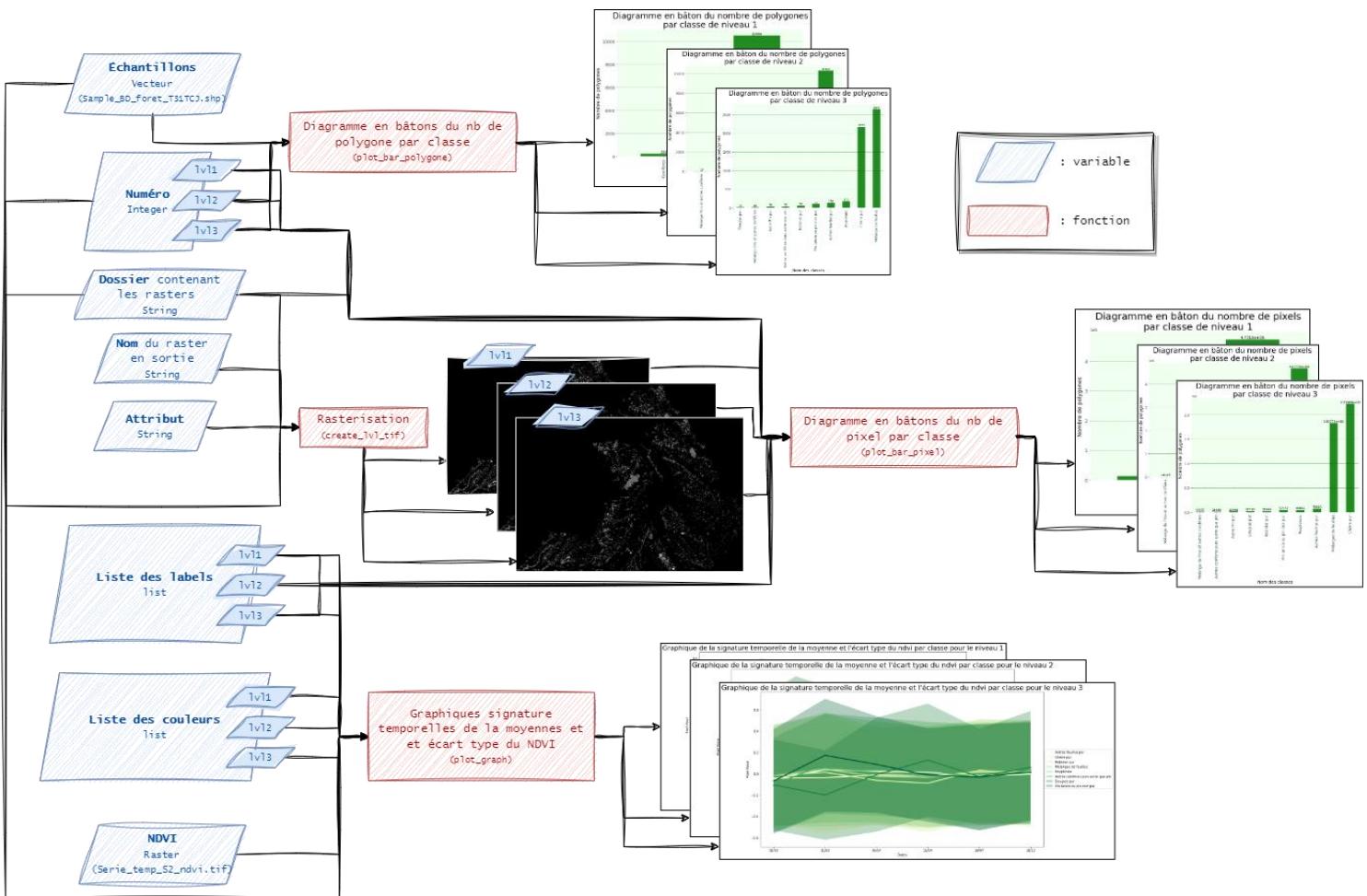


Figure 4. Diagramme de flux : analyse des échantillons.

Le script Python `sample_analysis.py` (Figure 4.) utilise la bibliothèque `geopandas` pour effectuer une analyse de données géospatiales. Ce script fait appel à plusieurs sous-fonctions (stockées dans la bibliothèque `my_function.py`) et se décompose en plusieurs étapes, détaillées ci-dessous :

- Lecture des données géospatiales et définition des étiquettes :

Un chemin d'accès au fichier shapefile contenant le jeu des échantillons (`Sample_BD_foret_T31TCJ.shp`) est spécifié. Trois listes (`labels_lv1`, `labels_lv2`, `labels_lv3`) des noms des espèces forestières à afficher sur le graphique sont définis pour les trois niveaux de nomenclature.

- Création de diagrammes en bâtons du nombre de polygones :

À partir du jeu d'échantillons (.shp) et du niveau de nomenclature (`Int`) que l'on souhaite étudier, la fonction `plot_bar_polygone` est appelée pour créer des diagrammes en bâtons représentant le nombre de polygones par classe.

- Création de diagrammes en bâtons du nombre de pixels :

Le jeu d'échantillons (.shp) est rastérisé via la fonction `create_lv1.tif` en trois images matricielles selon les champs attributaires : '`Code_lv1`', '`Code_lv2`', '`Code_lv3`' – correspondant aux différents niveaux de nomenclatures.

À partir de ces images et des listes d'étiquettes, la fonction `plot_bar_pixel` est appelée pour créer des diagrammes en bâtons représentant le nom de pixel par classe.

- Création des graphiques de signature temporelle moyenne et de l'écart-type du NDVI :

La fonction `plot_graphs` prend en compte la série temporelle du NDVI et un raster d'échantillonnage, puis génère un graphique représentant les moyennes et les écarts-types du NDVI pour chaque classe et selon les trois niveaux de nomenclature.

■ Classification (`classification.py`)

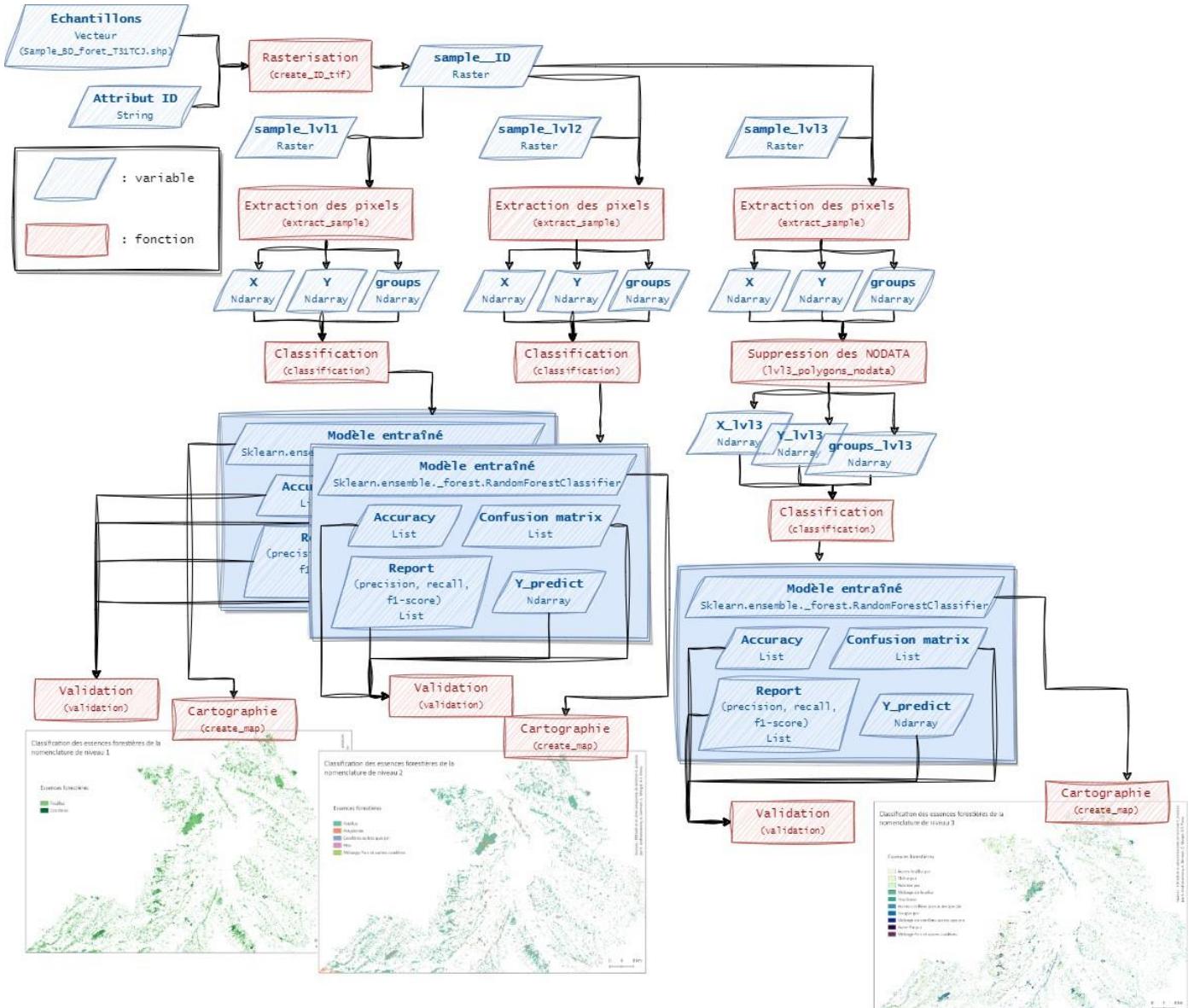


Figure 5. Diagramme de flux : classification.

