Université de Lille
Année Universitaire 2022/2023

Faculté de Pharmacie de Lille

THESE
POUR LE DIPLOME D'ETAT
DE DOCTEUR EN PHARMACIE

Soutenue publiquement le 12 septembre 2023
Par M. RIHANI Emir Kaïs

APPLICATION DE MODELES D'INTELLIGENCE ARTIFICIELLE

A LA CLASSIFICATION DES MACROMYCETES

Membres du jury :

Président : Nom, Prenom, titre et lieu de fonction

Directeur, conseiller de thèse : Nom, Prenom, titre et lieu de fonction

Assesseur(s) : Nom, Prenom, titre et lieu de fonction

Faculté de Pharmacie de Lille
3 Rue du Professeur Laguesse – 59000 Lille
03 20 96 40 40
https://pharmacie.univ-lille.fr

Université de Lille

Président	Régis BORDET
Premier Vice-président	Etienne PEYRAT
Vice-présidente Formation	Christel BEAUCOURT
Vice-président Recherche	Olivier COLOT
Vice-présidente Réseaux internationaux et européens	Kathleen O'CONNOR
Vice-président Ressources humaines	Jérôme FONCEL
Directrice Générale des Services	Marie-Dominique SAVINA

UFR3S

Doyen	Dominique LACROIX
Premier Vice-Doyen	Guillaume PENEL
Vice-Doyen Recherche	Éric BOULANGER
Vice-Doyen Finances et Patrimoine	Damien CUNY
Vice-Doyen Coordination pluriprofessionnelle et Formations sanitaires	Sébastien D'HARANCY
Vice-Doyen RH, SI et Qualité	Hervé HUBERT
Vice-Doyenne Formation tout au long de la vie	Caroline LANIER
Vice-Doyen Territoires-Partenariats	Thomas MORGENROTH
Vice-Doyenne Vie de Campus	Claire PINÇON
Vice-Doyen International et Communication	Vincent SOBANSKI
Vice-Doyen étudiant	Dorian QUINZAIN

Faculté de Pharmacie

Doyen	Delphine ALLORGE
Premier Assesseur et Assesseur en charge des études	Benjamin BERTIN
Assesseur aux Ressources et Personnels	Stéphanie DELBAERE
Assesseur à la Santé et à l'Accompagnement	Anne GARAT
Assesseur à la Vie de la Faculté	Emmanuelle LIPKA
Responsable des Services	Cyrille PORTA
Représentant étudiant	Honoré GUISE

Professeurs des Universités - Praticiens Hospitaliers (PU-PH)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	ALLORGE	Delphine	Toxicologie et Santé publique	81
M.	BROUSSEAU	Thierry	Biochimie	82
M.	DÉCAUDIN	Bertrand	Biopharmacie, Pharmacie galénique et hospitalière	81
M.	DINE	Thierry	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
Mme	DUPONT-PRADO	Annabelle	Hématologie	82
Mme	GOFFARD	Anne	Bactériologie - Virologie	82
M.	GRESSIER	Bernard	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
M.	ODOU	Pascal	Biopharmacie, Pharmacie galénique et hospitalière	80
Mme	POULAIN	Stéphanie	Hématologie	82
M.	SIMON	Nicolas	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
M.	STAELS	Bart	Biologie cellulaire	82

Professeurs des Universités (PU)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	ALIOUAT	El Moukhtar	Parasitologie - Biologie animale	87
Mme	AZAROUAL	Nathalie	Biophysique - RMN	85
M.	BLANCHEMAIN	Nicolas	Pharmacotechnie industrielle	85
M.	CARNOY	Christophe	Immunologie	87
M.	CAZIN	Jean-Louis	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
M.	CHAVATTE	Philippe	Institut de Chimie Pharmaceutique Albert Lespagnol	86
M.	COURTECUISSE	Régis	Sciences végétales et fongiques	87
M.	CUNY	Damien	Sciences végétales et fongiques	87
Mme	DELBAERE	Stéphanie	Biophysique - RMN	85
Mme	DEPREZ	Rebecca	Chimie thérapeutique	86
M.	DEPREZ	Benoît	Chimie bioinorganique	85
M.	DUPONT	Frédéric	Sciences végétales et fongiques	87

M.	DURIEZ	Patrick	Physiologie	86
M.	ELATI	Mohamed	Biomathématiques	27
M.	FOLIGNÉ	Benoît	Bactériologie - Virologie	87
Mme	FOULON	Catherine	Chimie analytique	85
M.	GARÇON	Guillaume	Toxicologie et Santé publique	86
M.	GOOSSENS	Jean-François	Chimie analytique	85
M.	HENNEBELLE	Thierry	Pharmacognosie	86
M.	LEBEGUE	Nicolas	Chimie thérapeutique	86
M.	LEMDANI	Mohamed	Biomathématiques	26
Mme	LESTAVEL	Sophie	Biologie cellulaire	87
Mme	LESTRELIN	Réjane	Biologie cellulaire	87
Mme	MELNYK	Patricia	Chimie physique	85
M.	MILLET	Régis	Institut de Chimie Pharmaceutique Albert Lespagnol	86
Mme	MUHR-TAILLEUX	Anne	Biochimie	87
Mme	PERROY	Anne-Catherine	Droit et Economie pharmaceutique	86
Mme	ROMOND	Marie-Bénédicte	Bactériologie - Virologie	87
Mme	SAHPAZ	Sevser	Pharmacognosie	86
M.	SERGHERAERT	Éric	Droit et Economie pharmaceutique	86
M.	SIEPMANN	Juergen	Pharmacotechnie industrielle	85
Mme	SIEPMANN	Florence	Pharmacotechnie industrielle	85
M.	WILLAND	Nicolas	Chimie organique	86

Maîtres de Conférences - Praticiens Hospitaliers (MCU-PH)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	BLONDIAUX	Nicolas	Bactériologie - Virologie	82
Mme	DEMARET	Julie	Immunologie	82
Mme	GARAT	Anne	Toxicologie et Santé publique	81
Mme	GENAY	Stéphanie	Biopharmacie, Pharmacie galénique et hospitalière	81
M.	LANNOY	Damien	Biopharmacie, Pharmacie galénique et hospitalière	80

Mme	ODOU	Marie-Françoise	Bactériologie - Virologie	82

Maîtres de Conférences des Universités (MCU)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	AGOURIDAS	Laurence	Chimie thérapeutique	85
Mme	ALIOUAT	Cécile-Marie	Parasitologie - Biologie animale	87
M.	ANTHÉRIEU	Sébastien	Toxicologie et Santé publique	86
Mme	AUMERCIER	Pierrette	Biochimie	87
M.	BANTUBUNGI-BLUM	Kadiombo	Biologie cellulaire	87
Mme	BARTHELEMY	Christine	Biopharmacie, Pharmacie galénique et hospitalière	85
Mme	BEHRA	Josette	Bactériologie - Virologie	87
M.	BELARBI	Karim-Ali	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
M.	BERTHET	Jérôme	Biophysique - RMN	85
M.	BERTIN	Benjamin	Immunologie	87
M.	BOCHU	Christophe	Biophysique - RMN	85
M.	BORDAGE	Simon	Pharmacognosie	86
M.	BOSC	Damien	Chimie thérapeutique	86
M.	BRIAND	Olivier	Biochimie	87
Mme	CARON-HOUDE	Sandrine	Biologie cellulaire	87
Mme	CARRIÉ	Hélène	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
Mme	CHABÉ	Magali	Parasitologie - Biologie animale	87
Mme	CHARTON	Julie	Chimie organique	86
M.	CHEVALIER	Dany	Toxicologie et Santé publique	86
Mme	DANEL	Cécile	Chimie analytique	85
Mme	DEMANCHE	Christine	Parasitologie - Biologie animale	87
Mme	DEMARQUILLY	Catherine	Biomathématiques	85
M.	DHIFLI	Wajdi	Biomathématiques	27
Mme	DUMONT	Julie	Biologie cellulaire	87
M.	EL BAKALI	Jamal	Chimie thérapeutique	86
M.	FARCE	Amaury	Institut de Chimie Pharmaceutique Albert Lespagnol	86

