**Télédétection avancée**

**UE 901\_21**

**Projet de groupe**



Abir BEN ABDELGHAFFAR

Manuel ESTEVEZ

Judith LACRAMPE

Gabriel ORABONA

**Sommaire**

| [**Table des figures 3**](#_cwpiot2pkxap)  [**Table des tableaux 3**](#_hu8a6p7hnutu)  [**Table des scripts 3**](#_3f777a8ufs5b)  [**Introduction 4**](#_uj4zx5iuhgka)  [**1. Données et matériel 5**](#_r9lfaekj98jm)  [a. Les données utilisées 5](#_xi7jzbfp8o81)  [b. L’environnement de travail 5](#_s6ryc2kft4e4)  [**2. Méthodologie 6**](#_f2qxj5dm8p53)  [a. Préparation des données 6](#_yapl6qojc37l)  [b. Sélection et préparation des échantillon 6](#_rovqzdhb6dcc)  [c. Construction du masque forestier 9](#_l2mmqd6a3wmi)  [d. Classification supervisée 11](#_nwskwe40zgzu)  [e. Analyse des résultats 11](#_sbdp5wfr6js8)  [**3. Analyse des échantillons 12**](#_lxp84e6cpm40)  [a. Distribution des échantillons 12](#_c2d8nbwu6pbl)  [b. Phénologie des classes forestières 13](#_kwdz3pnnbg1t)  [c. Variabilité spectrale 14](#_oegrwqkren29)  [**4. Résultats 15**](#_17vykkgex03z)  [a. Cartes produites 15](#_j92dird8sd6)  [b. Qualité des classifications 15](#_upzu6kop9rmo)  [**5. Analyse critique 16**](#_ia4fwqspxm30)  [a. Interprétation des résultats 16](#_bmuobmzn5y8)  [b. Perspectives 16](#_8b4d31scjkhb)  [**Conclusion 17**](#_jkywlxmtqo56)  [**Annexes 18**](#_fnt2fyarmcn2)  [Annexe 1 - Utilité des scripts 18](#_rozctj4ixntg) |
| --- |

# Table des figures

| [Figure 1 - Correspondance entre les classes de la BD Forêt et les nomenclatures des cartes à obtenir 6](#_m9sq1vx552vg) |
| --- |

# 

# Table des tableaux

| [Tableau 1 - Nomenclature du masque forêt 7](#_7l1ynhhjmu7w) |
| --- |

# 

# Table des scripts

| [Script 1 - Création et préparation de l’échantillon 8](#_mcab76v4d61)  [Script 2 - Calcul du NDVI 8](#_e2khsdq2zjxb)  [Script 3 - Importation des libraires et création du masque 10](#_di7zhgezsepd)  [Script 4 - Fonction de classification Random Forest 12](#_md8hymah4j7n)  [Script 5 - Etude de la phénologie 14](#_sk6ixhx707g0)  [Script 6 - Calcul des centroïdes 15](#_k3r1w74wn1j6)  [Script 7 - Image matricielle multi-bandes et raster de séries temporelles 17](#_ds4btumerz8q)  [Script 8 - Carte des essences forestières 17](#_2rugmsaqhbgc)  [Script 9 - Carte à l’échelle des peuplements 18](#_1byo8vil6x1r)  [Script 10 - Analyse de la qualité des classifications 20](#_lk5m2ovri5mw) |
| --- |

# 

# Introduction

La gestion durable des forêts constitue une priorité dans le cadre des enjeux environnementaux, économiques et sociaux actuels. Les forêts, qui couvrent une part significative du territoire français, jouent un rôle fondamental dans la régulation climatique, la préservation de la biodiversité et l'approvisionnement en ressources naturelles. Pour répondre aux besoins croissants d'informations précises et actualisées sur les formations forestières, les outils de télédétection et les bases de données géographiques se révèlent essentiels.

La **BD Forêt® version 2.0**, une base de données de référence décrivant les essences forestières et les milieux naturels semi-ouverts, offre un panorama riche et détaillé des peuplements forestiers à l'échelle nationale. Toutefois, malgré son utilité pour de nombreuses applications (gestion des ressources, prévention des incendies, certification, etc.), cette base présente des limitations importantes, notamment en ce qui concerne l'actualisation des données et la résolution intra-peuplement. Ces contraintes réduisent son efficacité pour les analyses de précision et la cartographie à des échelles plus fines. L'émergence des technologies de télédétection, notamment les séries temporelles d'images satellites comme celles fournies par Sentinel-2, ouvre de nouvelles perspectives pour pallier ces insuffisances. Ces images permettent une analyse fine de l'évolution spatio-temporelle des formations forestières grâce à des indices tels que le NDVI (Normalized Difference Vegetation Index), offrant ainsi une vision complémentaire et potentiellement enrichissante par rapport aux bases de données statiques.

Ce projet vise à explorer la possibilité d'utiliser la **BD Forêt® version 2.0** comme source d'échantillons de référence pour entraîner une classification supervisée de séries temporelles d'images Sentinel-2. À travers une approche combinant données géographiques et techniques de télédétection, notre objectif est double :

1. **Produire des cartes des essences forestières à l’échelle du pixel et à l’échelle des peuplements.**
2. **Évaluer la qualité des cartes produites en termes de précision et d'utilité pour la gestion forestière et la planification territoriale.**

Le projet repose sur une méthodologie rigoureuse comprenant le prétraitement des images, la génération d’échantillons, la classification des données et l’analyse critique des résultats. À terme, il permettra de poser les bases d’une intégration plus fine entre bases de données géographiques et observations satellitaires, contribuant ainsi à la modernisation des outils de suivi et de gestion des ressources forestières.