Le diagramme de flux des modèles issus de regroupement est identique au diagramme ci-dessus (Figure 5.) à la seule différence que la fonction classification est remplacée par la fonction classification_from où le regroupement à lieu à chaque itération sur Y_test et Y_predict.

Les six modèles ont une structure commune telle que :

- Extraction des identifiants des polygones :

La fonction `fct.create_ID.tif` crée une image à partir du shapefile des échantillons en créant un identifiant unique par polygone et en enregistrant le résultat sous forme de raster (TIF) basé sur l'attribut d'identifiant créé.

- Extraction des échantillons :

La fonction `fct.extract_sample` extrait des échantillons de données à partir de l'image `Serie_temp_S2_allbands.tif`, de l'image `sample_BD_foret_T31TCJ_1v1X.tif` et de l'image `sample_BD_foret_T31TCJ_ID.tif` produite à l'étape précédente. Les données extraites comprennent une matrice d'échantillons x représentant les pixels des 60 bandes de l'image, une matrice y contenant les étiquettes des pixels, et une matrice `groups` contenant les étiquettes des polygones.

- Classification `RandomForestClassifier` :

La fonction `fct.classification` effectue une classification à l'aide d'un `RandomForestClassifier` avec une validation stratifiée à 5 fold, sur 30 itérations et en prenant en compte l'appartenance des pixels au polygone. Elle renvoie des listes contenant les matrices de confusion de chaque fold et chaque itération, les coefficients de qualités, les rapports de classification et le modèle `clf`. Pour réduire le temps de traitement, les paramètres suivants sont utilisés : `max_depth = 10` ; `n_estimators = 50` et `n_jobs = -1`.

- Validation de la classification :

La fonction `fct.validation` génère des graphiques (matrice de confusion et diagramme en barres de qualités) pour visualiser les résultats du modèle et estimer sa validité. Elle prend en entrée des listes de matrices de confusion, de coefficients d'exactitude et de rapports de classification produites à l'étape précédente.

- Création de la carte des essences forestières :

La fonction `fct.create_map` crée une carte classifiée prédite `carte_essences_1v1X.tif` à partir de l'image `Serie_temp_S2_allbands.tif` en utilisant le modèle `RandomForestClassifier` entraîné.

En plus de cette structure commune, quelques particularités sont à noter :

Particularité des modèles utilisant le niveau 3 de classification :

Les modèles utilisant le niveau 3 de classification nécessite une étape en plus entre la division des échantillons et la classification pour prendre en compte les polygones inutilisés dans ce niveau de classification. Cette étape se fait à l'aide de la fonction `fct_LVL3_polygons_nodata`. Cette fonction élimine des trois tableaux d'échantillons (`x, Y, groups`), les données des polygones qui ne sont pas présents dans le niveau 3 de la classification et qui ont le label '999' ou '998' dans le shapefile.

Particularité des modèles '`from`' :

La fonction `fct.classification_from` est construite sur le même modèle que la fonction `fct.classification` explicitée plus haut à la différence qu'elle effectue un regroupement des classes à l'aide du tableau numpy `array_class` fournit en entrée. Le regroupement à lieu avant de calculer les listes contenant les matrices de confusion de chaque fold et chaque itération, les coefficients de qualités et les rapports de classification.

Résultats

- Analyse des échantillons (20%)

Nombre de polygones par classe

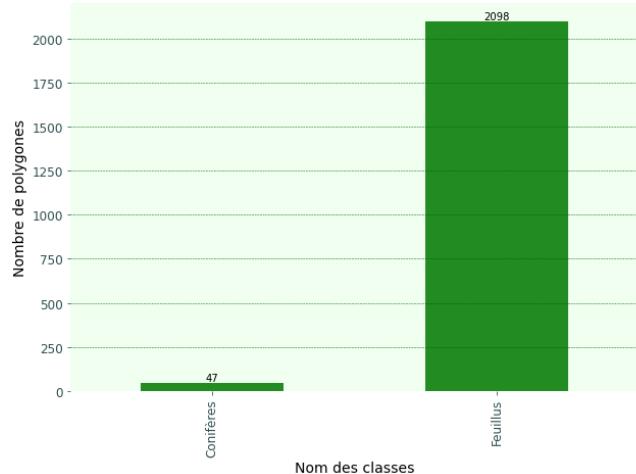


Figure 6. Diagramme en bâton du nombre de polygones par classe de niv 1.

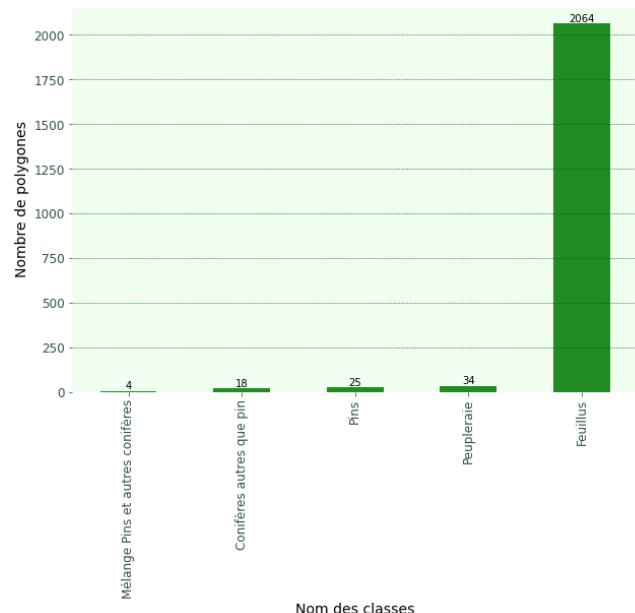


Figure 7. Diagramme en bâton du nombre de polygones par classe de niv 2.

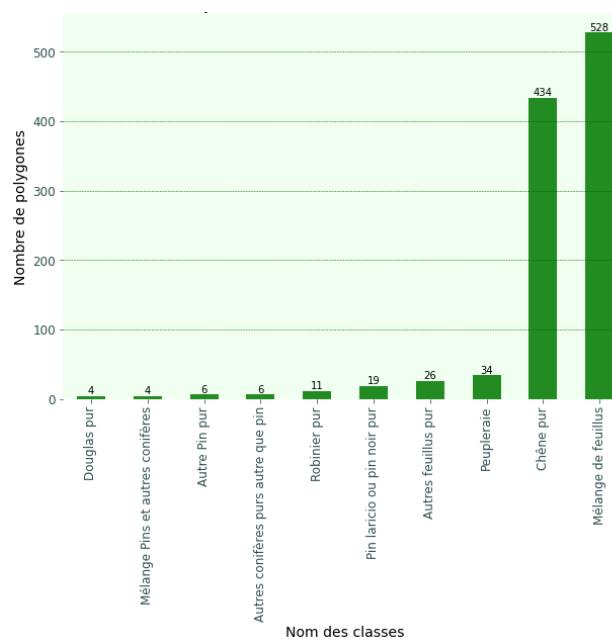


Figure 8. Diagramme en bâton du nombre de polygones par classe de niv 3.

À la vue de ces trois diagrammes nous pouvons distinguer deux classes dominantes : les classes « *Mélange de feuillus* » et « *Chêne pur* » présente dans le niveau 3 (**Figure 8.**) ; appartenant à la classe « *Feuillus* » du niveau 2 (**Figure 7.**) ; elle-même sous-catégorie de la classe « *Feuillus* » du niveau 1 (**Figure 6.**).

Les autres classes sont peu représentées en comparaison voire très peu représentées avec seulement quelques entités comme la classe « *Mélanges de pins et autres conifères* » du niveau 2 avec 4 polygones ou les classes « *Douglas pur* » et « *Mélanges de pins et autres conifères* » du niveau 3 avec seulement 4 polygones également.

