



UFR 6

Université Paul Valéry, Montpellier III

Projet TER

Rendu 5 - Outils, Logiciels et Modélisation

Alvin VEDEL, Florian DUBOIS, Matis BREILLAD

10 Février 2024

Résumé

Au cours du premier semestre, nous avons exploré certaines données mises à notre disposition pour entraîner divers modèles et obtenir nos premiers résultats. Dans ce rapport, nous détaillerons les outils et logiciels que nous avons utilisés, ainsi que les caractéristiques des modèles employés. Ensuite, nous aborderons les futurs outils et modèles que nous prévoyons d'utiliser pour traiter d'autres type de données et améliorer nos résultats.

Abstract

During the first half of the year, we explored some of the data provided to us in order to train various models and obtain our first results. In this report, we will detail the tools and software we used, as well as the characteristics of the models employed. We will then discuss the future tools and models that we plan to use to process other types of data and improve our results.

Table des matières

| | |
|--|-----------|
| Résumé | i |
| Table des figures | ii |
| 1 Organisation du semestre | 1 |
| 1.1 Outils et Programmes | 2 |
| 1.1.1 Python | 2 |
| 1.1.2 R | 2 |
| 1.1.3 Visusalisation | 2 |
| 1.1.4 Outils collaboratifs | 2 |
| 1.2 Données | 3 |
| 1.2.1 Données initiales | 3 |
| 1.2.2 Données produites et modèles | 3 |
| 1.2.3 Utiliser les autres données | 3 |
| 2 Modélisations | 4 |
| 2.1 Modèles mis en place | 5 |
| 2.1.1 Algorithme des K-voisins | 5 |
| 2.1.2 Random Forest | 5 |
| 2.1.3 SVM | 8 |
| 2.2 A mettre en place | 11 |
| 2.2.1 Multilayer perceptron | 11 |
| 2.2.2 Poisson | 11 |
| Conclusion | 12 |
| Bibliographie | 13 |

Table des figures

| | | |
|-----|--|----|
| 1.1 | Image matricielle environnementales | 3 |
| 2.1 | Arbres de Classification pour l'espèce 11 | 7 |
| 2.2 | Hitmap des résultats | 8 |
| 2.3 | Scores Kaggle SVM | 10 |
| 2.4 | Présence de l'espèce 6574 (<i>Hyracotherium leporinum</i>) | 11 |
| 2.5 | | 12 |

Chapitre 1

Organisation du semestre

Sommaire

| | |
|--|----------|
| 1.1 Outils et Programmes | 2 |
| 1.1.1 Python | 2 |
| 1.1.2 R | 2 |
| 1.1.3 Visusalisation | 2 |
| 1.1.4 Outils collaboratifs | 2 |
| 1.2 Données | 3 |
| 1.2.1 Données initiales | 3 |
| 1.2.2 Données produites et modèles | 3 |
| 1.2.3 Utiliser les autres données | 3 |

1.1 Outils et Programmes

1.1.1 Python

Python est le langage utilisé majoritairement. Il nous a permis de construire nos premières visualisations cartographiques et les libraires comme pandas, numpy et scikit learn ont permis la réalisation de nos premiers modèles de machine learning. Dans un second temps, c'est le framework PyTorch qui nous permettra d'évoluer avec des modèles de Deep Learning et des fonctionnalités propres à la librairie comme les tensors ou le calcul de gradient (autograd, forward, ...)

Python est majoritairement utilisé via Anaconda navigator et plus précisément Jupyter Notebook pour rendre l'exécution du code plus simple.

1.1.2 R

R utilisé via l'interface Rstudio nous a permis d'analyser et visualiser les données, il permet également la création de modèles comme celui des SVM.

1.1.3 Visusalisation

D3js est la librairie javascript utilisée pour créer un outil de visualisation interactif qui permettra de présenter les données à notre disposition lors de la soutenance TER. L'objectif est également d'utiliser nos modèles prédictifs de manière simplifiée (clic sur carte) dans un cadre hors de celui de Kaggle afin que l'utilité de notre projet soit plus concrète.