# 

# Données et matériel

### Les données utilisées

Pour ce projet, nous avons utilisé des images Sentinel-2, acquises entre janvier 2022 et février 2023, représentant la tuile T31TCJ couvrant Toulouse et ses environs. Nous avons également utilisé la BD Forêt V2.0 pour obtenir des échantillons de référence, permettant d'effectuer une classification supervisée des essences forestières. Ces données ont été enrichies avec des données vectorielles pour la délimitation des zones d'étude.

### L’environnement de travail

Le projet a été réalisé sur une instance Onyxia, préconfigurée avec les librairies Python nécessaires pour le traitement des données, notamment geopandas, rasterio, scikit-learn, matplotlib et numpy. L'utilisation de Jupyter Notebook ou de VS Code sur cette instance a permis de faciliter le développement, la visualisation et la validation des résultats.

# Méthodologie

Pour l'analyse des peuplements forestiers, nous avons suivi une démarche systématique visant à garantir la reproductibilité des résultats obtenus. Cette approche inclut des étapes clés allant du pré-traitement des données à la classification supervisée, en passant par la validation et l'analyse des résultats. Voici les étapes principales, à retrouver également dans le diagramme de flux des traitements ci-dessous :

1. **Préparation des couches d'entrée**
2. **Pré-traitement des données**
3. **Génération des échantillons d'apprentissage**
4. **Classification supervisée**
5. **Classification des peuplements forestiers**
6. **Analyse de la qualité des classifications** .

Un diagramme de flux est également inclus pour détailler chaque traitement effectué, les couches d'entrée utilisées, ainsi que les paramètres principaux appliqués à chaque étape. Ce diagramme permet d'avoir une vue d'ensemble claire et précise sur le flux de travail et les interactions entre les différentes étapes de traitement. L'objectif est de garantir la transparence et la reproductibilité du processus.

INTÉGRER LE DIAGRAMME DE FLUX ICI

<https://docs.google.com/presentation/d/1h5WQr-NFv46eGNyQOCDjq8lSvGqw1Wb0z0Ksgm6c6Sk/edit#slide=id.p>

### Préparation des données

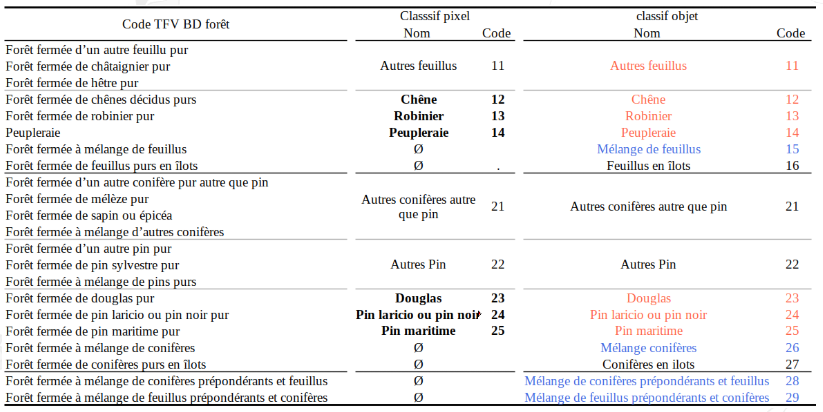
Les images Sentinel-2 ont été prétraitées afin d'harmoniser leur résolution spatiale, leur projection et leur découpage selon l'emprise d'étude. Ce prétraitement comprenait également la conversion des images en réflectance et la correction atmosphérique.

Le prétraitement a été réalisé pour s'assurer que toutes les images soient comparables en termes de résolution (10 m) et de projection (EPSG:2154 - Lambert 93). La correction atmosphérique a été appliquée pour réduire les effets des conditions d'acquisition (e.g., humidité de l'air, angles solaires), améliorant ainsi la précision des indices calculés comme le NDVI. Les images ont ensuite été découpées selon l'emprise de la zone d'étude afin de se concentrer uniquement sur les zones forestières pertinentes.

### Sélection et préparation des échantillon

À partir de la BD Forêt, des polygones représentatifs ont été extraits en respectant les classes pertinentes définies dans la nomenclature. Un fichier vecteur a été généré contenant les échantillons filtrés, destinés à l'entraînement et à la validation du modèle.

Les classes non pertinentes, telles que les landes, formations herbacées et forêts ouvertes, ont été exclues pour se concentrer sur les peuplements arborés. Les polygones retenus ont été transformés en échantillons utilisables dans le modèle de classification. Chaque polygone était lié à des informations attributaires permettant de connaître l'essence dominante et la densité de couvert, ce qui constitue la base de l'étiquetage pour la classification supervisée.