M.	FLIPO	Marion	Chimie organique	86
M.	FURMAN	Christophe	Institut de Chimie Pharmaceutique Albert Lespagnol	86
M.	GERVOIS	Philippe	Biochimie	87
Mme	GOOSSENS	Laurence	Institut de Chimie Pharmaceutique Albert Lespagnol	86
Mme	GRAVE	Béatrice	Toxicologie et Santé publique	86
Mme	GROSS	Barbara	Biochimie	87
M.	HAMONIER	Julien	Biomathématiques	26
Mme	HAMOUDI-BEN YELLES	Chérifa-Mounira	Pharmacotechnie industrielle	85
Mme	HANNOTHIAUX	Marie-Hélène	Toxicologie et Santé publique	86
Mme	HELLEBOID	Audrey	Physiologie	86
M.	HERMANN	Emmanuel	Immunologie	87
M.	KAMBIA KPAKPAGA	Nicolas	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86
M.	KARROUT	Younes	Pharmacotechnie industrielle	85
Mme	LALLOYER	Fanny	Biochimie	87
Mme	LECOEUR	Marie	Chimie analytique	85
Mme	LEHMANN	Hélène	Droit et Economie pharmaceutique	86
Mme	LELEU	Natascha	Institut de Chimie Pharmaceutique Albert Lespagnol	86
Mme	LIPKA	Emmanuelle	Chimie analytique	85
Mme	LOINGEVILLE	Florence	Biomathématiques	26
Mme	MARTIN	Françoise	Physiologie	86
M.	MOREAU	Pierre-Arthur	Sciences végétales et fongiques	87
M.	MORGENROTH	Thomas	Droit et Economie pharmaceutique	86
Mme	MUSCHERT	Susanne	Pharmacotechnie industrielle	85
Mme	NIKASINOVIC	Lydia	Toxicologie et Santé publique	86
Mme	PINÇON	Claire	Biomathématiques	85
M.	PIVA	Frank	Biochimie	85
Mme	PLATEL	Anne	Toxicologie et Santé publique	86
M.	POURCET	Benoît	Biochimie	87
M.	RAVAUX	Pierre	Biomathématiques / Innovations pédagogiques	85

Mme	RAVEZ	Séverine	Chimie thérapeutique	86
Mme	RIVIÈRE	Céline	Pharmacognosie	86
M.	ROUMY	Vincent	Pharmacognosie	86
Mme	SEBTI	Yasmine	Biochimie	87
Mme	SINGER	Elisabeth	Bactériologie - Virologie	87
Mme	STANDAERT	Annie	Parasitologie - Biologie animale	87
M.	TAGZIRT	Madjid	Hématologie	87
M.	VILLEMAGNE	Baptiste	Chimie organique	86
M.	WELTI	Stéphane	Sciences végétales et fongiques	87
M.	YOUS	Saïd	Chimie thérapeutique	86
M.	ZITOUNI	Djamel	Biomathématiques	85

Professeurs certifiés

Civ.	Nom	Prénom	Service d'enseignement
Mme	FAUQUANT	Soline	Anglais
M.	HUGES	Dominique	Anglais
M.	OSTYN	Gaël	Anglais

Professeurs Associés

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
M.	DAO PHAN	Haï Pascal	Chimie thérapeutique	86
M.	DHANANI	Alban	Droit et Economie pharmaceutique	86

Maîtres de Conférences Associés

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	CUCCHI	Malgorzata	Biomathématiques	85
M.	DUFOSSEZ	François	Biomathématiques	85
M.	FRIMAT	Bruno	Pharmacologie, Pharmacocinétique et Pharmacie clinique	85
M.	GILLOT	François	Droit et Economie pharmaceutique	86
M.	MASCAUT	Daniel	Pharmacologie, Pharmacocinétique et Pharmacie clinique	86

M.	MITOUMBA	Fabrice	Biopharmacie, Pharmacie galénique et hospitalière	86
M.	PELLETIER	Franck	Droit et Economie pharmaceutique	86
M.	ZANETTI	Sébastien	Biomathématiques	85

Assistants Hospitalo-Universitaire (AHU)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	CUVELIER	Élodie	Pharmacologie, Pharmacocinétique et Pharmacie clinique	81
M.	GRZYCH	Guillaume	Biochimie	82
Mme	LENSKI	Marie	Toxicologie et Santé publique	81
Mme	HENRY	Héloïse	Biopharmacie, Pharmacie galénique et hospitalière	80
Mme	MASSE	Morgane	Biopharmacie, Pharmacie galénique et hospitalière	81

Attachés Temporaires d'Enseignement et de Recherche (ATER)

Civ.	Nom	Prénom	Service d'enseignement	Section CNU
Mme	GEORGE	Fanny	Bactériologie - Virologie / Immunologie	87
Mme	N'GUESSAN	Cécilia	Parasitologie - Biologie animale	87
M.	RUEZ	Richard	Hématologie	87
M.	SAIED	Tarak	Biophysique - RMN	85
M.	SIEROCKI	Pierre	Chimie bioinorganique	85

Enseignant contractuel

Civ.	Nom	Prénom	Service d'enseignement
M.	MARTIN MENA	Anthony	Biopharmacie, Pharmacie galénique et hospitalière

Faculté de Pharmacie de Lille

3 Rue du Professeur Laguesse – 59000 Lille
03 20 96 40 40
https://pharmacie.univ-lille.fr

L'Université n'entend donner aucune approbation aux opinions émises dans les thèses ; celles-ci sont propres à leurs auteurs.

J'adresse mes sincères remerciements à :

La Dream Team de tous ceux que je vais remercier

Table des matières

1 Liste des abréviations

AI : *Artificial Intelligence* (intelligence artificielle)
AUC : *Area Under Curve* (aire sous la courbe)
CART : *Classification And Regression Tree* (arbre de classification et de régression)
CSV : *Comma Separated Values* ([fichier de] valeurs séparées par des virgules)
DOE : *Design of Experiments* (plan d'expériences)
EDA : *Exploratory Data Analysis* (analyse exploratoire des données)
ESE : *Enhanced Stochastic Evolutionnary* ([algorithme] évolutionnaire stochastique amélioré)
GAM : *Generalized Additive Model* (modèle additif généralisé)
IA : Intelligence Artificielle
LDA : *Linear Discriminant Analysis* (analyse linéaire discriminante)
LHS : *Latin Hypercube Sample* (échantillonnage par hypercube latin)
LOESS : *LOcally Estimated Scatterplot Smoothing* (régression locale)
NOLH : *Nearly Orthogonal Latin Hypercube* (hypercube latin quasi orthogonal)
PDA : *Penalized Discriminant Analysis* (analyse discriminante pénalisée)
RF : *Random Forest* (forêt aléatoire)
ROC : *Receiver Operating Characteristic* (fonction d'efficacité du récepteur)
VCS : *Version Control System* (logiciel de contrôle des versions)
XGB : *eXtreme Gradient Boosting*

2 Introduction

2.1 Propos liminaire

L'identification des macromycètes est un sujet difficile, ne devant évidemment pas être pris à la légère. Les espèces rencontrées varient considérablement d'un écosystème à un autre, d'un continent à un autre, et aucun lot de données ni ouvrage sur les champignons ne saurait couvrir toute la diversité du monde fongique.

Le lot de données mycologiques constitué dans cette étude, bien que constituant l'un des lots en libre accès les plus complets du domaine de la *data science*, n'est bien entendu pas exhaustif.

Ce lot se concentre exclusivement sur les champignons habituellement rencontrés au Nord de la France. Nombre de variétés, parfois très connues, ne sont donc pas présentes, parmi lesquelles nous pouvons par exemple citer les représentants du genre *psilocybe*, connus pour leurs propriétés psychédéliques. Certains critères pourront également varier de manière considérable selon le stade de maturité du champignon : alors que les chapeaux vert-olive de l'*Amanita phalloides* mature sont faciles à reconnaître, les spécimens jeunes sont blancs et pourraient facilement être confondus avec des espèces comestibles (par exemple du genre *Agaricus*).

L'ingestion de certains de ces champignons est *mortelle*, même en faible quantité. Le diagnostic de l'intoxication fongique peut être difficile, et parfois trop tardif pour un traitement efficace. Des composés toxiques tels que les amanitines ne sont pas altérés ou détruits par cuisson ou congélation, et seront absorbés par l'intestin, avant de passer dans la circulation sanguine afin d'être filtrés par le foie, détruisant les cellules hépatiques, puis excrétées dans l'intestin, réabsorbées, refiltrées... chaque passe détruisant les cellules hépatiques ayant survécu à la précédente, dans un cycle connu sous le nom de réabsorption hépato-entérique.

Il ne faut jamais, *sous aucune circonstance*, utiliser les lots de données générés par des méthodes similaires à celles de notre études dans le but de déterminer si un champignon est comestible ou non.

2.2 But de l'étude

L'identification des plantes et champignons est un problème de classification classique, qui est usuellement effectué de façon manuelle, à l'aide de clés d'identification. La plupart de ces clés sont basées sur un processus utilisant des arbres décisionnels, ce qui pourrait sembler logique car rappelant la logique en arbre de l'évolution. Quoique séduisant, cet argument rencontre certaines limites :

La première limite est le nombre de chaînons manquants. Certaines espèces sont évidemment éteintes, ce qui signifie que certaines branches et nœuds de l'arbre phylogénétique seront manquants,

ce qui peut compliquer l'analyse quand deux espèces apparentées ont un nombre élevé de chaînons et nœuds communs manquants. Certaines similarités entre espèces peuvent également ne pas être identifiables de façon macroscopique.