1.1.4 Outils collaboratifs

Github permet le partage de document entre les membres du groupe tout en versionnant notre projet, ce qui permet de sauvegarder notre avancée et de revenir à une version antérieure si besoin. Le projet le plus à jour est toujours accessible offrant un meilleur travail d'équipe. La gestion de projet est également assurée par GitHub, le suivi des tâches effectuées, en cours et planifiées permet aux membres du groupe de connaître les objectifs restants et ceux sur lesquels chacun travaille (meilleur cohésion d'équipe). La création d'issues permet aussi de signaler des problèmes ou des tâches prioritaires afin d'accentuer l'attention sur certains points.

Overleaf nous permet de travailler sur les rendus périodiques du TER de manière collaboritive, cet outil en ligne assure une qualité de rendu adaptée à nos besoins grâce au langage Latex.

Discord permet la communication entre les membres du groupe tout en archivant les discussions, l'envoi de fichier est également possible ce qui rend l'outil très utile.

1.2 Données

1.2.1 Données initiales

Les données à notre disposition sont disponibles sur le Kaggle du projet sous forme de fichiers csv ou d'archives zip contenant des csv. Une partie des données a été dupliquée sur Github pour y faciliter l'accès, quant aux fichiers trop volumineux ils ont été téléchargés individuellement et sont toujours accessible sur le Seafire Repository (stockage des données Kaggle).

1.2.2 Données produites et modèles

Des transformations sur les fichiers ont été effectuées pour répondre à nos besoins, les documents csv nouvellement créés ont été déposés sur github également si cela était possible. Le cas contraire, les codes ayant permis la transformation sont disponibles et ré-exécutables par chacun. Les notebooks Python et R donnant lieu à la création de modèles ou de visualisation sont entièrement accessibles sur le Repository du groupe de TER afin que chaque avancée soit reproductible.

1.2.3 Utiliser les autres données

PyTorch permettra par la suite d'intégrer les données d'images matricielles (rasters) puis les séries temporelles.

C. Environmental rasters

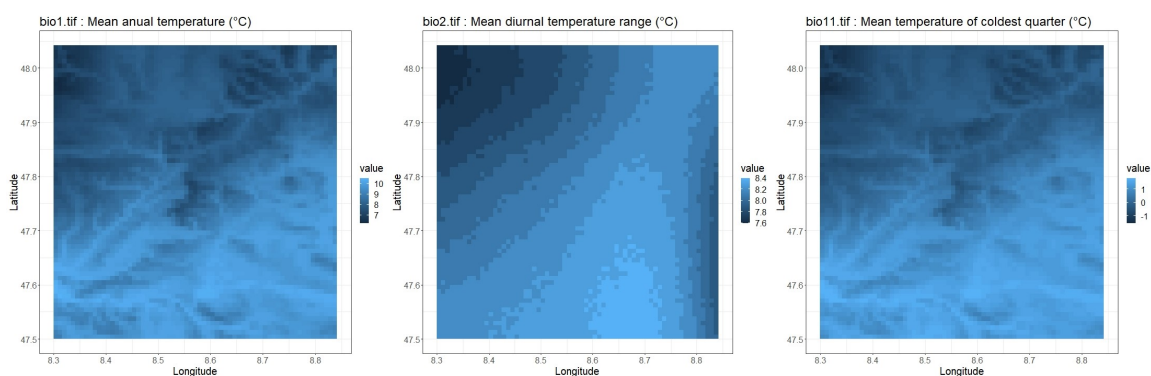


FIGURE 1.1 – Image matricielle environnementales

Les données de présences uniquement seront plus complexes à intégrer compte tenu de l'ambiguïté d'une observation : présence oui mais pas d'informations sur l'absence des autres espèces.

Chapitre 2

Modélisations

Sommaire

| | | |
|------------|---------------------------------------|-----------|
| 2.1 | Modèles mis en place | 5 |
| 2.1.1 | Algorithme des K-voisins | 5 |
| 2.1.2 | Random Forest | 5 |
| 2.1.3 | SVM | 8 |
| 2.2 | A mettre en place | 11 |
| 2.2.1 | Multilayer perceptron | 11 |
| 2.2.2 | Poisson | 11 |

2.1 Modèles mis en place

2.1.1 Algorithme des K-voisins

L'algorithme des K-voisins est le premier que nous avons implémenter et sur lequel nous avons obtenu un score qui nous a paru satisfaisant. Il paraît particulièrement judicieux dans le cas de la prédiction d'espèces végétal étant donné qu'une espèce présente a un point x peut parfaitement être présente dans le voisinage de ce point. Nous avons donc commencer par les voisins géographique puis selon les différentes variables abiotiques avec le quel nous avons eu des résultats moindres.