###### Figure 1 - Correspondance entre les classes de la BD Forêt et les nomenclatures des cartes à obtenir

### 

| #Fonction pour créer des échantillons depuis la BD Forêt  def create\_samples(bd\_forest\_path, output\_path):  bd\_forest = gpd.read\_file(bd\_forest\_path)  selected\_classes = bd\_forest[bd\_forest['TFV'].notna()] #Ajouter des classes spécifiques en fonction des exigences du projet  selected\_classes['Code'] = selected\_classes.index #En supposant qu'un identifiant unique soit l'index  selected\_classes.to\_file(output\_path)  #Fonction pour tracer un graphique en barres (décomptes d’échantillons)  def plot\_sample\_counts(samples\_gdf, class\_column, output\_path):  counts = samples\_gdf[class\_column].value\_counts()  counts.plot(kind='bar')  plt.xlabel('Classes')  plt.ylabel('Number of Samples')  plt.title('Number of Samples per Class')  plt.savefig(output\_path)  plt.close()  #Fonction pour tracer un graphique en barres du nombre de pixels par classe  def plot\_pixels\_by\_class(samples\_gdf, pixel\_counts\_column, output\_path):  counts = samples\_gdf[pixel\_counts\_column].value\_counts()  counts.plot(kind='bar')  plt.xlabel('Classes')  plt.ylabel('Number of Pixels')  plt.title('Number of Pixels per Class')  plt.savefig(output\_path)  plt.close()  # Fonction pour produire un “violin plot” de la distribution du nombre de pixels par polygone et par classe  def plot\_violin\_pixels\_per\_polygon(samples\_gdf, class\_column, pixel\_counts\_column, output\_path):  plt.figure(figsize=(10, 6))  sns.violinplot(x=class\_column, y=pixel\_counts\_column, data=samples\_gdf)  plt.xlabel('Classes')  plt.ylabel('Number of Pixels per Polygon')  plt.title('Distribution of Number of Pixels per Polygon by Class')  plt.savefig(output\_path)  plt.close() |
| --- |

###### Script 1 - Création et préparation de l’échantillon

###### 

###### 

| #Fonction pour calculer le NDVI  def calculate\_ndvi(nir\_band, red\_band):  nir = nir\_band.astype(np.float32)  red = red\_band.astype(np.float32)  ndvi = (nir - red) / (nir + red)  return ndvi |
| --- |

###### Script 2 - Calcul du NDVI

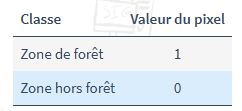
### Construction du masque forestier

Une étape clé de la méthodologie a été la construction d'un masque raster à partir de la BD Forêt afin de distinguer les zones à classer (zones forestières) des zones à ne pas classer (zones non forestières). Pour cela, nous avons exclu les polygones de type **lande**, **formation herbacée**, **forêt ouverte**, ainsi que la classe **forêt fermée sans couvert arboré**. Ces types de formations végétales ne correspondent pas aux peuplements forestiers pertinents pour notre analyse.

Le masque a été construit avec les caractéristiques suivantes :

* Format raster **GeoTIFF**, encodé en **8 bits**.
* Il possède la même emprise spatiale et la même résolution spatiale que les images Sentinel-2 prétraitées (c'est-à-dire une résolution de 10 m).
* Le fichier masque a été nommé **masque\_foret.tif** et stocké dans le répertoire **results/data/img\_pretraitees**.
* Le masque contient deux classes principales :
  + **Zone de forêt** : valeur du pixel = 1
  + **Zone hors forêt** : valeur du pixel = 0

Ce masque forestier est crucial pour garantir que seules les zones pertinentes soient incluses dans la classification supervisée. Les zones hors forêt ont été ignorées afin de limiter les erreurs liées à des classes non pertinentes, et ainsi améliorer la précision du modèle de classification. Le masque a été appliqué sur les images prétraitées afin de masquer les zones non pertinentes, ce qui a également réduit le temps de calcul nécessaire.



###### Tableau 1 - Nomenclature du masque forêt

| #Importation des librairies nécessaires  import os  import geopandas as gpd  import rasterio  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from rasterio.features import rasterize  from rasterio.mask import mask  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import StratifiedKFold, GroupKFold  from sklearn.metrics import accuracy\_score, confusion\_matrix  import joblib  #Fonction pour créer un masque sur la BD Forêt  def create\_forest\_mask(bd\_forest\_path, s2\_reference\_path, output\_path):  bd\_forest = gpd.read\_file(bd\_forest\_path)  with rasterio.open(s2\_reference\_path) as src:  out\_meta = src.meta.copy()  out\_meta.update({"driver": "GTiff", "dtype": "uint8"})    mask\_shapes = bd\_forest[~bd\_forest['TFV'].isin(['Lande', 'Formation Herbacée', 'Forêt Ouverte', 'Forêt Fermée Sans Couvert'])]  mask\_shapes = mask\_shapes.geometry.values    forest\_mask = rasterize(  [(geom, 1) for geom in mask\_shapes],  out\_shape=src.shape,  transform=src.transform,  fill=0,  dtype='uint8'  )    with rasterio.open(output\_path, 'w', \*\*out\_meta) as dest:  dest.write(forest\_mask, 1) |
| --- |

###### Script 3 - Importation des libraires et création du masque

### 

### Classification supervisée

Un modèle de classification supervisée a été mis en place, basé sur l'algorithme Random Forest de scikit-learn. Les échantillons d'apprentissage ont été utilisés pour entraîner le modèle, qui a ensuite été appliqué pour cartographier les essences forestières à l'échelle du pixel, puis les résultats ont été agrégés à l'échelle des peuplements.