La seconde limite, plus profonde, est la logique inhérente au processus évolutionnaire. Deux phénomènes antagonistes sont en jeu : convergence et divergence évolutives. Ces deux phénomènes sont liés à la nécessaire adaptation des espèces à leurs environnements. La divergence évolutive explique par exemple la diversité des mammifères : les chauves-souris, baleines et chevaux sont apparentés, mais ont des aspects très dissemblables en raison de leur adaptation à des environnements très différents. D'un autre côté, la convergence évolutive explique la similarité entre l'aile de la chauve-souris et celle de l'abeille. Toutefois, malgré leur apparente dissimilarité, l'aile de la chauve-souris est plus proche de la main humaine ou de la nageoire de la baleine que de l'aile de l'abeille. La façon la plus fiable pour évaluer le processus évolutionnaire et trouver les liens phylogénétiques de la manière la plus précise possible est l'analyse des génomes : les caractéristiques visibles peuvent être trompeuses. Malheureusement, ces caractéristiques sont souvent les seules aisément identifiables.

Le troisième problème est le critère principal de la classification. Ce critère peut être lié ou non au processus évolutionnaire ou aux critères visible, surtout si ce critère principal est vague. Le critère de comestibilité ou de non-comestibilité retenu pour les lots de données mycologiques usuellement utilisés en *data science* souffre de ce problème : il est essentiellement centré sur la toxicité contre les humains, or de nombreux mécanismes de toxicité peuvent exister, et une toxicité ou non-toxicité d'un métabolite fongique ou végétal peut être liée à des variations métaboliques très ténues entre une espèce et une autre.

Pour ces raisons parmi d'autres, la logique arborescente, bien qu'utilisée habituellement dans l'identification des champignons et des plantes, et souvent justifiée par la nature arborescente du processus évolutionnaire, pourrait ne pas nécessairement être l'approche optimale à la classification des espèces basée sur des critères macroscopiques. Le but de cette étude est notamment de déployer des algorithmes d'apprentissage machine afin d'effectuer cette tâche de classification basée sur des indices visuels limités, et d'évaluer les performances relatives de différentes stratégies de classification.

2.3 État de l'art des lots de données mycologiques

Le tout premier lot de données mycologiques en libre accès mentionné en data science est probablement le *Mushroom Dataset* créé par Jeff Schlimmer en 1987.[1]

Un lot de données plus conséquent a été publié par Dennis Wagner en 2021[2] et mis en libre accès sous le nom de *Secondary Mushroom Dataset*.

3 Création du lot de données

3.1 Configuration matérielle et logicielle

Le code d'apprentissage machine, les méthodes d'évaluation, ainsi que cette thèse ont été rédigés sur l'équipement suivant :

- CPU : AMD Ryzen 5 5600G
- RAM : 2x16 Go DDR4-3200
- SSD : Crucial P5 M2 NVMe
- OS : Xubuntu Linux 22.04.2 LTS
- R : version 4.2.2 (2022)
- IDE : RStudio version 2022.7.2.576, "Spotted Wakerobin"
- VCS : git version 2.34.1
- Librairies : tidyverse[3] (v1.3.2), microbenchmark[4] (v1.4.9), MASS[5] (v7.3.58.2), caret[6] (v6.0.93), ?GGally?[7] (v2.1.2), twinning[8] (v1.0), rpart[9] (v4.1.19), ?C50?[10] (v0.1.8), party[11] (v1.3.11), ranger[12] (v0.14.1), rFerns[13] (v5.0.0), Rborist[14] (v0.3.2), rmarkdown[15] (v2.20), knitr[16] (v1.41), ggpubr[17] (v0.6.0), DiceDesign[18] (v1.9), DiceEval[19] (v1.5.1), bookdown[20] (v0.32).

3.2 Principes de conception d'un lot de données synthétiques

3.2.1 Principes généraux

Un lot de données synthétiques est un lot de données généré par un algorithme, par opposition aux lots de données issus d'une collecte effectuée en "vie réelle".

Trois stratégies sont usuellement utilisées :

- Données factices (*dummy data*) : l'ensemble des données est généré aléatoirement.
- Données générées à partir de règles (*rule-based data*) : l'ensemble des données est généré suivant des lois définies au préalable (distribution, valeurs moyennes, minimales, maximales...)
- Données générées par intelligence artificielle (*AI generated*) : l'ensemble des données est généré suivant des lois extraites par l'IA suite à l'analyse d'un échantillon de données obtenues en "vie réelle".

Les données générées par ces stratégies peuvent être de type variés, que nous pouvons grossièrement regrouper en données alphanumériques (quantitatives et qualitatives), en séries temporelles, et en données d'imagerie.

Pour des raisons pratiques et de maturité des technologies disponibles à l'heure actuelle, la méthode retenue pour créer le lot de données exploité dans notre étude sera la génération de données alphanumériques à partir de règles, extraites d'ouvrages mycologiques de référence.[21–23]

3.2.2 Principes de génération des paramètres quantitatifs

Dans le cadre de cette étude, les variables quantitatives générées aléatoirement sont :

- La longueur du stipe L_S,
- Le diamètre du stipe D_S,
- Le diamètre du chapeau D_C.

En première approximation, nous pouvons considérer que toutes ces valeurs sont intrinsèquement liées à la croissance du champignon. Ces trois variables peuvent, dans l'absolu, être susceptibles de varier indépendamment des autres au cours de la croissance du champignon, les variables L_S, D_S et D_C obéissant alors aux lois suivantes :

$$\begin{cases} L_S = L_{S_{max}}.F_{Ls} \\ D_S = D_{S_{max}}.F_{Ds} \\ D_C = D_{C_{max}}.F_{Dc} \end{cases}$$

Avec :

- $L_{S_{max}}$, $D_{S_{max}}$ et $D_{C_{max}}$ les valeurs maximales de longueur de stipe, diamètre du stipe et diamètre de chapeau de chaque variété de champignon, extraites de la littérature,
- F_{Ls}, F_{Ds}, F_{Dc} des variables générées aléatoirement dans l'intervalle $]0;1]$, et représentatives de la croissance du spécimen.

Toutefois, nos recherches bibliographiques concernant la cinétique de croissance des sporophores n'ont pas permis de distinguer de différences de la cinétique de croissance de chacun de ces trois paramètres. Nous supposerons donc, en première approximation, que la croissance du stipe en longueur et en largeur, ainsi que la croissance du chapeau s'effectuent à des vitesses identiques. Nous obtenons par conséquent :

$$F_{Ls} = F_{Ds} = F_{Dc} = F_T$$

Avec F_T un facteur représentatif de la taille globale de chaque spécimen, généré aléatoirement.

Ainsi, le problème de génération de nos trois variables aléatoires se simplifie en un problème de génération d'une seule variable aléatoire : le facteur de taille de chaque spécimen. Un certain nombre de distributions d'intérêt sont susceptibles d'être utilisées afin de générer des facteurs de taille F_T aléatoires, il convient donc de définir le cahier des charges de la distribution la plus adaptée au sujet de cette étude.

Les critères de sélection retenus afin de choisir la loi la plus appropriée sont :

- Efficience calculatoire,
- Distribution continue,
- Distribution bornée, ou aisément normalisable sur un intervalle $[0;1]$,
- Distribution asymétrique.

Le premier critère n'est, en pratique, pas un facteur limitant, les temps de calcul pour la génération d'un nombre de facteurs de taille F_T suffisant étant typiquement inférieurs à 200 ms (pour 10^6 facteurs générés) avec la plupart des distributions d'intérêt (voir figure 1).

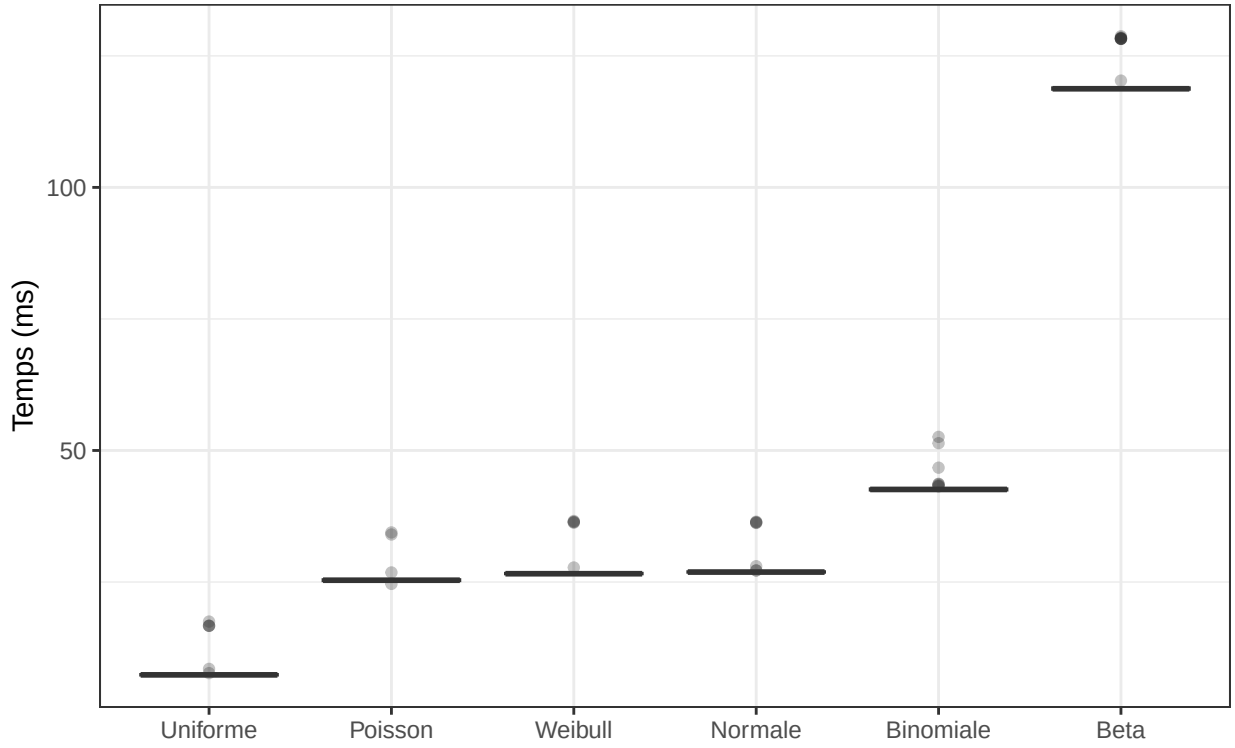

Figure 1: Temps de calcul des principales distributions d'intérêt pour 1e+06 facteurs, (100 iter.)