Équations mathématiques :

Dans le cas général, l'équation pour un algorithme de k-voisins s'écrit comme suit :

$$\hat{y} = \arg \max_y \sum_{k=1}^K \mathbb{1}(y_k = y)$$

Avec K le nombre de voisins considérés, et y_k la valeur du voisins voisin k .

Cependant cela correspond au cas où l'on souhaite prédire une seule classe pour un individu. Dans notre cas on souhaite prédire plusieurs espèces pour un individus.

$$\hat{y} = \mathbb{1}[\frac{1}{k} \sum_{k=1}^K \sum_{i=1}^I \mathbb{1}(p_{ki} = i) > Seuil]$$

Avec p_{ki} l'espèce i présente dans le voisin k et le seuil que l'on fixe comme étant la fréquence d'apparition de l'espèce dans les voisins à dépasser pour qu'elle soit prédite.

Score et Choix des hyperparamètres :

Nous avons donc deux paramètres à faire varier afin d'améliorer le modèle. K le nombre de voisins que l'on considère et le Seuil qui détermine à quel point nous allons prédire des espèces. Nous avons donc essayé plusieurs valeurs et on a sélectionner en fonction du score public obtenu sur kaggle.

2.1.2 Random Forest

Nous avons employé l'algorithme des forêts aléatoires pour analyser nos données abiotiques. Avec près de 6000 patchID (zones géographiques) distincts, chaque patch étant caractérisé par 19 variables climatiques (telles que les températures moyennes ou maximales) et 20 variables relatives au sol (comme la composition en azote ou la présence d'une route). De plus, chaque patchID peut présenter la présence ou l'absence de plus de 2000 espèces.

Équations mathématiques :

L'algorithme des forêts aléatoires utilise le principe du bagging pour construire plusieurs arbres de décision à partir d'ensembles de données légèrement différents et ainsi obtenir de meilleures performances.

Représentons notre jeu de données d'entraînements pour une seule espèce par $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_{6000}, y_{6000})\}$ où x_i est la matrice des coefficients des paramètres pour le patchID i et y_i (1 ou 0) indique si l'espèce est présente ou non.

On va construire k arbres de décision en utilisant le bootstrap. Cela signifie que chaque arbre sera construit à partir d'un échantillon

$$D_i^* = \{(x_{i_1}^*, y_{i_1}^*), (x_{i_2}^*, y_{i_2}^*), \dots, (x_{i_{6000}}^*, y_{i_{6000}}^*)\}$$

de l'ensemble d'entraînement D . Cela permet d'obtenir une grande variété d'arbres, ce qui contribue à corriger les erreurs qu'un seul arbre pourrait avoir et réduit le risque de surajustement du modèle.

La construction d'un arbre de décision passe par un algorithme récursif qui va diviser notre ensemble D_i^* en plusieurs sous ensembles pour isoler les individus où $y_i = 1$ en fonction de certaines variables. On se base sur différents critères (Gini) pour évaluer lesquelles des variables séparent le mieux notre jeu de données.

Soit T_1, T_2, \dots, T_k les arbres de la forêt aléatoire, la prédiction finale \hat{y} pour une observation x peut être exprimée comme :

$$\hat{y} = \frac{1}{k} \sum_{i=1}^k T_i(x)$$

Où $T_i(x)$ est la prédiction de l'arbre T_i pour l'observation x .

Conditions d'application :

- Volume de données élevé
- Nombre de variables assez élevé
- Pas d'autres conditions particulières, l'algorithme peut traiter à la fois des variables quantitatives et qualitatives et est assez résistant à la présence de valeurs manquantes.

Visualisation :

Choix des hyperparamètres :

Il est possible de modifier plusieurs paramètres mais nous avons seulement tester différents modèles en modifiant le nombre d'arbre et le seuil minimum de probabilité d'appartenance, c'est à dire la probabilité minimal pour prédire qu'une espèce est présente.