Le modèle Random Forest a été choisi en raison de sa capacité à gérer des données complexes, non linéaires, et sa robustesse face aux problèmes de surapprentissage. Les hyperparamètres ont été optimisés pour maximiser la précision sans compromettre la généralisation. La classification à l'échelle du pixel a permis de capter les différences subtiles entre les essences, tandis que l'agrégation des résultats à l'échelle des peuplements a offert une vision simplifiée, mieux adaptée à certaines applications de gestion forestière.

| # Fonction de Classification Random Forest avec validation croisée  def random\_forest\_classification(X, y, n\_splits=5, groups=None, repeat=1):  clf = RandomForestClassifier(max\_depth=50, oob\_score=True, max\_samples=0.75, class\_weight='balanced', n\_jobs=-1)  if groups is None:  skf = StratifiedKFold(n\_splits=n\_splits)  else:  skf = GroupKFold(n\_splits=n\_splits)  scores = []  for train\_index, test\_index in skf.split(X, y):  X\_train, X\_test = X[train\_index], X[test\_index]  y\_train, y\_test = y[train\_index], y[test\_index]  clf.fit(X\_train, y\_train)  score = clf.score(X\_test, y\_test)  scores.append(score)  return np.mean(scores)  # Fonction pour entraînant et sauvegarder le modèle Random Forest  def train\_random\_forest(X, y, model\_output\_path):  clf = RandomForestClassifier(max\_depth=50, oob\_score=True, max\_samples=0.75, class\_weight='balanced', n\_jobs=-1)  clf.fit(X, y)  joblib.dump(clf, model\_output\_path)  return clf  # Fonction de classification des données matricielles à l’aide du modèle entraîné  def classify\_raster(model\_path, raster\_path, output\_path):  clf = joblib.load(model\_path)  with rasterio.open(raster\_path) as src:  img = src.read()  img\_reshaped = img.reshape((img.shape[0], -1)).T #Remodelé pour prédire  prediction = clf.predict(img\_reshaped)  prediction = prediction.reshape((src.height, src.width))  out\_meta = src.meta.copy()  out\_meta.update({"driver": "GTiff", "dtype": "uint8", "count": 1})  with rasterio.open(output\_path, 'w', \*\*out\_meta) as dest:  dest.write(prediction, 1)  # Fonction pour calculer la précision du modèle et la matrice de confusion  def evaluate\_model(X, y, clf):  y\_pred = clf.predict(X)  accuracy = accuracy\_score(y, y\_pred)  conf\_matrix = confusion\_matrix(y, y\_pred)  return accuracy, conf\_matrix |
| --- |

###### Script 4 - Fonction de classification Random Forest

### Stratégie de validation

Les performances du modèle ont été évaluées à travers des métriques de qualité, des matrices de confusion et une analyse des confusions remarquables. L'évaluation a été réalisée pour les classifications à l'échelle du pixel et des peuplements.

# Analyse des échantillons

L’analyse des échantillons a mis en lumière la répartition des classes forestières et leur représentativité. Un histogramme des polygones et des pixels par classe a révélé des disparités dans leur distribution, certaines classes étant surreprésentées. Les signatures temporelles NDVI ont permis d’identifier des comportements phénologiques distincts, essentiels pour différencier les classes. Par ailleurs, l’étude de la variabilité spectrale a montré que les classes en mélange présentaient une plus grande dispersion que les classes pures, confirmant la complexité de leur classification.

### Distribution des échantillons

L'analyse des échantillons a permis de mettre en évidence la répartition des classes forestières et leur représentativité. Un histogramme des polygones et des pixels par classe a révélé des disparités dans leur distribution, certaines classes étant surreprésentées.

### 

### Phénologie des classes forestières

Les signatures temporelles NDVI ont permis d'identifier des comportements phénologiques distincts, essentiels pour différencier les classes forestières. Les courbes de NDVI montrent des variations saisonnières caractéristiques pour chaque classe, facilitant leur distinction.

La phénologie des classes forestières a été analysée à travers les valeurs moyennes et les écarts types de NDVI pour chaque classe sur plusieurs dates. Ces analyses ont permis de distinguer des classes ayant des dynamiques saisonnières spécifiques, comme les feuillus qui présentent une forte variation de NDVI entre l'été et l'hiver, par rapport aux résineux qui ont un NDVI plus stable. Ces différences phénologiques ont été exploitées par le modèle de classification pour améliorer la précision de la distinction entre essences.

| #Fonction permettant de tracer la signature temporelle du NDVI  def plot\_ndvi\_temporal\_signature(ndvi\_data, class\_labels, output\_path):  plt.figure(figsize=(12, 8))  for class\_label in class\_labels:  class\_ndvi = ndvi\_data[ndvi\_data['class'] == class\_label]  mean\_ndvi = class\_ndvi.mean(axis=0)  std\_ndvi = class\_ndvi.std(axis=0)  plt.plot(mean\_ndvi.index, mean\_ndvi.values, label=f'Class {class\_label}')  plt.fill\_between(mean\_ndvi.index, mean\_ndvi - std\_ndvi, mean\_ndvi + std\_ndvi, alpha=0.2)  plt.xlabel('Date')  plt.ylabel('NDVI')  plt.title('NDVI Temporal Signature (Mean and Standard Deviation) per Class')  plt.legend()  plt.savefig(output\_path)  plt.close() |
| --- |

###### Script 5 - Etude de la phénologie

### 

### Variabilité spectrale

L'étude de la variabilité spectrale a révélé que les classes en mélange présentaient une plus grande dispersion que les classes pures, ce qui complique leur classification. La distance moyenne au centroïde a été utilisée pour mesurer cette variabilité, confirmant la complexité des classes mixtes.