Les critères de continuité et de normalité n'appellent que peu de commentaires. Ces critères permettent simplement de garantir la possibilité d'une infinité de valeurs dimensionnelles, dans l'intervalle considéré. Le critère de continuité proscrit toutefois l'utilisation de lois de distributions discrètes telles que la loi binomiale ou la loi de Poisson, et celui de normalité écarte des distributions telles que la loi de Weibull, dont la normalisation est parfois délicate.

Le critère d'asymétrie est un critère permettant de tenir compte des différents paramètres pouvant impacter la distribution de taille des spécimens prélevés, parmi lesquels :

- Différences de cinétique de croissance d'une famille à une autre,
- Particularités de la croissance fongique, notamment par la croissance hyphale,[24,25]
- Probabilité de prélèvement variable selon la taille du spécimen (par difficulté de détection, considérations éthiques, intérêt mycologique ou gastronomique. . .).

Le premier paramètre évoqué précédemment n'a pu être exploité dans le cadre de cette étude en raison du manque de données concernant les cinétiques relatives de croissance des sporophores des différentes familles de macromycètes. Le modèle que nous proposons permet toutefois des développements ultérieurs dans ce domaine.

Les deux derniers paramètres permettent de supposer que la distribution de taille des spécimens d'une même espèce à l'issue d'une récolte en vie réelle ne sera pas symétrique, d'une part en raison de la rapidité de la croissance fongique, et d'autre part parce que le prélèvement se fera préférentiellement en épargnant les spécimens de petite taille.

Ainsi, la génération de la variable aléatoire F_T obéira idéalement à une loi de distribution asymétrique vers la droite ($G_1 < 0$). Ce critère d'asymétrie écarte par conséquent les lois de distribution symétriques telles que la loi normale ou la loi uniforme.

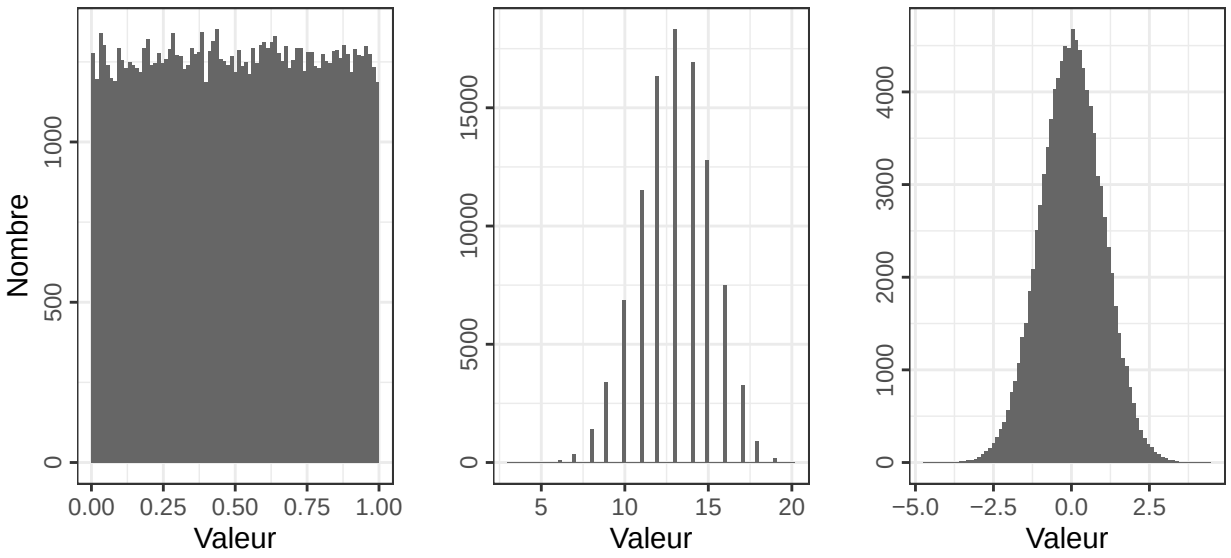

Figure 2: Exemples de distributions de la loi uniforme (à gauche), binomiale (au centre) et normale (à droite)

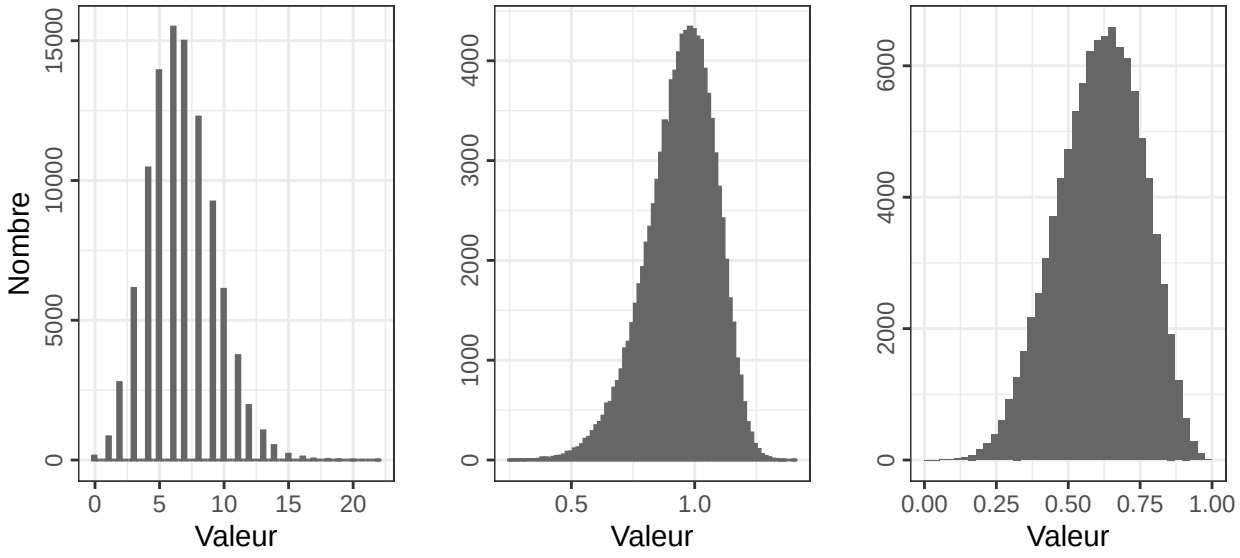

Figure 3: Exemples de distributions de la loi de Poisson (à gauche), de Weibull (au centre) et bêta (à droite)

En raison des contraintes imposées précédemment ainsi que de par sa grande polyvalence,[26] la loi retenue dans le cadre de cette étude pour la génération des facteurs de taille aléatoires (F_T) est une loi bêta non-centrale, définie comme la fonction de distribution de[26,27] :

$$X = \frac{\chi^2_{2\alpha}(\lambda)}{\chi^2_{2\alpha}(\lambda) + \chi^2_{2\beta}}$$

Avec, comme paramètres définis empiriquement pour cette étude :

$$\begin{cases} \alpha = 6\,F_c & (shape1) \\ \beta = 4 & (shape2) \\ \lambda = F_c/2 & (ncp) \end{cases}$$

F_c est ici défini comme un facteur de croissance permettant de rendre compte de la cinétique de croissance de chaque variété d'une part, et du prélèvement préférentiel des spécimens de plus grande taille d'autre part, comme l'illustre la figure 4.