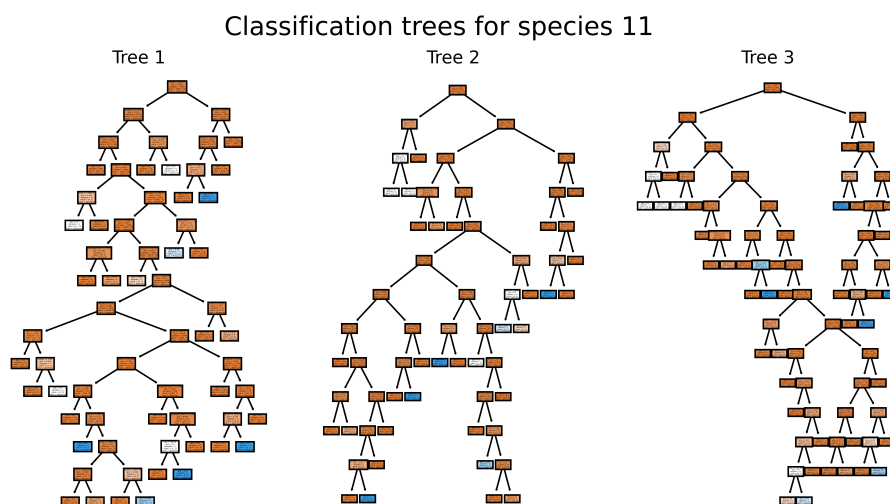


FIGURE 2.1 – Arbres de Classification pour l'espèce 11

Score :

Les meilleurs résultats sont atteints lorsque l'on augmente le nombre d'arbre de notre modèle (300) malgré que cela augmente le temps d'exécution de l'algorithme. Et lorsque que le seuil minimal pour prédire la présence est fixé à 0.235, ce qui assez faible mais s'explique par la pénalité importante imposée lorsque qu'une espèce n'est pas prédite.