Nombre de pixel par classe

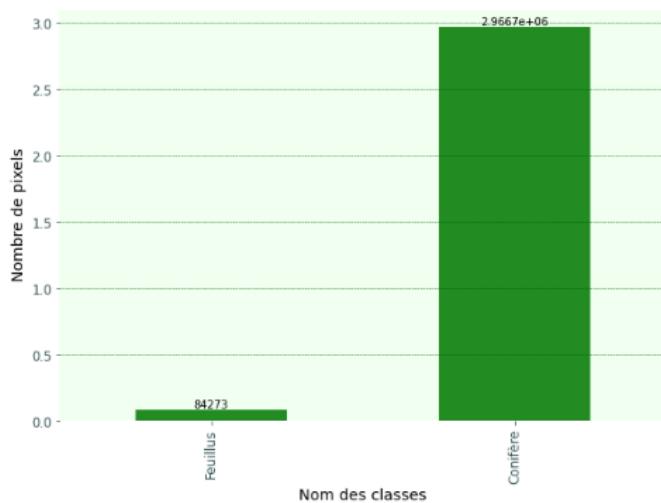


Figure 9. Diagramme en bâton du nombre de pixel par classe de niv 1.

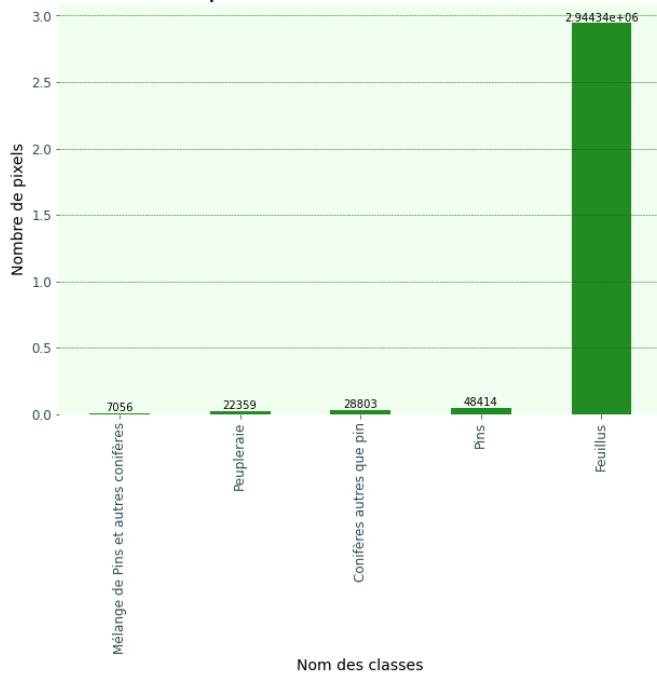


Figure 10. Diagramme en bâton du nombre de pixel par classe de niv 2.

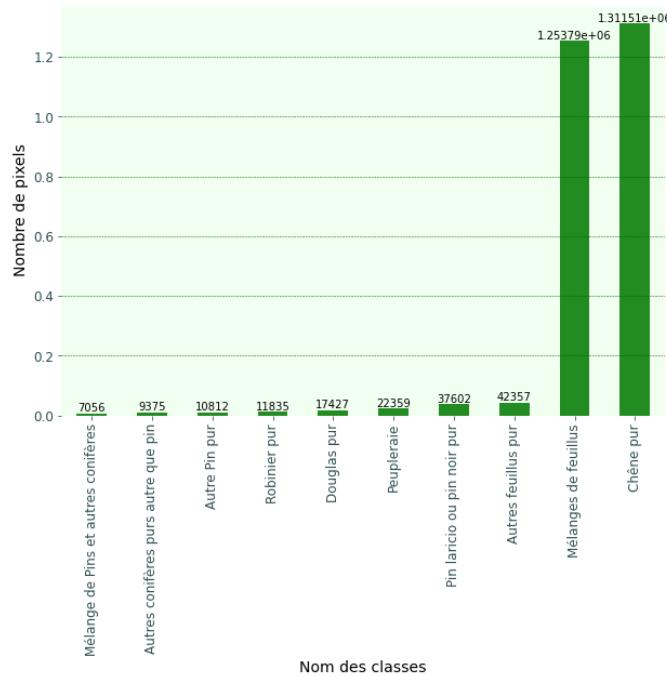


Figure 11. Diagramme en bâton du nombre de pixel par classe de niv 3.

À la vue de ces trois diagrammes, nous remarquons que les classes dominantes identifiées sur les diagrammes précédents sont ceux qui comportent le plus de pixels dans chaque niveau de nomenclature : 51,1% de pixels présents dans la classe « Chêne pur » et 48,9% de pixels présents dans la classe « Mélange de feuilllus » (niveau 3 ([Figure 11.](#))); 99,9% de pixels présents dans la classe « Feuilllus » (niveau 2 ([Figure 10.](#))) ; 99,9% de pixels présent dans la classe « Feuilllus » (niveau 1 ([Figure 9.](#))).

Nous pouvons faire le même constat que précédemment pour les autres classes qui ne contiennent que peu de pixels. La classe « Douglas purs » contient plus de pixels que la classe « Mélanges de pins et autres conifères » du niveau 3 et donc des polygones plus étendus.

Graphique de la signature temporelle de la moyenne et l'écart type du NDVI par classe

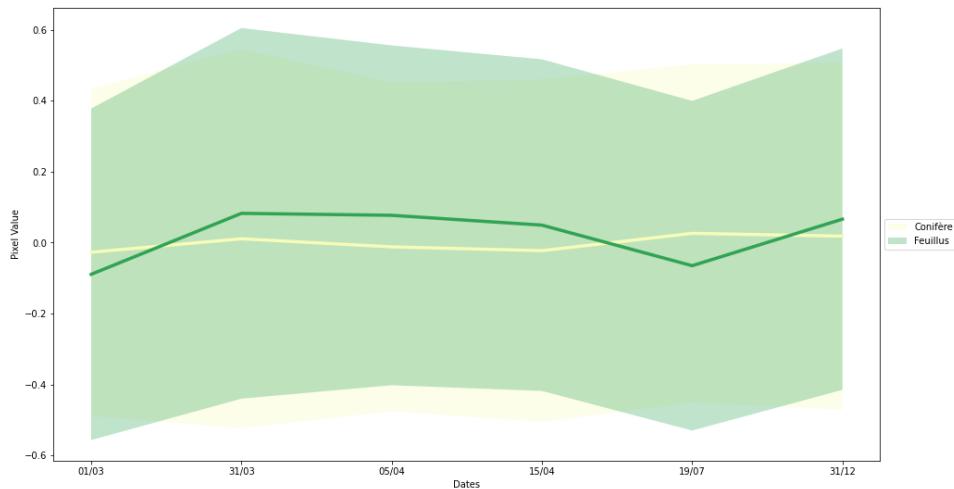


Figure 12. Graphique de la signature temporelle de la moyenne et écart-type du NDVI par classe pour le niveau 1

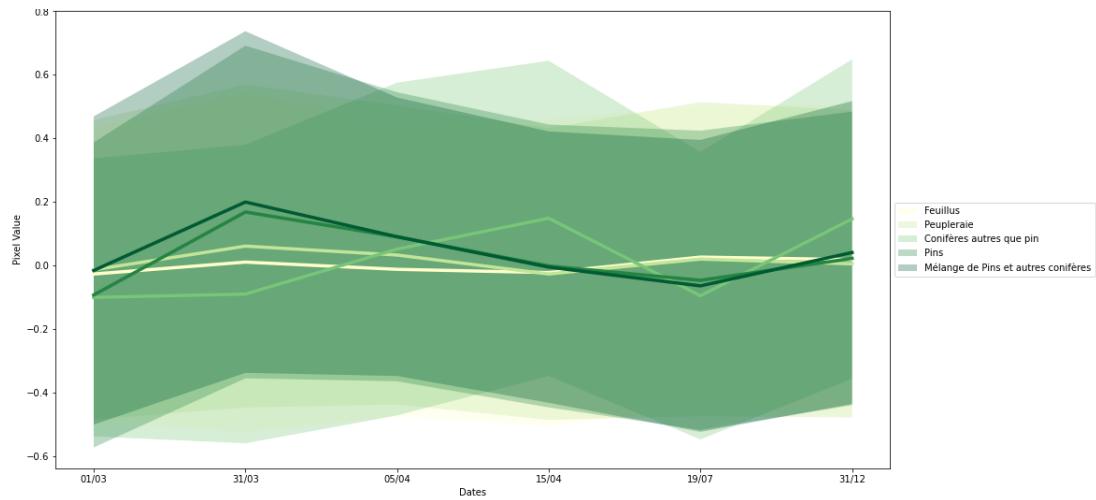


Figure 13. Graphique de la signature temporelle de la moyenne et écart-type du NDVI par classe pour le niveau 2.