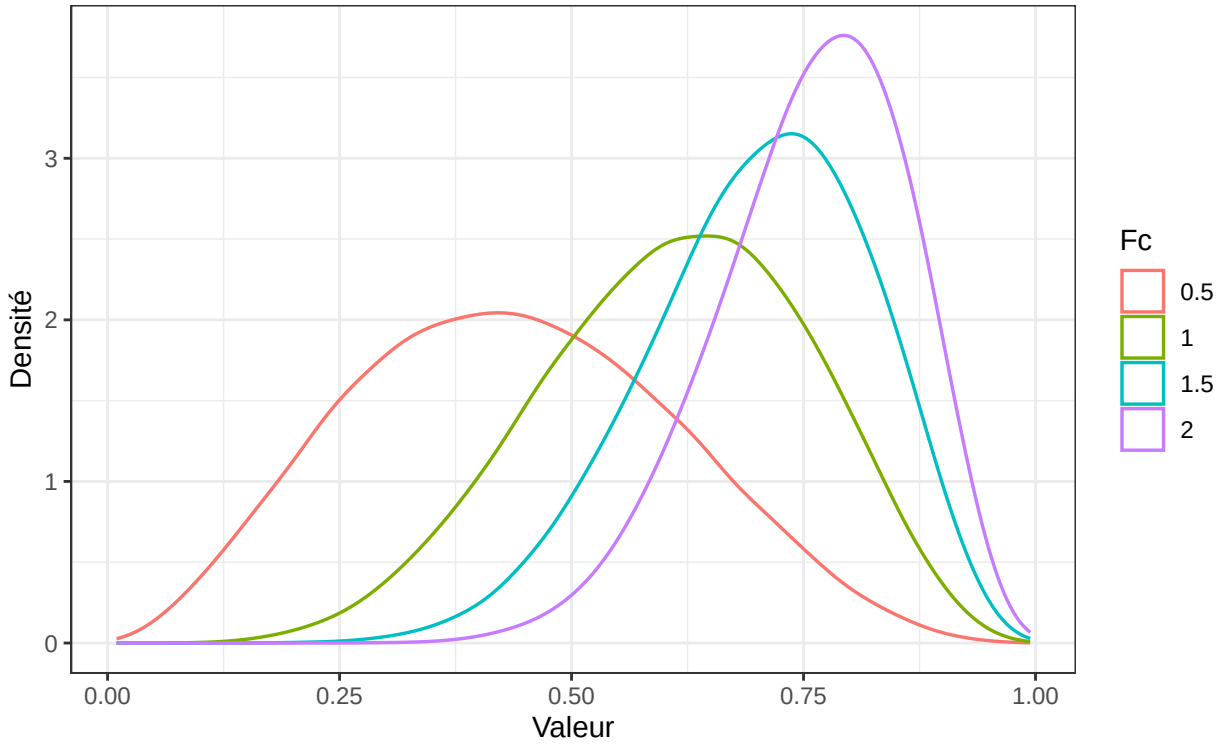

Figure 4: Distribution de différentes lois bêta, en fonction du facteur de croissance Fc

Le modèle défini à ce stade impose une stricte proportionnalité entre diamètre du chapeau D_c, diamètre du stipe D_s et longueur du stipe L_s.

Dans un souci de réalisme, il apparaît souhaitable d'améliorer ce modèle mathématique en y ajoutant un facteur de dispersion, afin de proposer le modèle suivant :

$$\begin{cases} L_S = L_{Smax}.F_T.\delta_{Ls} & avec \quad \delta_{Ls} \sim \mathcal{N}(\mu = 1; \sigma = 0.05) \\ D_S = D_{Smax}.F_T.\delta_{Ds} & avec \quad \delta_{Ds} \sim \mathcal{N}(\mu = 1; \sigma = 0.05) \\ D_C = D_{Cmax}.F_T.\delta_{Dc} & avec \quad \delta_{Dc} \sim \mathcal{N}(\mu = 1; \sigma = 0.05) \end{cases}$$

L'impact de cette dispersion sur la distribution des paramètres de taille L_S, D_S et D_C est illustré par les figures 5 et 6.

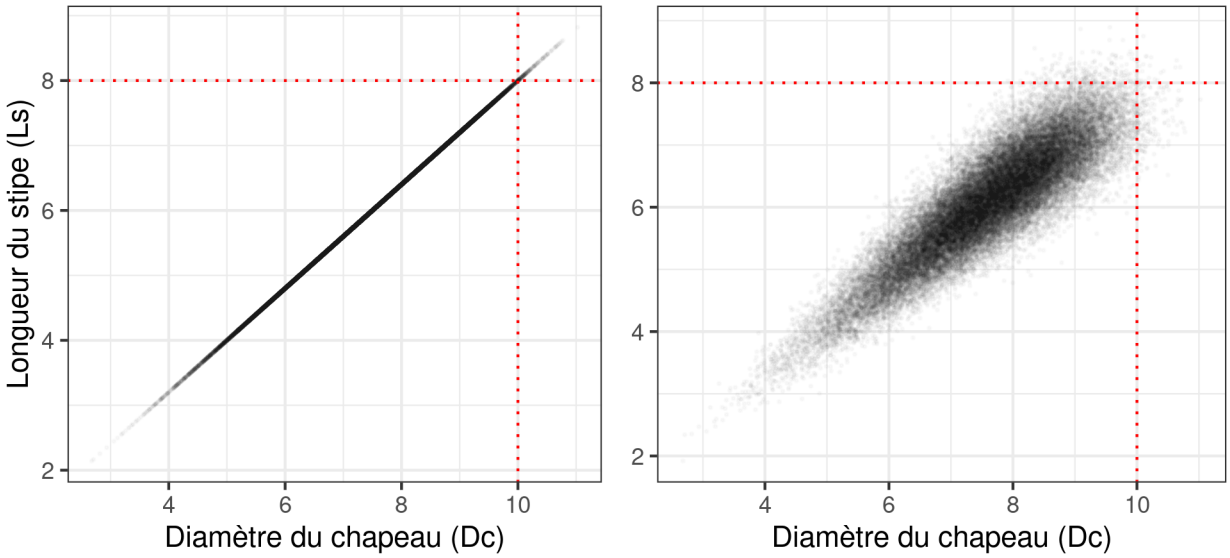

Figure 5: Nuages de points de 2 paramètres de taille (Ls et Dc), sans dispersion (à gauche) et avec dispersion (à droite), pour 50000 champignons

La dispersion ainsi créée permet ainsi de créer de légères variations des rapports entre les différents paramètres de taille, tout en se situant à proximité de la première bissectrice et majoritairement dans la zone 50-90% de la taille maximale (voir figure 6). Cette dispersion autorise par ailleurs l'existence d'une faible proportion de spécimens dépassant les valeurs dimensionnelles maximales généralement admises par la littérature.

Une simulation de Monte Carlo unidimensionnelle effectuée sur 10^5 spécimens nous permet ainsi d'évaluer la proportion de spécimens "hors normes" dépassant la valeur dimensionnelle maximale à environ 0.4 % (cf. figure 7). La même simulation nous permet d'évaluer que la proportion de spécimens "exceptionnels", dépassant de plus de 10% cette valeur maximale, sera quant à elle inférieure à 0.01 %.