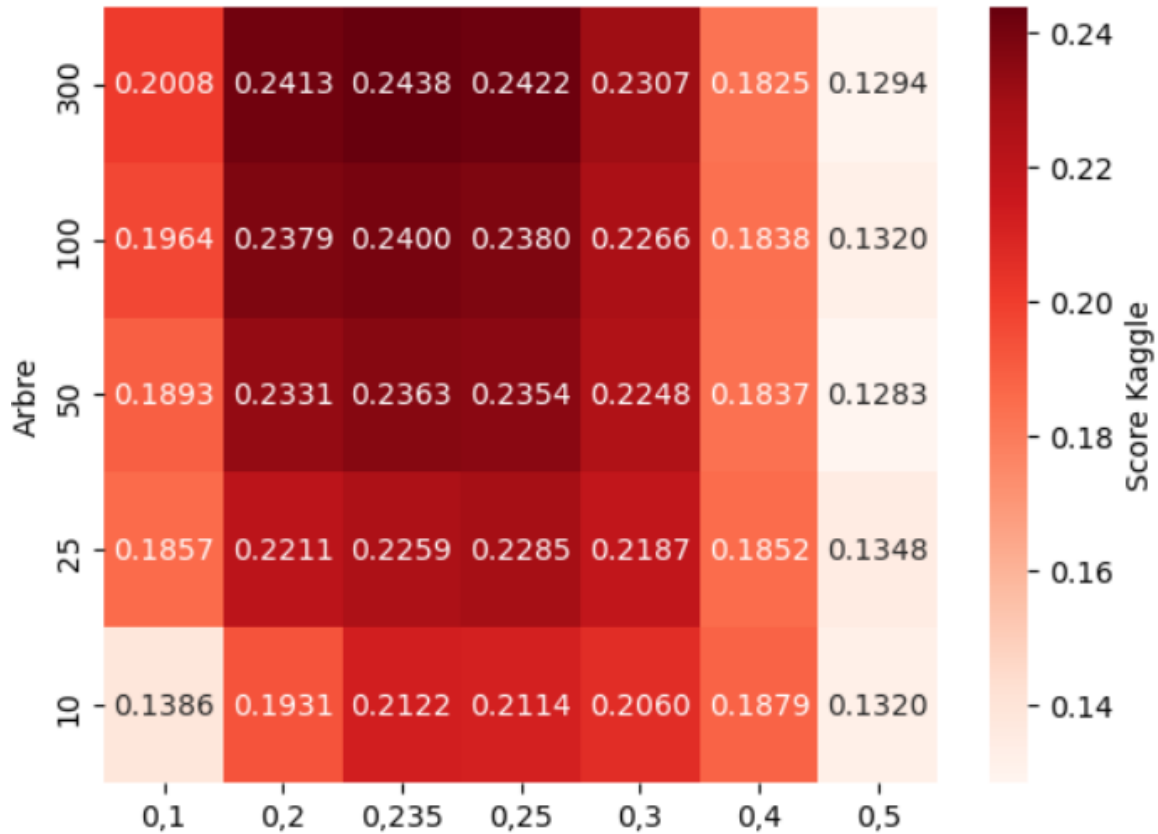


FIGURE 2.2 – Hitmap des résultats

2.1.3 SVM

L'algorithme implémenté par la suite est celui du SVM qui consiste à trouver un hyper-plan H séparant les données. La tâche de classification qui nous incombe étant plutôt complexe, c'est l'algorithme du SVM à marge souple qui nous intéresse le plus car nos données ont peu de chances d'être séparables complètement. L'hyperplan H possède donc une marge de norme w qui correspond à la zone d'incertitude.

Le modèle repose sur 28 variables descriptives quantitatives, afin de simplifier l'analyse on effectue une ACP sur les variables. 99% de la variance étant contenue dans les 4 premières dimensions de l'espace réduit, on considérera parfois 3 dimensions (98% de la variance), parfois 4.

Équations mathématiques :

L'équation de l'hyperplan séparateur est la suivante : $H = \{x \in R^d, \langle \hat{w}, x \rangle + \hat{b} = 0\}$, cela revient à maximiser la marge. Les hyperplans H^- et H^+ ont des équations similaires avec $\langle \hat{w}, x \rangle + \hat{b} = -1, 1$ respectivement. Dans le cadre d'un SVM à marge souple on cherchera les paramètres \hat{w} et \hat{b} (la variable d'ajustement mesurant à quel point la contrainte de marge a été enfreinte par un point) de telle sorte à avoir $\arg \min_w \left(\frac{1}{2} \|\hat{w}\|^2 + C \sum_i e_i \right)$ en

respectant les contraintes $y_i(\langle w, x_i \rangle + b) \geq 1 - e_i$ et $e_i \geq 0$. C est un hyperparamètre utilisé pour pénaliser les erreurs.

Ces conditions ne suffisant pas à séparer linéairement les données, des "astuces" seront employées pour projeter les données dans un autre espace à l'aide d'une fonction $\phi : X \rightarrow H$ (l'espace de redescription) et en utilisant l'astuce du noyau consistant à remplacer le produit scalaire des fonctions phi par une fonction K (noyaux polynomiaux de degré 2, 3 ou 4, noyaux gaussiens).

Conditions d'application :

L'application d'un tel modèle suppose une séparabilité linéaire entre les variables justifiant la réduction de dimensions par ACP ou l'utilisation de fonctions K pour projeter les données dans un autre espace. Cela implique aussi un effet des variables en question sur la présence ou l'absence des espèces. Les coordonnées géographiques qui jouent un rôle important dans la répartition des espèces ne sont pas considérés ici, c'est un modèle qui repose purement sur les variables abiotiques.

Choix des hyperparamètres :

Le premier hyperparamètres est C : le coefficient associé aux erreurs pour les pénaliser. On jouera également sur la fonction K utilisée pour définir l'espace de redescription.