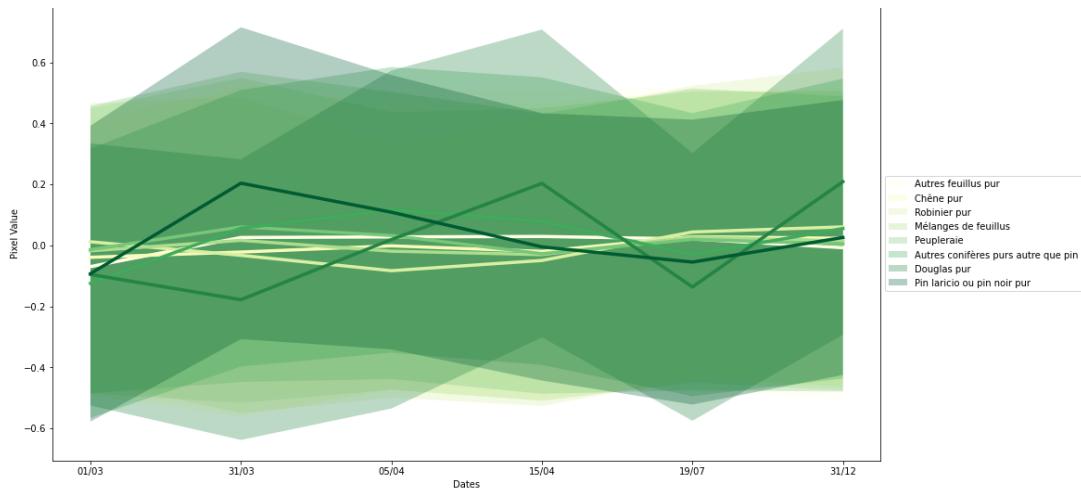


Figure 14. Graphique de la signature temporelle de la moyenne et écart-type du NDVI par classe pour le niveau 3.

Au regard de la signature temporelle du NDVI de la [Figure 12.](#) (niveau 1), on constate une moyenne de NDVI constante autour de 0.0 pour la classe « *Conifères* », caractéristique de leur non-perte de feuille. Au contraire, la classe « *Feuillus* » est caractérisée par une fluctuation de la moyenne du NDVI selon les saisons : un NDVI maximal au printemps et un NDVI minimal à l'approche de l'automne.

Sur la [Figure 14.](#) les classes « *Pins* » et « *Mélange de pins et autres conifères* » ont des signatures temporelles similaires. A l'inverse, la classe « *Conifères autres que pins* », est caractérisée par une signature temporelle très distincte. Cette différence peut expliquer la moyenne constante de la classe « *Conifères* » ([Figure 12.](#)) dont appartiennent ces 3 sous-classes.

En toute cohérence, les sous-classes « *Feuillus* » et « *Peupleraie* » découlant de la classe « *Feuillus* » ([Figure 12.](#)), ont des signatures temporelles équivalentes.

Suivant la logique précédente, les sous-classes de « *Feuillus* » concernant la nomenclature de niveau 3 ([Figure 14.](#)) : « *Autres feuillus purs* », « *Chêne pur* », « *Robinier pur* », « *Mélange de feuillus* » et « *Peupleraie* » semblent avoir des signatures temporelles de NDVI corrélées.

Cependant, dans les sous-classes de « *Conifères* », deux signatures temporelles se démarquent : « *Pin laricio ou pin noir pur* » et « *Douglas pur* ». Cette différence peut être expliquée par leur appartences à deux classes différentes du niveau de nomenclature 2 (« *Pins* » et « *Mélange de pins et autres conifères* »).

■ Analyse des cartes des essences

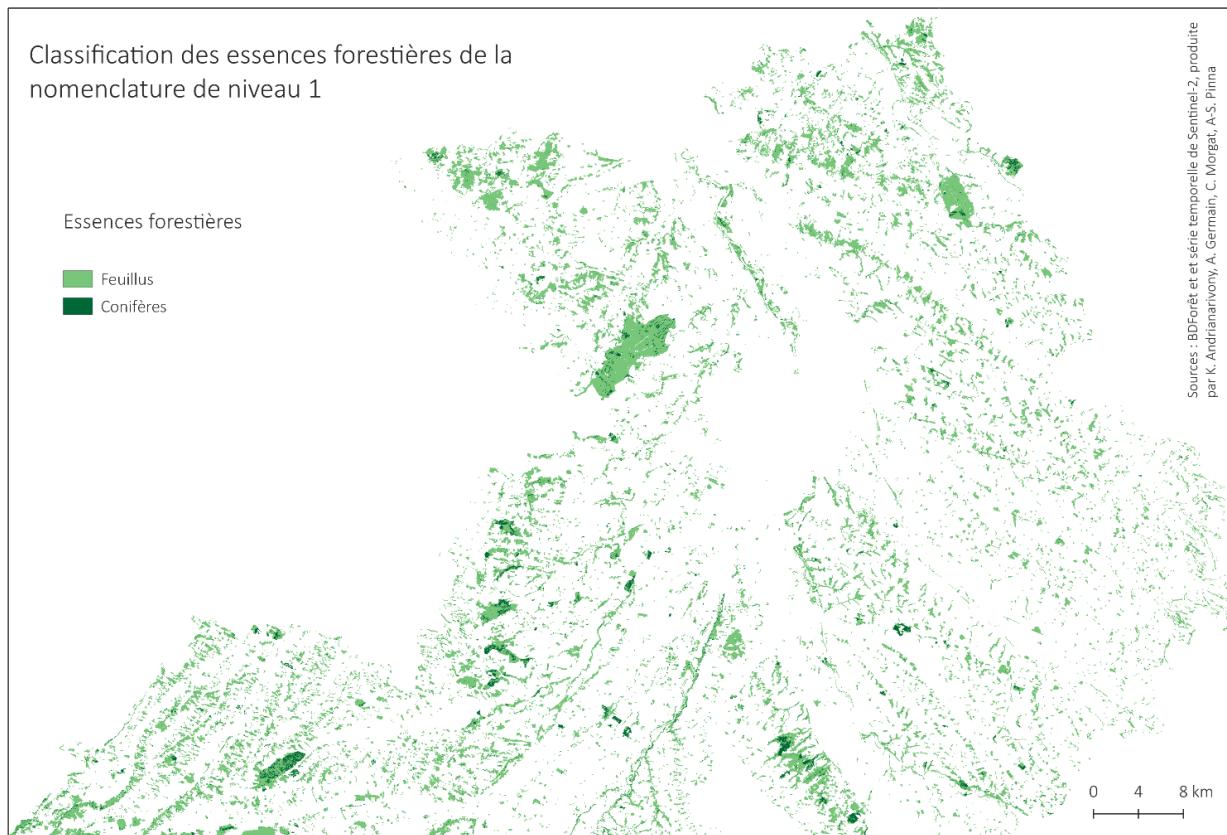


Figure 15. Classification des essences forestières de la nomenclature de niveau 1