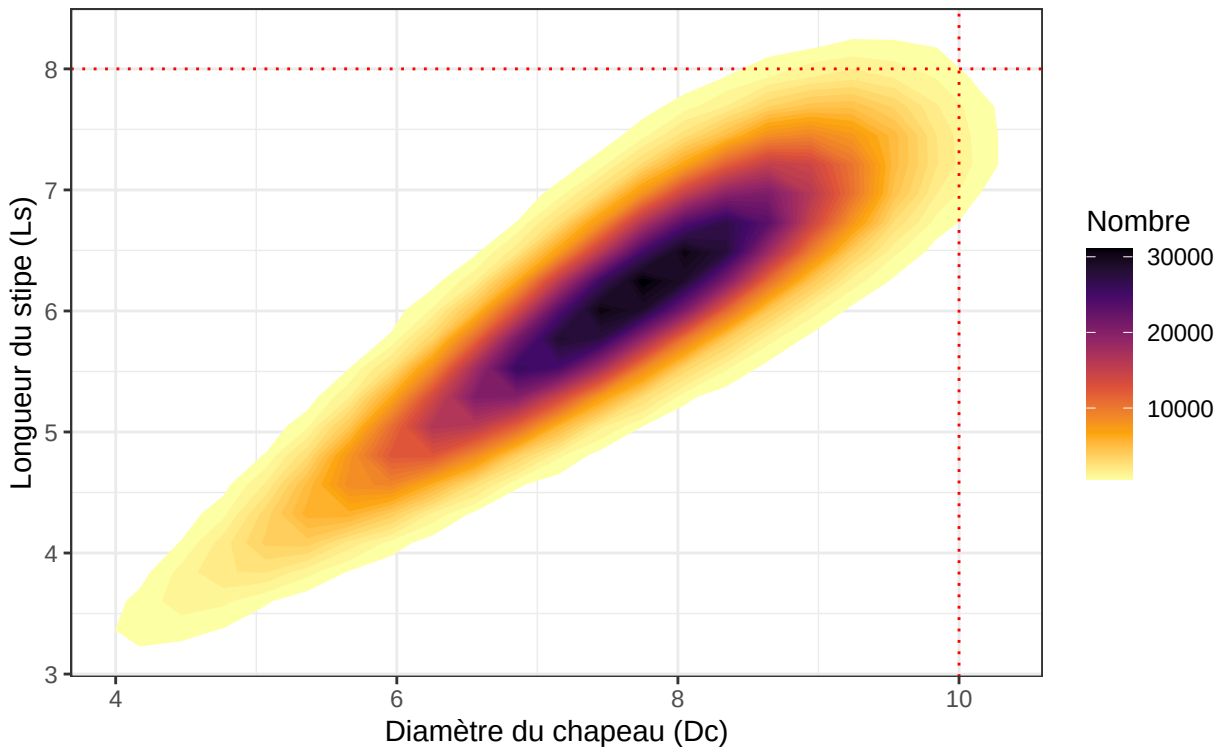

Figure 6: Diagramme de densité de 2 paramètres de taille, avec dispersion

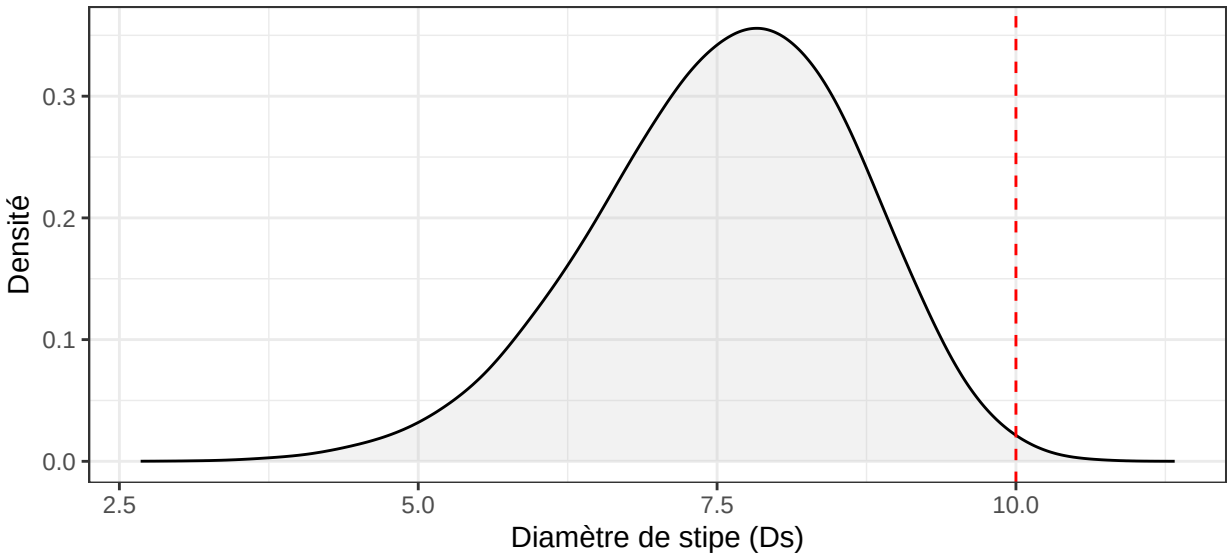

Figure 7: Distribution du diamètre de stipe Ds, pour Dsmax = 10

3.2.3 Principes de génération des paramètres qualitatifs

La génération des paramètres qualitatifs, tels que la couleur des spores ou le type d'hyménophore, est nettement moins complexe que celle des paramètres quantitatifs.

L'ensemble des valeurs quantitatives possibles pour un critère et pour une variété donnée est insérée dans un vecteur de valeurs, et une valeur sera tirée aléatoirement parmi celles contenues dans ce vecteur afin de caractériser le paramètre en question pour chaque spécimen.

4 Principes de l'apprentissage machine

4.1 Jeux de données

Les jeux de données dédiés à l'apprentissage machine sont tous construits sur la base de couples données-résultats. Selon l'étape de l'apprentissage machine, le résultat peut être fourni à la machine ou lui être caché, le but étant dans le premier cas de permettre à la machine d'effectuer son apprentissage, et dans le second cas d'évaluer les performances de la prédiction par rapport au résultat réel.

Le déroulement de l'apprentissage machine se décompose conceptuellement en trois étapes principales, mettant en jeu trois lots de données distincts :

1. Entraînement : le modèle d'apprentissage est exposé à un *jeu de données d'entraînement* (*training data set*), censé être représentatif (cf. section 3.2.1) des données auquel le modèle sera exposé en utilisation réelle. Cette phase est la phase d'apprentissage du modèle.
2. Validation : le modèle d'apprentissage machine élaboré à l'étape précédente, est ici soumis à un *jeu de données de validation* (*validation data set*) et tentera d'apporter des prédictions quant à une variable d'intérêt considérée comme le résultat (ex: comestibilité, espèce...), sur la base des informations contenues dans le lot de données de validation (ex : dimensions, couleurs, morphologie du champignon...). Ces prédictions sont comparées avec les valeurs réelles (ex : comestibilité, espèce...), ce qui permet d'évaluer les performances prédictives du modèle proposé en fonction des indicateurs retenus (spécificité, sensibilité, F1-score, temps de calcul...). Les étapes d'apprentissage et de validation sont répétées de manière itérative en explorant l'ensemble des paramètres de configuration du modèle (hyperparamètres) à fins d'optimisation.
3. Test : les performances du meilleur modèle (avec hyperparamètres optimaux), sélectionné à l'issue de l'étape de validation, sont évaluées vis-à-vis d'un *jeu de données test* (*test* ou *holdout data set*).

La séparation entre étapes d'optimisation et de test peut sembler artificielle. Le problème est en partie lié à un flou sémantique : si l'étape initiale d'entraînement ou d'apprentissage ne pose que peu de problèmes conceptuels, l'étape intermédiaire, dite de *validation* correspond en réalité à une étape d'*optimisation* du modèle et de ses hyperparamètres. Par ailleurs, l'étape finale de *test* sera parfois qualifiée d'étape de *validation* dans la littérature, ce qui peut entretenir la confusion entre ces étapes.[28]

Une distinction sémantique plus nette entre phases d'*apprentissage*, d'*optimisation* et de *test* permet de comprendre plus aisément le fondement épistémologique de cette dernière phase pouvant parfois sembler superflue : l'optimisation effectuée lors de l'étape de validation aboutit à un modèle potentiellement biaisé (problème dit d'*overfitting*) vis-à-vis du jeu de données utilisé comme référence lors de cette étape. Seule une exposition du modèle à des données n'ayant jamais servi à

son entraînement ou son optimisation permettra réellement d'évaluer avec précision son caractère prédictif, donc sa validité.

Dans un souci de clarté, nous utiliserons les termes de lots et de phases d'entraînement, d'optimisation et d'évaluation dans la suite de cette étude.

Les phases d'entraînement, d'optimisation et d'évaluation utilisent chacune un lot de données spécifique. Chacun de ces lots de données est habituellement obtenu suite à dichotomies successives (voir figure 8) du lot de données initial, avec des proportions pouvant être variables d'une scission à l'autre :

1. Découpage du jeu de données initial, en un jeu d'évaluation d'une part, et un jeu d'entraînement + optimisation d'autre part,
2. Découpage du jeu de données entraînement + optimisation, en un jeu d'entraînement et un jeu d'optimisation.

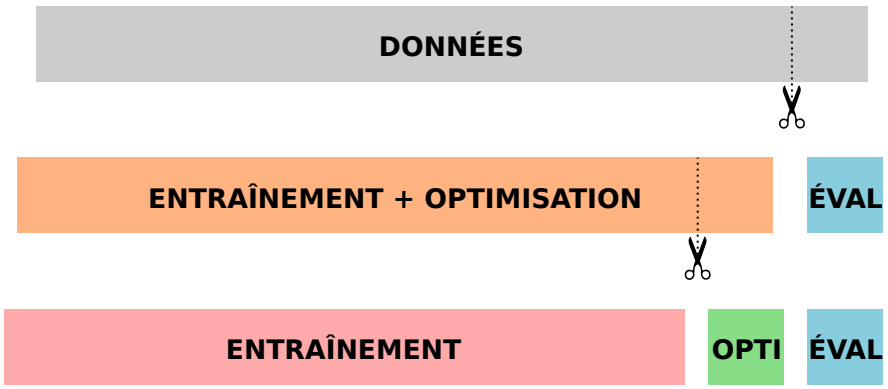

Figure 8: Principe de séparations successives d'un jeu de données initial en jeux d'apprentissage, d'optimisation et d'évaluation.