Score :










| | | | |
|--|----------------|----------------|--------------------------|
|  modeles_svm_d4_p02_p4.csv Complete (after deadline) · 2mo ago | 0.11068 | 0.11396 | <input type="checkbox"/> |
|  modeles_svm_d4_p012_p2.csv Complete (after deadline) · 2mo ago | 0.16375 | 0.1667 | <input type="checkbox"/> |
|  modeles_svm_d4_p005_p2.csv Complete (after deadline) · 2mo ago | 0.14295 | 0.14561 | <input type="checkbox"/> |
|  modeles_svm_d4_p02_p2.csv Complete (after deadline) · 2mo ago | 0.10984 | 0.11323 | <input type="checkbox"/> |
|  modeles_svm_d4_p01_p2.csv Complete (after deadline) · 2mo ago | 0.16647 | 0.17059 | <input type="checkbox"/> |
|  modeles_svm_d4_p02_p3.csv Complete (after deadline) · 2mo ago | 0.13622 | 0.13881 | <input type="checkbox"/> |
|  modeles_svm_d4_p01_p3.csv Complete (after deadline) · 2mo ago | 0.17112 | 0.17581 | <input type="checkbox"/> |
|  modeles_svm_d3_p01_p3.csv Complete (after deadline) · 2mo ago | 0.17465 | 0.17909 | <input type="checkbox"/> |
|  modeles_svm_d3_p02_p3.csv Complete (after deadline) · 2mo ago | 0.12772 | 0.13112 | <input type="checkbox"/> |

FIGURE 2.3 – Scores Kaggle SVM

Visualisation :

Un exemple de présences / absences pour l'espèce n°6574

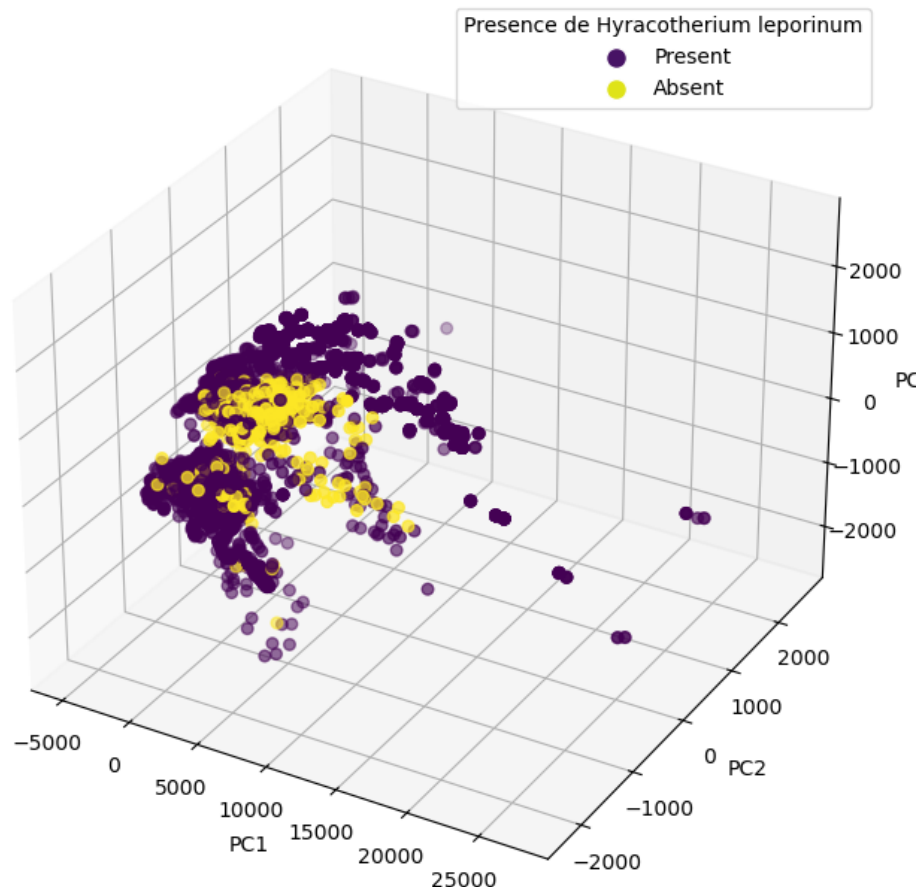


FIGURE 2.4 – Présence de l'espèce 6574 (*Hyracotherium leporinum*)

2.2 A mettre en place

2.2.1 Multilayer perceptron

Afin de s'initier à PyTorch, l'objectif est de construire un réseau de neurones à plusieurs couches intégrant en entrée les données de présence/absence. Les modèles se complexifieront avec l'ajout de nouvelles données. Par exemple dans une démarche de boosting, nous envisageons d'ajouter 2174 nouvelles variables d'entrées correspondantes à la probabilité de prédiction de chaque espèce par le modèle précédent.