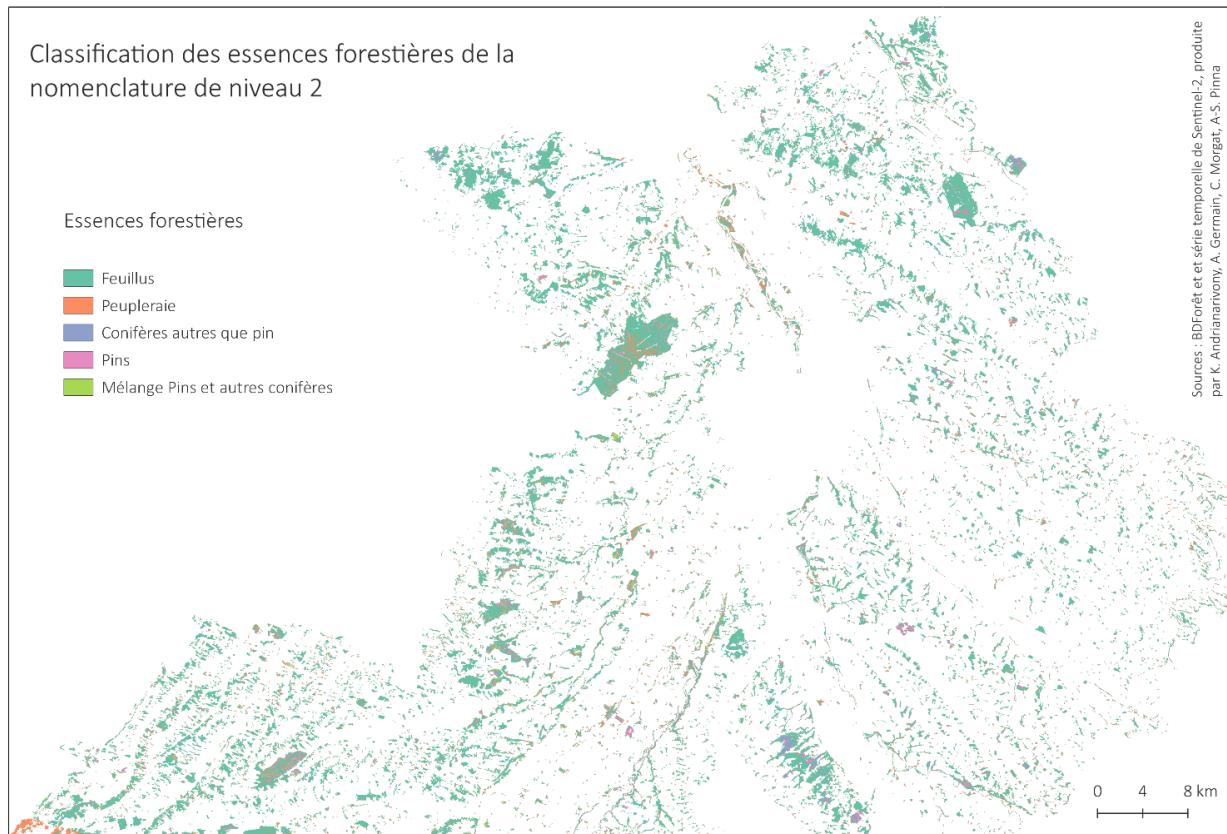


Figure 16. Classification des essences forestières de la nomenclature de niveau 2

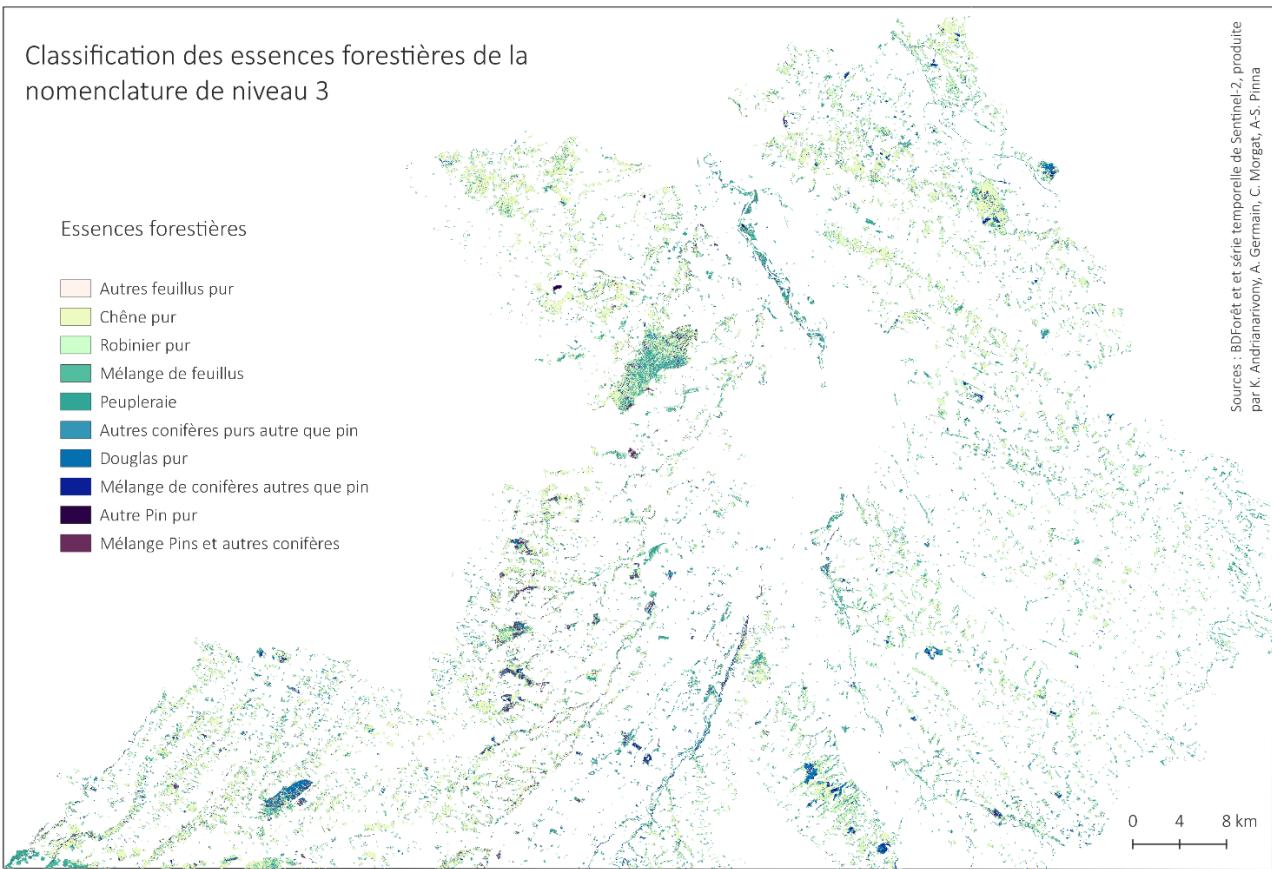


Figure 17. Classification des essences forestières de la nomenclature de niveau 3

La classification des essences forestières de la nomenclature de niveau 1 ([Figure 15.](#)) tout comme celle de la nomenclature de niveau 2 ([Figure 16.](#)) montre la dominance des feuillus sur les conifères.

Tout comme le niveau 1 et 2, la carte de classification des essences forestières de la nomenclature de niveau 3 ([Figure 17.](#)) fait ressortir la dominance de la classe « *Feuillus* » et plus particulièrement la sous-classe « *Chêne pur* ». De plus, cette nomenclature étant plus détaillée (10 classes), permet de constater une présence significative des « *Douglas* » pur appartenant à la classe des Conifères au Sud de la zone d'étude.

Classification des essences forestières de la nomenclature de niveau 2 issue du regroupement de niveau 3

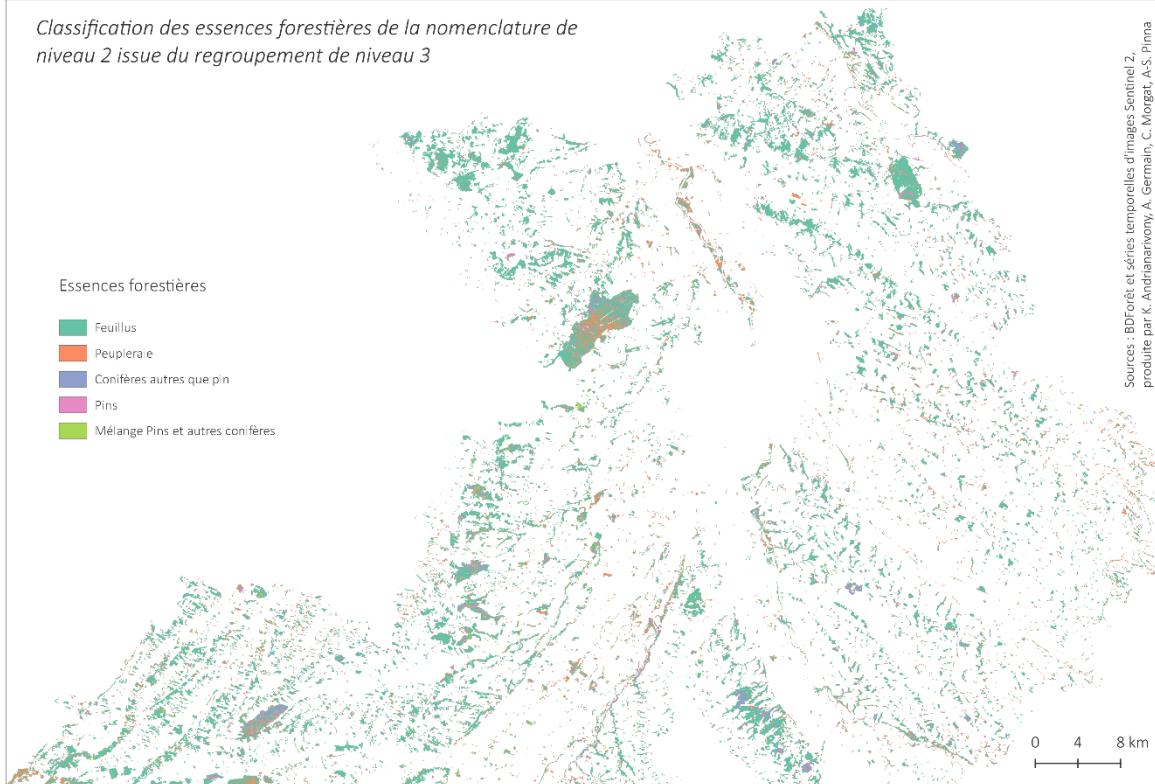


Figure 18. Classification des essences forestières de la nomenclature de niveau 2 issue du regroupement de niveau 3