Pour mesurer la variabilité spectrale, la distance moyenne au centroïde des pixels appartenant à chaque classe a été calculée. Cette mesure permet d'évaluer la cohérence interne de chaque classe : plus la distance est faible, plus les pixels d'une même classe sont spectrally similaires. Les résultats ont montré que les classes en mélange avaient des distances au centroïde plus élevées, indiquant une plus grande hétérogénéité, ce qui rend leur classification plus difficile. Cela a été particulièrement vrai pour les peuplements mixtes, où la présence de plusieurs essences entraîne une diversité spectrale accrue.

| #Fonction pour calculer et tracer la distance du centroïde par polygone  def plot\_centroid\_distance\_by\_polygon(features, class\_labels, output\_path):  polygon\_distances = []  for class\_label in class\_labels:  class\_features = features[features['class'] == class\_label]  for \_, polygon in class\_features.iterrows():  polygon\_features = polygon.iloc[1:].values.reshape(1, -1)  centroid = calculate\_class\_centroids(polygon\_features)  avg\_distance = calculate\_average\_distance\_to\_centroid(polygon\_features, centroid)  polygon\_distances.append({'class': class\_label, 'distance': avg\_distance})    distances\_df = gpd.pd.DataFrame(polygon\_distances)  plt.figure(figsize=(12, 8))  sns.violinplot(x='class', y='distance', data=distances\_df)  plt.xlabel('Class')  plt.ylabel('Average Distance to Centroid')  plt.title('Average Distance to Centroid per Polygon by Class')  plt.savefig(output\_path)  plt.close()  #Fonction pour calculer et tracer la distance entre les centroïdes pour chaque classe  def plot\_centroid\_distance\_by\_class(features, class\_labels, output\_path):  centroid\_distances = {}  for class\_label in class\_labels:  class\_features = features[features['class'] == class\_label].iloc[:, 1:].values  centroid = calculate\_class\_centroids(class\_features)  avg\_distance = calculate\_average\_distance\_to\_centroid(class\_features, centroid)  centroid\_distances[class\_label] = avg\_distance    plt.figure(figsize=(10, 6))  plt.bar(centroid\_distances.keys(), centroid\_distances.values())  plt.xlabel('Class')  plt.ylabel('Average Distance to Centroid')  plt.title('Average Distance to Centroid per Class')  plt.savefig(output\_path)  plt.close() |
| --- |

###### Script 6 - Calcul des centroïdes

# Résultats

### Analyse des échantillons

### 

**Pré-traitement des données**

| # Fonction pour créer une image matricielle multi-bandes à partir de plusieurs images  def create\_multiband\_raster(image\_paths, emprise\_path, mask\_path, output\_path, bands, dtype='uint16'):  emprise = gpd.read\_file(emprise\_path)  with rasterio.open(image\_paths[0]) as src:  out\_meta = src.meta.copy()  out\_meta.update({  "driver": "GTiff",  "dtype": dtype,  "count": len(bands) \* len(image\_paths),  "crs": "EPSG:2154",  "transform": src.transform,  "nodata": 0  })    with rasterio.open(output\_path, 'w', \*\*out\_meta) as dest:  for i, image\_path in enumerate(image\_paths):  with rasterio.open(image\_path) as src:  for j, band in enumerate(bands):  band\_data = src.read(band)  masked, \_ = mask(src, emprise.geometry, crop=True)  with rasterio.open(mask\_path) as mask\_src:  mask\_data = mask\_src.read(1)  band\_data[mask\_data == 0] = 0  dest.write(band\_data, i \* len(bands) + j + 1)  # Fonction pour créer un raster de séries temporelles NDVI  def create\_ndvi\_timeseries(image\_paths, emprise\_path, mask\_path, output\_path, dtype='float32'):  emprise = gpd.read\_file(emprise\_path)  with rasterio.open(image\_paths[0]) as src:  out\_meta = src.meta.copy()  out\_meta.update({  "driver": "GTiff",  "dtype": dtype,  "count": len(image\_paths),  "crs": "EPSG:2154",  "transform": src.transform,  "nodata": -9999  })    with rasterio.open(output\_path, 'w', \*\*out\_meta) as dest:  for i, image\_path in enumerate(image\_paths):  with rasterio.open(image\_path) as src:  red\_band = src.read(3) # En supposant que la bande 3 soit rouge  nir\_band = src.read(8) # En supposant que la bande 8 soit NIR  ndvi = calculate\_ndvi(nir\_band, red\_band)  masked, \_ = mask(src, emprise.geometry, crop=True)  with rasterio.open(mask\_path) as mask\_src:  mask\_data = mask\_src.read(1)  ndvi[mask\_data == 0] = -9999  dest.write(ndvi, i + 1) |
| --- |