Les rapports de taille entre jeux de données d'entraînement, d'optimisation et d'évaluation de cette étude suivront la loi $p : \sqrt{p} : \sqrt{p} + 1$, avec p le nombre de coefficients du modèle.[29]

Ce nombre p de coefficients peut être approché par l'expression $p \approx \sqrt{N_u}$, avec N_u le nombre de lignes uniques de notre jeu de données, c'est-à-dire, dans notre cas, le nombre de champignons contenus dans le lot de données.[29]

Un rapide calcul nous montre qu'il est possible d'obtenir ce rapport $p : \sqrt{p} : \sqrt{p} + 1$ par une première scission entre jeu d'entraînement et optimisation d'une part (de taille relative $p + \sqrt{p}$) et jeu d'évaluation d'autre part (de taille relative $\sqrt{p}+1$), avec, pour ce dernier, une taille représentant la fraction du lot total :

$$f_{test} = \frac{\sqrt{p} + 1}{p + 2\sqrt{p} + 1} = \frac{1}{\sqrt{p} + 1}$$

Cette première dichotomie peut être suivie par une seconde dichotomie entre jeu d'entraînement (de taille relative p) et jeu d'optimisation (de taille relative $\sqrt{p}$), de fraction :

$$f_{opti} = \frac{\sqrt{p}}{p + \sqrt{p}} = \frac{1}{\sqrt{p} + 1}$$

En pratique, pour notre lot de données contenant $N_u = 61069$ spécimens, nous pouvons calculer $p \approx 247.1$, soit deux dichotomies successives de ratio 16:1.

4.2 Méthodes de construction des jeux de données

Les méthodes de division mises en œuvre dans cette étude appellent quelques précisions, car elles apportent certaines améliorations par rapport à l'utilisation de deux scissions successives effectuées de manière purement aléatoire.

La première division, entre jeu d'entraînement/optimisation et jeu d'évaluation, mettra en œuvre une méthode de découpage basée sur les points-supports[30] (*support-points based splitting*) exploitant un algorithme du plus proche voisin (*nearest neighbour*), afin d'optimiser la représentativité des jeux de données par rapport à ceux pouvant être obtenus par un découpage aléatoire.[31,32]

Notre seconde division, entre jeu d'entraînement et d'optimisation, exploitera quant à elle la méthode de validation croisée à k blocs (*k-folds cross-validation*). Le principe de la validation croisée repose sur une rotation de la séparation créée entre jeux d'entraînement et d'optimisation (voir figure 9).

Le jeu d'entraînement/optimisation est découpé, de façon aléatoire, en k blocs de données de taille égale, dont k-1 sont utilisés pour l'entraînement du modèle prédictif et 1 pour son optimisation. Cette opération est répétée k fois, en utilisant un jeu d'optimisation différent à chaque itération. L'évaluation de la performance globale s'effectue en évaluant la performance moyenne des k itérations. Cette méthode permet de limiter les biais potentiels générés par une simple dichotomie des données d'entraînement et d'optimisation en exploitant la totalité des données du lot afin d'effectuer ces deux tâches.

Comme démontré précédemment, une validation croisée *k-folds* avec $k = 17$ permettrait d'optimiser l'apprentissage et l'optimisation des modèles de cette étude.[29]

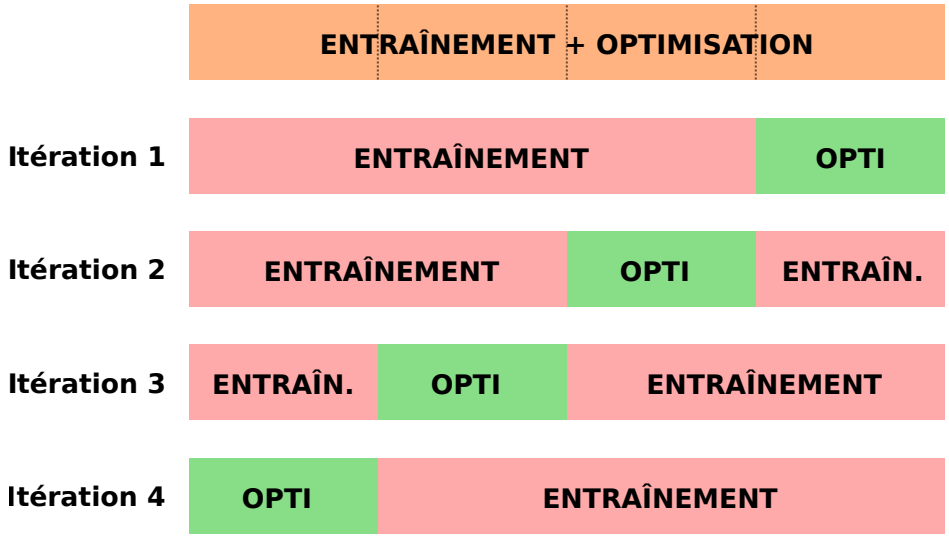

Figure 9: Principe de la validation croisée à k blocs (*k-fold cross-validation*), pour $k = 4$.

4.3 Modèles utilisés

4.3.1 Analyses discriminantes

Développements théoriques sur les modèles d'analyse discriminante linéaire, pénalisée, quadratique

[A FAIRE]

4.3.2 Modèle additif généralisé

Développements théoriques sur les modèles GAM...

[A FAIRE]

4.3.3 Arbres de décision

Développements théoriques sur les arbres décisionnels...

[A FAIRE]

4.3.4 Forêts aléatoires

Développements théoriques sur les modèles de forêts aléatoires...

[A FAIRE]

4.4 Optimisation par plans d'expérience (DOE)

Certains modèles nécessiteront une optimisation de leurs hyperparamètres, afin d'obtenir des performances maximales. Cette optimisation relève du domaine des plans d'expérience. De nombreux plans et stratégies sont envisageables, le choix dépendant en partie des caractéristiques du processus à optimiser.

En effet, l'optimisation des paramètres d'un modèle informatique présente quelques particularités notables ayant un impact sur l'utilisation des plans d'expérience :

- La réalisation d'une expérience supplémentaire a un coût faible,
- Plusieurs métriques relatives aux performances peuvent coexister,[a]
- La fonction de réponse peut s'avérer relativement complexe.

Ces particularités imposent d'explorer de manière méthodique la totalité de l'espace expérimental. Il existe une multitude de méthodes permettant de générer des plans expérimentaux dits SFD (*Space Filling Design*), afin d'optimiser l'occupation de l'espace expérimental. La méthode retenue pour cette étude sera celle des hypercubes latins, en raison de son utilisation répandue[33] et de sa simplicité conceptuelle.

La méthode des hypercubes latins est une extension du principe des carrés latins. Un carré latin est une grille $n \times n$, remplie de n éléments distincts arrangés de sorte que chaque ligne et chaque colonne ne contienne qu'un seul exemplaire de chacun des n éléments. Dans le domaine des plans d'expériences, l'application des carrés latins revient à diviser un domaine expérimental bidimensionnel en une grille $n \times n$, et à placer une expérience et une seule sur chaque ligne et chaque colonne.

L'application du concept de carré latin dans un domaine expérimental à trois dimensions aboutit au cube latin. La généralisation dans un espace n-dimensionnel mène au concept d'hypercube latin.

De nombreux plans expérimentaux basés sur les hypercubes latins peuvent être générés. Nous pouvons citer principalement trois types d'hypercubes latins :

- Aléatoires,
- Optimisés, afin d'optimiser l'occupation spatiale,
- Orthogonaux, visant à minimiser la corrélation des estimateurs des effets principaux.

Dans le cadre de cette étude, nous utiliserons des hypercubes latins quasi-orthogonaux, dont les propriétés nous permettront de modéliser de façon plus précise les performances de nos modèles en fonction de leurs paramètres de configuration (*hyperparamètres*).

Le but des plans expérimentaux de cette étude ne sera pas l'obtention d'une prédiction exacte de la valeur de la réponse, mais plus modestement la recherche des facteurs permettant d'obtenir cette réponse optimale. A cet effet, la modélisation de la performance pourra s'effectuer à l'aide d'un modèle quadratique avec interactions, de formule générale :

$$Y = \beta_0 + \sum_{i=1}^{k} \beta_i . X_i + \sum_{i<j}^{k} \sum_{j>1}^{k} \beta_{ij} . X_i . X_j + \sum_{i=1}^{k} + \beta_{ii} . X_i^2 + \varepsilon$$

Avec β_n les coefficients des effets principaux et X_n les facteurs réduits.