Classification des essences forestières de la nomenclature de niveau 1 issue du regroupement de niveau 2

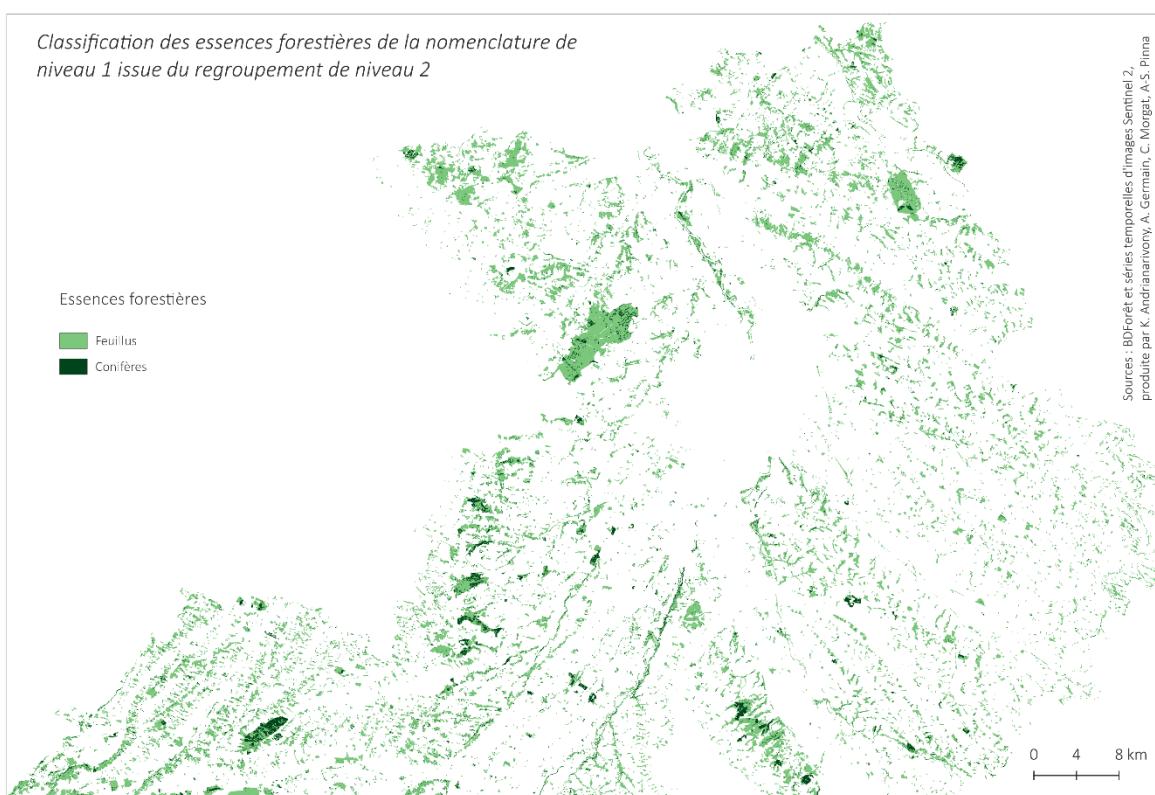
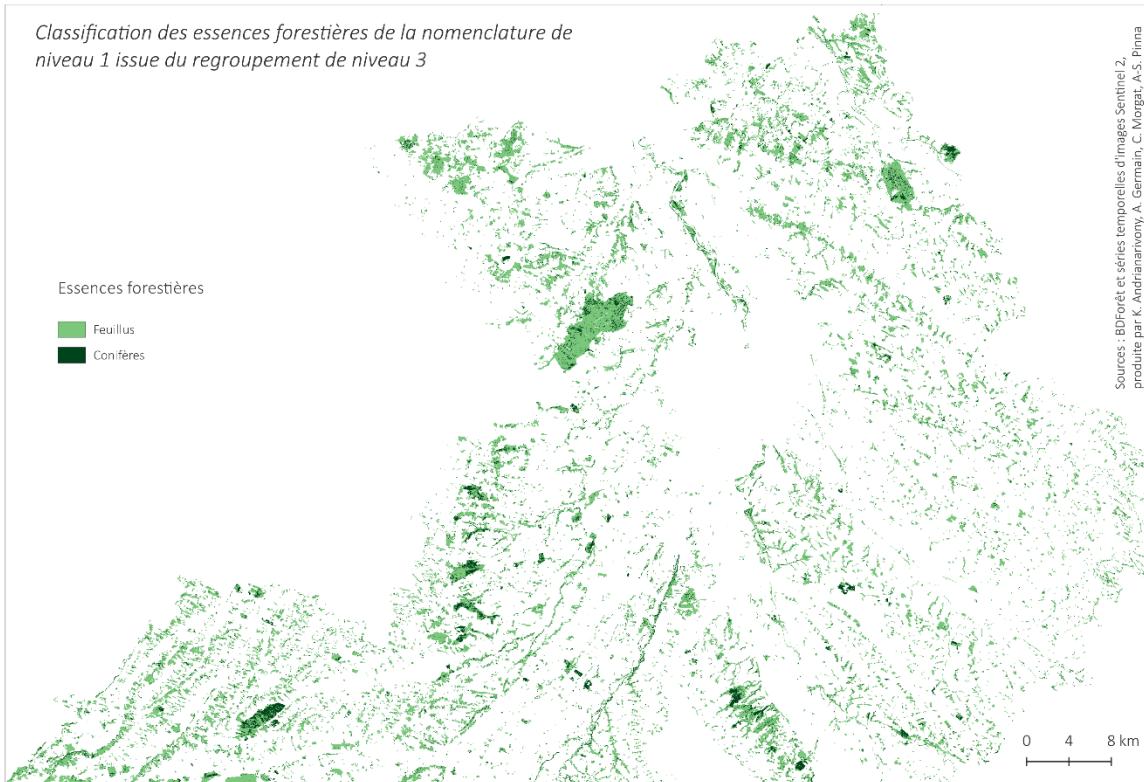


Figure 19. Classification des essences forestières de la nomenclature de niveau 1 issue du regroupement de niveau 2

Classification des essences forestières de la nomenclature de niveau 1 issue du regroupement de niveau 3



Sources : BDForêt et séries temporelles d'images Sentinel 2
produite par K. Andriamananjara, A. Germain, C. Mongat, A.S. Prina

Figure 20. Classification des essences forestières de la nomenclature de niveau 1 issue du regroupement de niveau 3

Les niveaux de nomenclature sont formés par regroupement des classes prédites par un niveau de nomenclature supérieur, à savoir : les classes de niveau 2 sont calculées d'après le regroupement des classes de niveau 3 ('*lvl2_from_lvl3*'), les classes de niveau 1 sont calculées d'après le regroupement des classes de niveau 2 ('*lvl1_from_lvl2*'), et les classes de niveau 1 sont calculées d'après le regroupement des classes de niveau 3 ('*lvl1_from_lvl3*'').

On retrouve les mêmes observations globales que précédemment : une dominance de la classe « *Feuillus* » pour '*lvl1_from_lvl2*' ([Figure 19.](#)) et '*lvl1_from_lvl3*' ([Figure 20.](#)). Toutefois, on remarque que le '*lvl1_from_lvl3*' semble apporter sensiblement plus de pixels classifiés comme « *Conifère* ».

Pour la classification '*lvl2_from_lvl3*' ([Figure 18.](#)), on retrouve une ascendance de la classe 'Feuillus'. Néanmoins, on peut remarquer que certains pixels considérés « *Feuillus* » pour '*lvl2*' simple, se disperse dans d'autres classes pour '*lvl2_from_lvl3*', notamment les classes « *Peupleraie* », « *Conifères autres que pin* » et « *Pins* ». Tout comme les classifications précédentes, on peut relever dans la zone Sud, une présence significative de « *Peupleraie* » et de « *Conifères autres que pin* ».

Discussions

- Validation du modèle de niveau 1

La matrice de confusion (*Figure 21.*) montre une nette performance sur la classification de la classe 1 « *Feuillus* » avec un F1-score de 96. Près de la totalité des pixels étant réellement des « *Feuillus* » ont été classé dans cette classe. Concernant la classe 2 « *Conifères* », la performance est bien moindre avec un F1-score de 36 bien que son Rappel soit un peu plus élevé de 63. Notons que la performance est dégradée notamment du fait que des pixels de la classe 1 ont été incorrectement classés comme des « *Conifères* ». Ces résultats sont visibles sur le diagramme en bâton (*Figure 22.*).