###### Script 7 - Image matricielle multi-bandes et raster de séries temporelles

### Cartes produites

Les cartes produites offrent une vision détaillée des essences forestières à deux échelles :

* **À l’échelle du pixel** : La classification Random Forest a permis de distinguer les différentes essences avec une précision globale satisfaisante, bien que certaines confusions aient été observées entre classes proches.

| # Fonction permettant de classer les données matricielles et de produire une carte des essences forestières  def produce\_forest\_species\_map(X, y, raster\_path, model\_output\_path, output\_path):  # Entrainement du modèle  clf = train\_random\_forest(X, y, model\_output\_path)  # Classifier les données matricielles à l'aide du modèle entraîné  classify\_raster(model\_output\_path, raster\_path, output\_path) |
| --- |

###### Script 8 - Carte des essences forestières

* **À l’échelle des peuplements** : L’agrégation des résultats a simplifié la lecture des cartes tout en reflétant la composition dominante des peuplements.

| # Fonction de classification des peuplements forestiers basée sur des règles de décision  def classify\_forest\_stands(samples\_gdf, pixel\_class\_counts, output\_path):  # Appliquer des règles de décision basées sur la figure 4 pour classer chaque polygone  def classify\_polygon(area, class\_proportions):  if area < 2:  if class\_proportions['Feuillus'] >= 0.75:  if class\_proportions['Conifères'] >= 0.75:  return 'Conifères en îlots'  return 'Feuillus en îlots'  elif class\_proportions['Conifères'] >= 0.75:  return 'Conifères en îlots'  elif class\_proportions['Conifères'] >= class\_proportions['Feuillus']:  return 'Mélange de conifères prépondérants et feuillus'  else:  return 'Mélange de feuillus prépondérants et conifères'  else:  if max(class\_proportions.values()) >= 0.75:  return max(class\_proportions, key=class\_proportions.get)  elif class\_proportions['Feuillus'] > class\_proportions['Conifères']:  return 'Mélange de feuillus prépondérants et conifères'  elif class\_proportions['Conifères'] > class\_proportions['Feuillus']:  return 'Mélange de conifères prépondérants et feuillus'  else:  return 'Mélange conifères et feuillus  # Ajouter des prédictions à chaque polygone  samples\_gdf['area\_ha'] = samples\_gdf.geometry.area / 10000 # Convert area to hectares  samples\_gdf['code\_predit'] = samples\_gdf.apply(  lambda row: classify\_polygon(  row['area\_ha'],  pixel\_class\_counts.get(row['Code'], {'Feuillus': 0, 'Conifères': 0})  ),  axis=1  )    samples\_gdf.to\_file(output\_path)  # Fonction permettant de créer une matrice de confusion pour la classification des peuplements forestiers  def create\_confusion\_matrix(samples\_gdf, output\_path):  y\_true = samples\_gdf['Code']  y\_pred = samples\_gdf['code\_predit']  conf\_matrix = confusion\_matrix(y\_true, y\_pred, labels=y\_true.unique())  plt.figure(figsize=(10, 8))  sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=y\_true.unique(), yticklabels=y\_true.unique())  plt.xlabel('Predicted')  plt.ylabel('True')  plt.title('Confusion Matrix for Forest Stand Classification')  plt.savefig(output\_path)  plt.close() |
| --- |

###### Script 9 - Carte à l’échelle des peuplements

Les matrices de confusion ont révélé des performances variées selon les classes, mettant en avant des forces (classes pures bien identifiées) et des limites (confusions pour les classes en mélange).

Les cartes générées à l'échelle du pixel montrent une variabilité importante dans la répartition des essences, tandis que celles à l'échelle des peuplements fournissent une vue d'ensemble plus synthétique. Ces cartes ont été validées à l'aide d'échantillons de validation indépendants, et les résultats ont été évalués en termes de précision, rappel, et score F1.

### 

### Qualité des classifications

Les matrices de confusion ont révélé des performances variées selon les classes, mettant en avant des forces (classes pures bien identifiées) et des limites (confusions pour les classes en mélange). La précision globale de la classification pixel était satisfaisante, avec des confusions logiques entre classes similaires.

Certaines classes, telles que les forêts de résineux pures, ont été bien distinguées en raison de leur signature spectrale unique, tandis que d'autres, notamment les peuplements mixtes, ont présenté des taux de confusion élevés. Les confusions les plus fréquentes ont été observées entre les feuillus et les peuplements mixtes contenant une forte proportion de feuillus. Ces résultats suggèrent qu'une amélioration des données d'entrée ou une approche de classification plus avancée pourrait être nécessaire pour réduire ces confusions.

| # Fonction d'analyse de la qualité de la classification  def analyze\_classification\_quality(samples\_gdf, output\_path):  y\_true = samples\_gdf['Code']  y\_pred = samples\_gdf['code\_predit']  report = classification\_report(y\_true, y\_pred, labels=y\_true.unique(), output\_dict=True)  accuracy = accuracy\_score(y\_true, y\_pred)    # Analyse de la précision et de la qualité du tracé  fig, ax = plt.subplots(figsize=(10, 8))  ax.bar(report.keys(), [report[label]['f1-score'] for label in report.keys() if label != 'accuracy'])  plt.xlabel('Classes')  plt.ylabel('F1 Score')  plt.title(f'Classification Quality per Class (Accuracy: {accuracy:.2f})')  plt.xticks(rotation=45)  plt.tight\_layout()  plt.savefig(output\_path)  plt.close() |
| --- |