La fonction de perte considérée sera celle de la "Binary cross entropy" utilisée par exemple dans des modèles de régression logistique et pour de la classification 0/1 (on peut se rapporter à un problème de classification binaire en prenant espèce par espèce)

2.2.2 Poisson

Le framework considéré en finalité de ce projet reposera sur les données de présences uniquement et la loi de poisson (comptage des espèces) en définissant la probabilité d'obte-

nir un nombre compris entre $[0 ; +\text{Inf}]$. Il sera possible de se ramener à une problématique de présence / absence grâce aux processus ponctuels de poisson (division du nombre prédit par l'intégrale sous la courbe). Les données de présence/absence permettront de stabiliser le modèle par leur fiabilité.

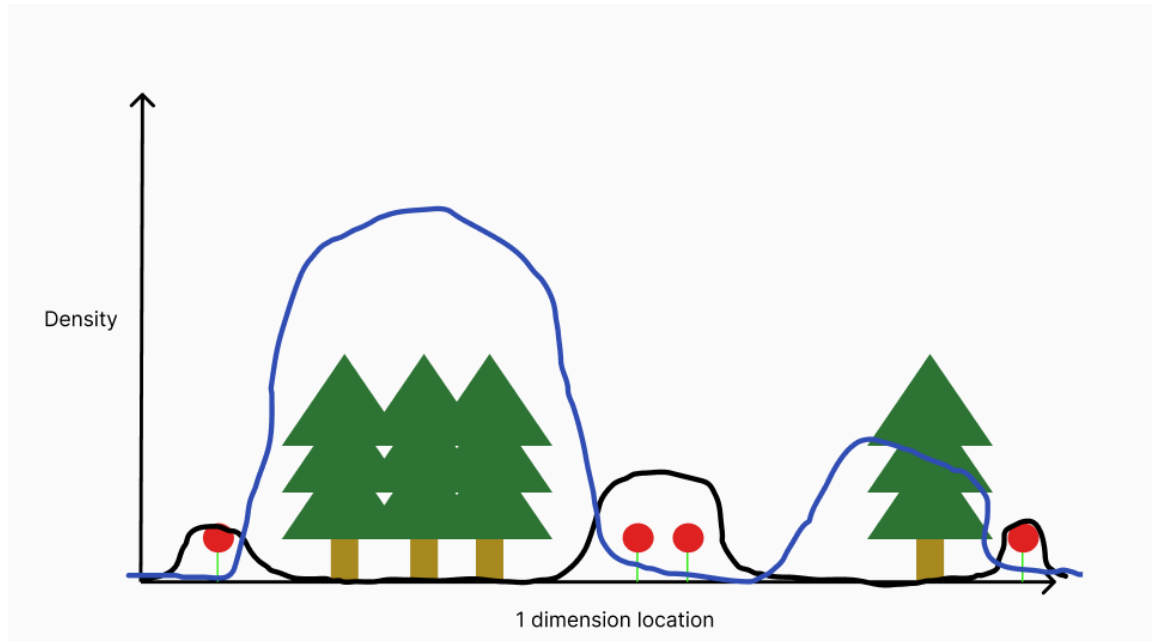


FIGURE 2.5

Conclusion

Ce semestre, nous visons l'application de modèles d'apprentissage profond pour améliorer nos résultats, tout en tirant pleinement parti de l'ensemble des données à notre disposition malgré leur complexité de mise en place. Parallèlement, nous prévoyons de développer un outil de visualisation dédié à nos résultats.

Bibliographie

1. *Tutoriel PyTorch, Tensors* : https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html
2. *Tutoriel PyTorch, AutoGrade* : https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html
3. *Tutoriel PyTorch, Neural Networks* : https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
4. *Tutoriel PyTorch, Training a Classifier* : https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
5. *Documentation Random Forest* : <https://blent.ai/blog/a/random-forest-comment-ca-mar>