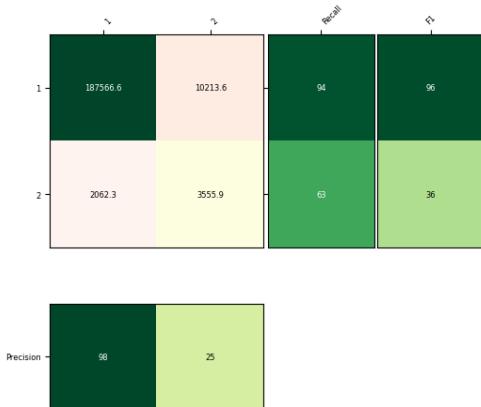


Figure 21. Matrice de confusion lvl1

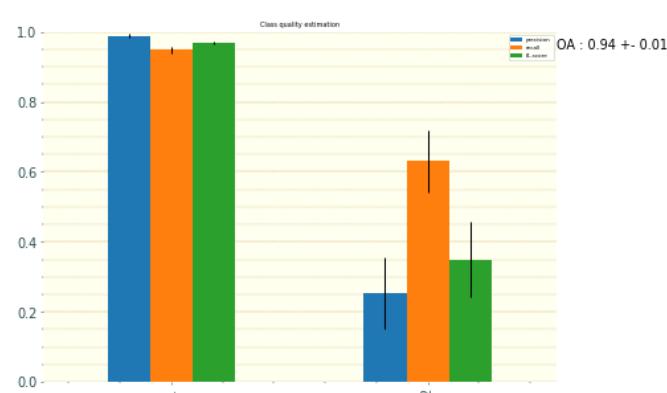


Figure 22. Diagramme en bâtons des indicateurs de validation du lvl1

- Validation du modèle de niveau 2

Sur le plan global, la précision est satisfaisante avec un indice d'overall accuracy de 0.78 (*Figure 24.*). Cependant, la matrice de confusion (*Figure 23.*) explique cette satisfaction globale par la sous-classe 10 « *Feuillus* » avec sa précision de 0.98 signifiant que les pixels ayant été classé en tant que « *Feuillus* » étaient des vrais positifs. De même, cette même sous-classe est celle contribuant au maximum à hauteur de plus de 80% sur la valeur élevée des indices F1-score et Rappel.

A contrario, c'est la sous-classe « *Mélange de Pins et autres conifères* » qui présente le plus d'erreur de classification ainsi que d'omissions. En effet, sa précision est de 0.0 et son Rappel est de 3.

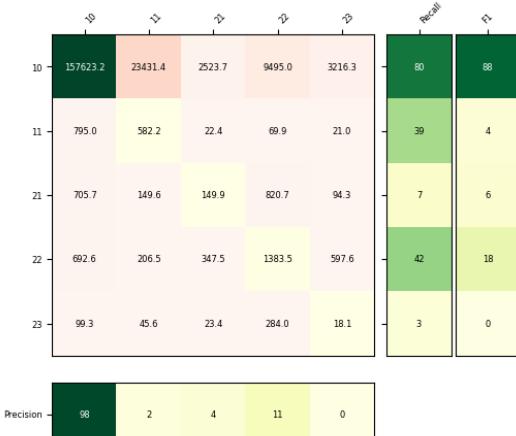


Figure 23. Matrice de confusion lvl2

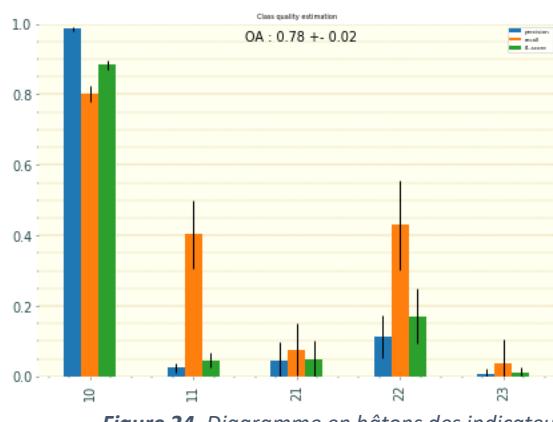


Figure 24. Diagramme en bâtons des indicateurs de validation du lvl2

■ Validation du modèle de niveau 3

A l'échelle globale, l'indice overall accuracy de 0.36 (Figure 26.) démontre la performance médiocre de la classification. La matrice de confusion (Figure 25.) montre une meilleure performance sur la classification de la classe 102 « Chêne pur » avec une fort nombre de pixels correctement classés. Les classes « Autres feuillus pur » et « Pin laricio ou pin noir pur » ont des performances beaucoup plus faibles : inférieur à 20 ; quant aux autres classes (« Robinier pur », « Mélange de feuillus », « Peupleraie », « Autres conifères purs autre que pin », « Douglas pur », « Mélange de conifères autres que pin », « Autre Pin pur », « Mélange Pins » et « Mélange Pins et autres conifères »), le F1-score est très faible : inférieur à 10.

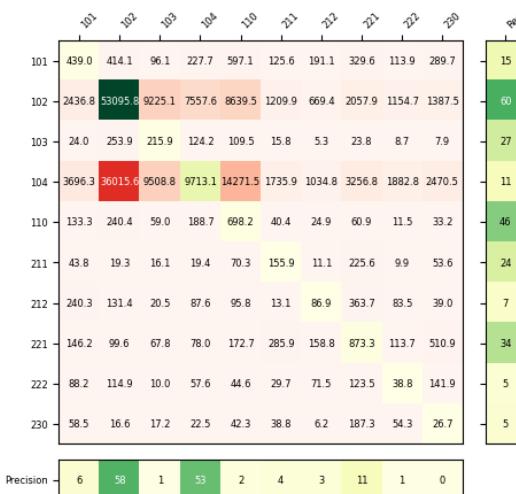


Figure 25. Matrice de confusion lvl3

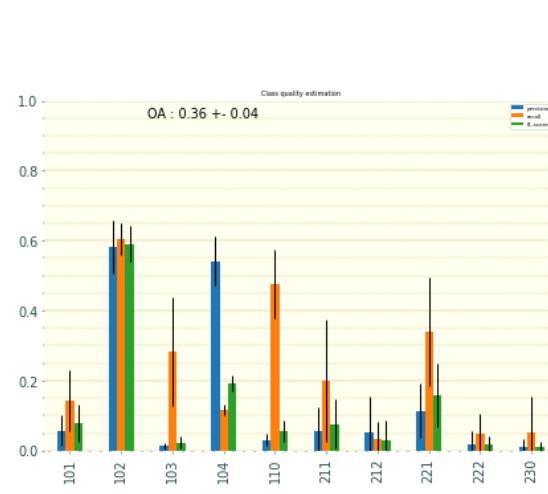


Figure 26. Diagramme en bâtons des indicateurs de validation du lvl3

Notons que la classe 102 est régulièrement confondus avec les classes 103, 104 et 110, ce qui peut s'expliquer facilement par le fait que ce sont toutes des classes de feuillus et ainsi que les ressemblances sont importantes. La principale confusion se fait entre la classe 102 « Chênes purs » et la classe 104 « Mélange de feuillus ». Nous pouvons observer plus de pixels appartenant à la classe 102 prédis dans la classe 104 que de pixels de la classe 104 prédis correctement (36 000 pour seulement 9 700).

- Comparaison des validations des modèles de niveau 1

Les matrices de confusion montrent une nette performance sur la classification de la classe 1 « *Feuillus* » avec un F1-score allant de 94 à 96 dans les trois modèles. Le modèle directement prédict sur le niveau 1 reste le plus performant suivi de près du modèle issu du regroupement des classes du niveau 2 puis de celui issu du regroupement des classes du modèle 3. La classe 2 « *Conifères* » reste prédictée avec peu de précision dans les trois modèles avec des F1-score allant de 26 à 36. Le nombre de pixels de la classe 2 correctement prédicts sont toujours d'environ 3 600 pixels. Ce qui diffère légèrement dans les modèles issus de regroupement est le nombre de pixels prédicts comme appartenant à la classe 2 alors que ce sont des « *Feuillus* ». Cette constatation diminue de fait la précision et le rappel et donc le F1-score de la classe 2 « *Conifères* ». Ces résultats sont également visibles sur les diagrammes en bâton.