###### Script 10 - Analyse de la qualité des classifications

# 

# Analyse critique

### Discussion

Les résultats obtenus montrent des performances globalement satisfaisantes pour certaines classes, tandis que d'autres présentent des confusions notables. Ces confusions peuvent être attribuées à plusieurs facteurs :

1. **Similitudes spectrales** : Certaines classes d'essences, telles que les mélanges de feuillus et de conifères, présentent des signatures spectrales très similaires, ce qui rend leur différenciation difficile. Cela a entraîné des confusions, en particulier dans les zones où les proportions d'essences sont équilibrées. Par exemple, la confusion entre les mélanges de conifères et de feuillus est cohérente avec les caractéristiques spectrales proches de ces types de végétation.
2. **Hétérogénéité** : La variabilité intra-classe peut également expliquer certaines erreurs de classification. Par exemple, certaines classes comme les peuplements mixtes peuvent présenter une grande diversité de conditions locales (densité, type de sous-bois, etc.), ce qui complique la tâche du modèle.
3. **Qualité des données** : La qualité des données d'entrée, notamment la couverture nuageuse résiduelle et la résolution spatiale des images, a pu limiter la précision de la classification. Des nuages résiduels ou des ombres peuvent altérer les valeurs des indices spectrales, entraînant des erreurs de classification.

### Limites et pistes d’amélioration

Plusieurs axes d’amélioration et de développement peuvent être envisagés :

* **Règles de décision** : Les règles de décision utilisées pour la classification des peuplements forestiers sont basées sur des seuils fixes. Ces règles pourraient être affinées pour mieux refléter la diversité des peuplements, par exemple en utilisant des approches basées sur des critères plus adaptatifs ou des méthodes d'apprentissage automatique plus avancées.
* **Enrichissement des données** : L'utilisation de données complémentaires, telles que des données LiDAR pour une meilleure estimation de la structure verticale de la végétation, pourrait permettre d'améliorer la qualité de la classification.
* **Stratégie d'échantillonnage** : La qualité des échantillons d'apprentissage a un impact direct sur la performance du modèle. Une stratégie d'échantillonnage plus représentative des différentes classes, en tenant compte de la variabilité intra-classe, pourrait améliorer la robustesse du modèle.

L'utilisation de données multispectrales supplémentaires, telles que des observations infrarouges thermiques, pourrait également améliorer la distinction entre certaines classes, notamment en période hivernale. En outre, des approches de fusion de données pourraient être explorées pour combiner des images optiques et radar, offrant une meilleure résilience aux conditions atmosphériques défavorables.

Les cartes produites sont utilisables dans le contexte de la gestion forestière, notamment pour le suivi des peuplements et la planification des interventions sylvicoles. Elles peuvent également être utiles pour des études d'impact environnemental, en fournissant une vue d'ensemble des essences présentes et de leur répartition spatiale.

Cependant, leur utilisation doit être faite avec prudence, en tenant compte des limites évoquées. Les résultats pourraient être particulièrement utiles pour une première évaluation à grande échelle, mais des vérifications sur le terrain seraient nécessaires pour valider les classifications dans des contextes spécifiques.

Enfin, ces travaux ouvrent des perspectives intéressantes pour l'amélioration des méthodes de classification des peuplements forestiers, notamment par l'intégration de nouvelles sources de données et l'utilisation de techniques d'apprentissage plus avancées, comme les réseaux de neurones convolutifs (CNN) pour une meilleure extraction des caractéristiques.

### 

# 

# Conclusion

La gestion des forêts, face aux défis environnementaux et aux besoins croissants en ressources, nécessite des outils d’observation et d’analyse précis et actualisés. Ce projet s’est inscrit dans cette dynamique en explorant l’utilisation de la **BD Forêt® version 2.0** comme base d’échantillons de référence pour la classification supervisée de séries temporelles d’images Sentinel-2.

À travers une méthodologie rigoureuse, intégrant le prétraitement des données, la sélection des échantillons, la classification supervisée et l’analyse critique des résultats, plusieurs objectifs ont été atteints :

1. **La production de cartes d’essences forestières à l’échelle du pixel et à l’échelle des peuplements.**
2. **L’évaluation de la qualité des classifications, mettant en évidence les forces et limites des approches utilisées.**

Les résultats obtenus ont montré que la combinaison des données géographiques de la BD Forêt avec les capacités d’observation de Sentinel-2 permet de produire des cartes de qualité, bien qu’avec certaines limitations :

* **Les classes aux signatures spectrales proches ont entraîné des confusions logiques mais pénalisantes pour la précision des classifications.**
* **La variabilité intra-classe a parfois diminué la qualité des prédictions, notamment pour les peuplements mixtes.**

Malgré ces défis, la démarche a permis de démontrer l’intérêt de coupler bases de données géographiques et imagerie satellite pour une cartographie fine et actualisée des essences forestières. Les analyses phénologiques et spectrales menées ont également enrichi la compréhension des dynamiques végétales, ouvrant la voie à des applications futures.