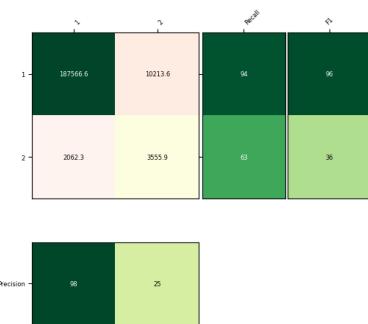


Figure 21. Matrice de confusion lvl1



Figure 27. Matrice de confusion lvl1_from_lvl3

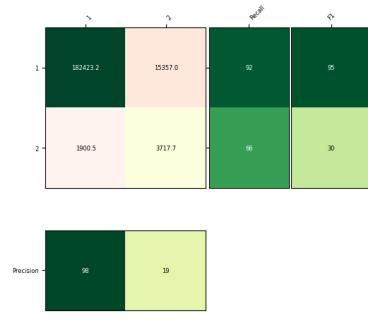


Figure 28. Matrice de confusion lvl1_from_lvl2

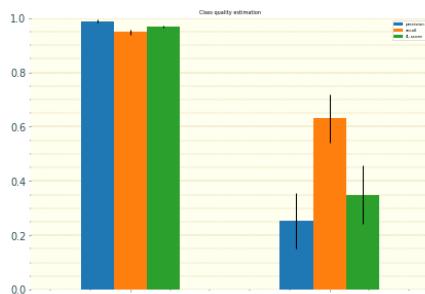


Figure 29. Diagramme en bâtons des indicateurs de validation du lvl1

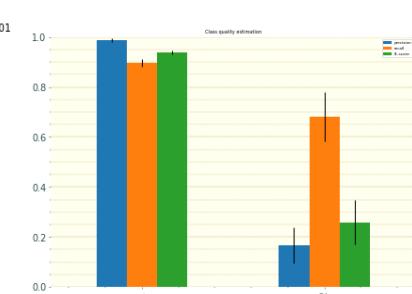


Figure 30. Diagramme en bâtons des indicateurs de validation du lvl1_from_lvl3

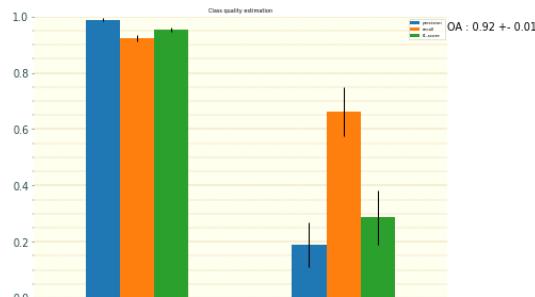


Figure 31. Diagramme en bâtons des indicateurs de validation du lvl1_from_lvl2

- Comparaison des validations des modèles de niveau 2

Les matrices de confusion montrent une nette performance sur la classification de la classe 10 “*Feuillus*” avec un F1-score de 85 et 88 dans les deux modèles. Le modèle directement prédit sur le niveau 2 reste le plus performant que le modèle issu du regroupement des classes du niveau 3, à l'exception des classes 11 et 23 qui ont un point de plus sur leur F1-score. Notons que les principales confusions sont entre la classe 10 dans les classes 11 “*Peupleraie*” et 22 “*Pins*”.

Ces résultats sont également visibles sur les diagrammes en bâton.

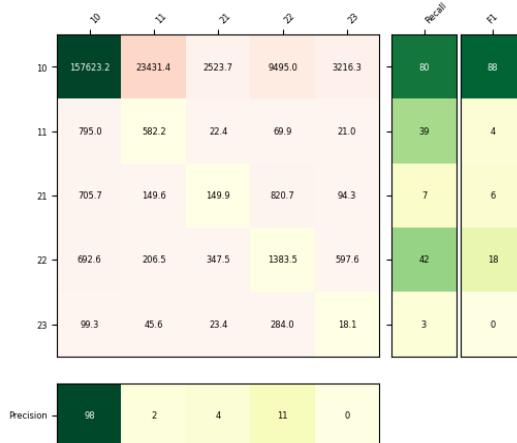


Figure 23. Matrice de confusion lvl2

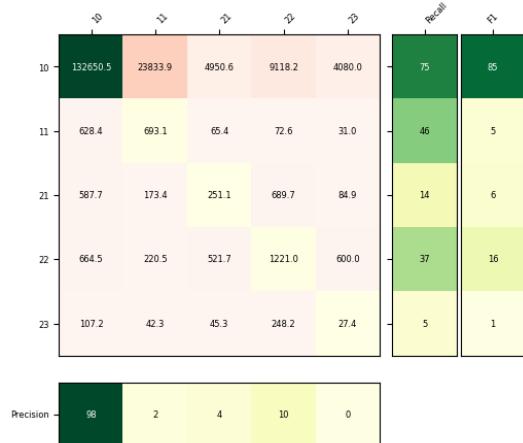


Figure 32. Matrice de confusion lvl2_from_lvl3

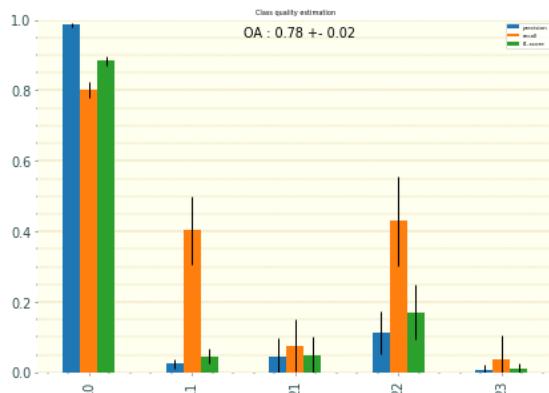


Figure 33. Diagramme en bâtons des indicateurs de validation du lvl2

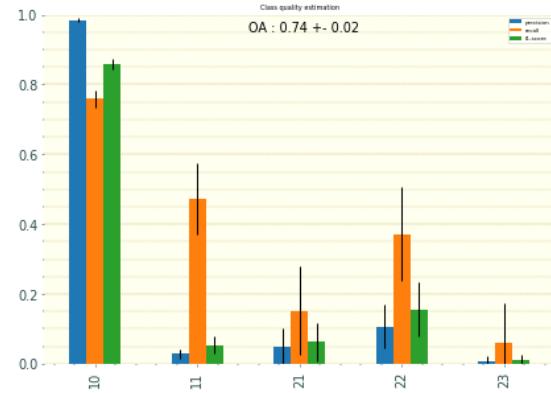


Figure 34. Diagramme en bâtons des indicateurs de validation du lvl2_from_lvl3

Conclusion

La mise en comparaison de nos modèles a révélé que les modèles indépendants sont plus performants que les modèles issus de regroupements. Cette performance suit la croissance suivante : le modèle *lvl1_from_lvl3* est moins performant que le modèle *lvl1_from_lvl2* qui est lui-même moins performant que le modèle *lvl1*. De même, le modèle *lvl2_from_lvl3* est moins performant que le modèle *lvl3*.

L'efficacité des modèles indépendants suit la performance croissante suivante : un niveau 3 avec une précision globale médiocre ($OA = 0.36$), un niveau 2 moyen ($OA = 0.74$), et un niveau 1 avec une précision globale excellente ($OA = 0.94$). Ce constat était attendu puisque le nombre de classes augmente de niveau en niveau et donc le nombre de données disponibles par classes est de plus en plus petit pour effectuer l'apprentissage ; une multiplicité de classes génère forcément une confusion supérieure lors de la classification.

L'objectif de ce projet était de déterminer si la *BDForet_V2.0* pouvait être employée comme source d'échantillons de référence pour réaliser une classification supervisée de séries temporelles d'images Sentinel 2. La réponse est mitigée en fonction des besoins de l'utilisateur. Si l'utilisateur cherche à définir seulement les étendus de feuillus, notre classification semble être pertinente sur le niveau 1. En revanche, si l'utilisateur souhaite explorer des classifications de niveau 2 ou 3, la réponse apportée par nos modèles est peu satisfaisante. L'entraînement et la validation sur les échantillons complets pourraient peut-être réduire cette médiocrité.

D'autres éléments sont à mettre en perspective de la performance de nos modèles, plusieurs biais interviennent notamment. Tout d'abord les données de bases étudiées : un inventaire forestier datant de 2016, et des images Sentinel 2 à l'année 2021. Cette disparité temporelle occulte de possibles modifications du couvert végétal qui seront alors mal interprétées par nos modèles. Aussi, rappelons que nous avons appliqué nos classifications supervisées sur seulement 20% des échantillons totaux de la BDForet. Avec les échantillons complets, nos performances devraient être supérieures. Enfin, le choix des classes regroupées par niveau de nomenclature, bien que logique, détient une part d'arbitraire.

Concernant notre méthodologie, quelques points peuvent être optimisés. Dans la fonction de classification pour les modèles de type « *lvl_from_lvl* » (fonction '*classification_from()*') pourrait être sous-jacent à la fonction de '*lvl*' indépendant (fonction '*classification()*'), et non constituer une nouvelle fonction à part entière. Une variable déterminant le besoin ou non du regroupement. Néanmoins, nous avons fait le choix de bien distinguer ces deux fonctions pour pouvoir subdiviser les tâches, et pouvoir faire tourner de manière autonome chaque modèle. Enfin, la limite de temps et de matériel a constitué un enjeu majeur tout le long de ce projet. Cette composante a été décisive dans la prise de décision méthodologique, à savoir : diminuer les arbres de décision ainsi que la taille des échantillons.