Ce projet illustre bien le potentiel des outils de télédétection pour améliorer la gestion forestière et l’aménagement du territoire. Les résultats obtenus, bien que perfectibles, représentent une contribution significative à la modernisation des approches actuelles, répondant aux besoins des acteurs de la filière forêt-bois et de l’environnement.

# Annexes

### Annexe 1 - Utilité des scripts

| **my\_function.py**  Ce fichier regroupe toutes les fonctions réutilisables pour les autres scripts. Il est le cœur de la modularité du projet, permettant de structurer le code et de simplifier les appels dans les autres scripts.  **Utilisation :**   * Traiter les données géospatiales (filtrage des classes, création de masques, calculs statistiques). * Effectuer des analyses spécifiques (calcul des distances au centroïde, NDVI, etc.). * Gérer la classification supervisée (entraînement du modèle, validation croisée). * Produire des graphiques et visualisations. |
| --- |

| **sample\_curation.py**  Ce script sert à sélectionner et préparer les échantillons d’apprentissage et de validation à partir de la BD Forêt.  **Utilisation :**   * Filtrer les polygones pertinents en fonction des classes d’intérêt. * Générer un fichier vecteur contenant uniquement les échantillons valides. * Produire le fichier Sample\_BD\_foret\_T31TCJ.shp pour les étapes suivantes du projet.   **Sortie :** Sample\_BD\_foret\_T31TCJ.shp, un fichier contenant les échantillons sélectionnés. |
| --- |

| **sample\_analysis\_nb\_sample.py**  Analyser la répartition des échantillons et produire des graphiques descriptifs.  **Utilisation :**   * Compter le nombre de polygones et de pixels par classe. * Visualiser la répartition des échantillons sous forme de diagrammes en bâton. * Identifier d’éventuelles classes surreprésentées ou sous-représentées.   **Sorties :**   * diag\_baton\_nb\_poly\_by\_class.png : Histogramme du nombre de polygones par classe. * diag\_baton\_nb\_pix\_by\_class.png : Histogramme du nombre de pixels par classe. |
| --- |

| **sample\_analysis\_temp\_signature.py**  Étudier la phénologie des classes forestières en analysant leurs signatures temporelles NDVI.  **Utilisation :**   * Calculer les valeurs moyennes et les écarts types de NDVI pour chaque classe. * Produire des graphiques montrant les évolutions temporelles du NDVI par classe. * Identifier des tendances phénologiques permettant de distinguer les classes.   **Sortie :** temp\_mean\_ndvi.png, un graphique illustrant la variation temporelle du NDVI moyen par classe. |
| --- |

| **sample\_analysis\_spectral\_variability.py**  Analyser la variabilité spectrale des classes et comparer les classes pures et mixtes.  **Utilisation :**   * Calculer la distance moyenne au centroïde des pixels pour chaque classe. * Visualiser la variabilité spectrale à l’échelle de l’image entière et des polygones individuels. * Identifier si les classes mixtes présentent une plus grande variabilité spectrale que les classes pures.   **Sorties :**   * diag\_baton\_dist\_centroide\_classe.png : Histogramme de la distance moyenne au centroïde par classe. * violin\_plot\_dist\_centroide\_by\_poly\_by\_class.png : Violin plot de la variabilité spectrale à l’échelle des polygones. |
| --- |

| **pre\_traitement.py**  Préparer les données d’entrée, notamment les images Sentinel-2, pour les étapes de classification.  **Utilisation :**   * Harmoniser la résolution et la projection des bandes Sentinel-2. * Découper les images selon l’emprise d’étude et appliquer un masque de forêt. * Produire deux jeux de données :   + Une image multi-bandes (Serie\_temp\_S2\_allbands.tif).   + Une série temporelle d’images NDVI (Serie\_temp\_S2\_ndvi.tif).   **Sorties :**   * Serie\_temp\_S2\_allbands.tif : Image contenant toutes les bandes spectrales pour les 6 dates. * Serie\_temp\_S2\_ndvi.tif : Image représentant les NDVI pour les 6 dates. |
| --- |

| **classification\_pixel.py**  Produire une carte des essences forestières à l’échelle du pixel à partir d’une classification supervisée.  **Utilisation :**   * Entraîner un modèle Random Forest sur les échantillons sélectionnés. * Appliquer le modèle sur l’image multi-bandes Sentinel-2. * Générer une carte raster représentant les essences forestières à l’échelle du pixel.   **Sortie :** carte\_essences\_echelle\_pixel.tif, une carte des essences forestières à l’échelle du pixel. |
| --- |

| **classification\_stand.py**  Produire une carte des peuplements forestiers en agrégeant les résultats de la classification pixel.  **Utilisation :**   * Calculer la composition des classes de pixels pour chaque polygone de la BD Forêt. * Appliquer des règles décisionnelles pour attribuer une classe dominante à chaque polygone. * Mettre à jour le fichier vecteur avec les nouvelles classifications.   **Sortie :** Sample\_BD\_foret\_T31TCJ\_updated.shp, un fichier vecteur avec les classes prédominantes attribuées à chaque peuplement. |
| --- |

* Diagramme de flux des traitements
* Extraits de code Python (sélectionnés)
* Figures et tableaux complémentaires
* Bibliographie et ressources utilisées

### Annexe 2 - Script complet, mis bout à bout