[a] cf. section 4.5

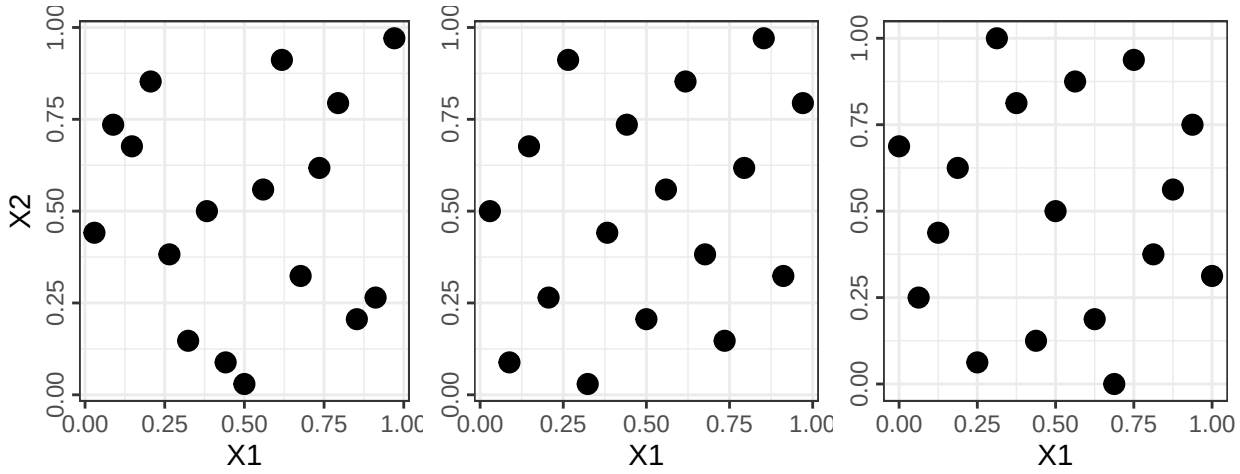

Figure 10: Carré latin aléatoire (à gauche), carré latin avec optimisation évolutive ESE maximin (au milieu), carré latin quasi-orthogonal (à droite)

4.5 Évaluation des performances des modèles

L'optimisation des modèles ainsi que la comparaison de leurs performances relatives implique nécessairement de définir quel sera le critère vis-à-vis duquel cette performance sera évaluée.

De nombreux critères sont utilisables, en fonction du cahier des charges défini pour la résolution du problème, mais également du type de tâche effectuée : régression, classification binaire, classification multiclasse.

Dans une tâche de classification binaire, les critères usuels sont la spécificité, la sensibilité, et l'aire sous la courbe de fonction d'efficacité du récepteur (*AUC ROC*, parfois abrégé en *ROC*). Il conviendra bien évidemment, avant d'utiliser des indicateurs tels que la spécificité et la sensibilité, de définir la notion de test positif et test négatif.

D'autres indicateurs d'intérêt existent, nous retiendrons ici l'index J de Youden[34] pondéré, qui permet de d'ajuster l'importance relative accordée à la spécificité et à la sensibilité, au sein d'un index synthétique.[35] Cet indicateur présente un intérêt particulier lorsqu'il apparaît souhaitable de tenir compte de la différence d'impact entre un faux positif et un faux négatif, sans pour autant autoriser des sensibilités ou spécificités trop faibles.

En l'espèce, l'index J de Youden pondéré nous permet d'élaborer un indice synthétique tenant compte du fait qu'il est plus grave de classer à tort comme comestible un champignon toxique que d'écarter à tort un champignon parfaitement comestible – sans pour autant autoriser le modèle à écarter un nombre inconsidéré de champignons comestibles.

L'index J de Youden pondéré est donné par :[35]

$$J_w = 2.\left(w.Sen + (1-w).Spe\right) - 1 \qquad avec \quad w \in [0;1]$$

Dans la classification binaire de cette étude, problème qui revient classiquement en mycologie à classer les espèces en fonction de leur toxicité, la valeur positive sera ici arbitrairement attribuée à la valeur "champignon toxique". Nous cherchons donc à maximiser la sensibilité de la détection, afin d'écarter les espèces toxiques, la spécificité – c'est-à-dire la capacité à ne pas écarter trop d'espèces comestibles – apparaissant alors comme un critère relativement secondaire. En établissant arbitrairement un index J de Youden pondéré accordant 10 fois plus d'importance à la sensibilité qu'à la spécificité, nous pouvons établir $w = 10/11$.

Le problème de classification binaire étant relativement simple (*comestible* ou *non-comestible*), nous fixerons arbitrairement le critère de performance minimum à atteindre à $J_w \geq 0.999$, soit :

$$\begin{cases} Sen_{max} = 1 \Leftrightarrow Spe_{min} = 0.9945 \\ Spe_{max} = 1 \Leftrightarrow Sen_{min} = 0.99945 \end{cases}$$

Dans les tâches de classification multiclasse, d'autres indicateurs d'intérêt pourront être utilisés, tels que le kappa de Cohen, l'indice de Rand (*accuracy*, ou précision), mais aussi la sensibilité et la spécificité moyennes.

Nous retiendrons dans le cadre de notre étude le kappa de Cohen,[36] calculé à partir de la matrice de confusion (cf. figure 11), et donné par :

$$\kappa = \frac{\pi_0 - \pi_e}{1 - \pi_e}$$

Avec π_0 la probabilité d'accord entre notre modèle et la classe réelle du champignon, et π_e la probabilité d'un même accord résultant du pur hasard.

Landis et Koch ont élaboré une échelle de validité du kappa de Cohen, avec un accord qualifié de *quasi-parfait* pour $\kappa > 0.80$.[37] Nous considérerons donc que ce critère sera le minimum requis pour qu'un modèle de classifieur multiclasse élaboré au cours de cette étude puisse être considéré comme ayant des performances acceptables.

L'interprétation du kappa pouvant parfois être assez contre-intuitive, cette étude la complétera parfois par l'indice de Rand R, métrique moins robuste en présence de données non-équilibrées[b], mais présentant l'avantage d'être de compréhension plus aisée, car représentant le pourcentage de prédictions exactes.

[b] Ce cas se présente habituellement lors d'une surreprésentation de certaines classes dans les jeux de données.

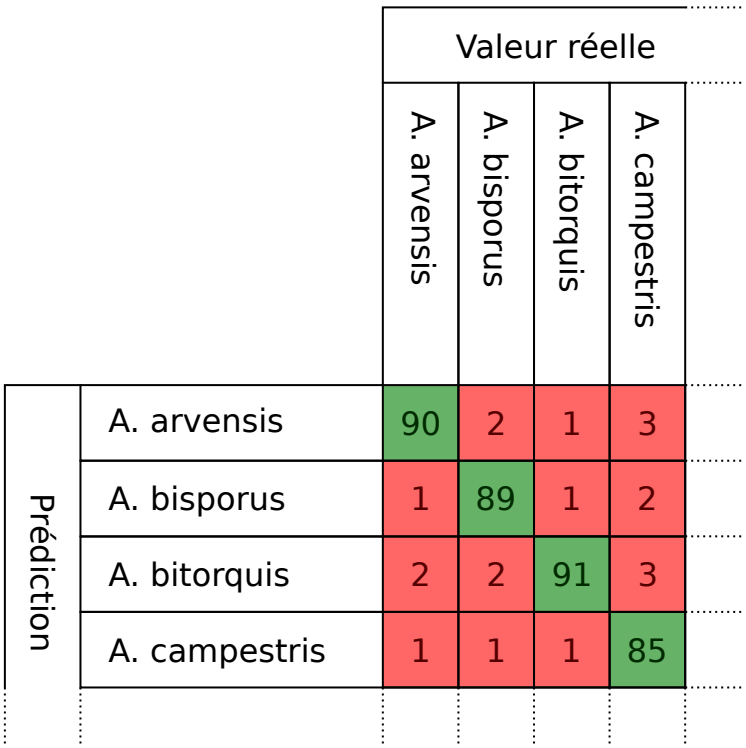

Figure 11: Extrait d'une matrice de confusion, pour une classification multiclasse

5 Apprentissage machine et classification binaire

BROUILLON, le lot de données utilisé ici est le Secondary Mushroom Dataset de D.Wagner.

5.1 Analyse exploratoire des données (EDA)

Partie permettant de présenter globalement le lot de données synthétiques créé dans cette étude (cf. section 3.2.1) avant utilisation des IA.

La génération du lot de données fonctionne, la génération des graphiques aussi, mais le texte sera à retravailler en fonction du lot de données synthétiques obtenu. . .

Le lot de données d'origine contient 61069 spécimens de champignons, caractérisés par 21 propriétés morphologiques ou environnementales. La structure de ce lot de données est résumé dans le tableau 1.

Ce lot de données original a été découpé en un jeu d'apprentissage/optimisation et un jeu de données d'évaluation, avec un rapport 16:1, conformément aux principes mentionnés dans nos développements précédents.[c]

[c] cf. section 4.1, page 29.

Table 1: Structure du lot de données initial

	Type	Niveaux
class	factor	2
cap.diameter	numeric	2571
cap.shape	factor	7
cap.surface	factor	12
cap.color	factor	12
does.bruise.or.bleed	factor	2
gill.attachment	factor	8
gill.spacing	factor	4
gill.color	factor	12
stem.height	numeric	2226
stem.width	numeric	4630
stem.root	factor	6
stem.surface	factor	9
stem.color	factor	13
veil.type	factor	2
veil.color	factor	7
has.ring	factor	2
ring.type	factor	9
spore.print.color	factor	8
habitat	factor	8
season	factor	4

Toutes les distributions des variables du lot d'entraînement ont ensuite été tracées par histogrammes pour les variables numériques, et diagrammes en barres pour les variables alphabétiques et catégorielles.

Les diagrammes en barres n'ont rien illustré de particulièrement remarquable et n'ont pas été inclus dans le rapport. Toutefois, les distributions dimensionnelles sont plus intéressantes : à première vue, elles semblent suivre une courbe en cloche (figure 12), avec une longue queue à droite. Une transformation logarithmique (figure 13) montre plus nettement la forme de cette queue.

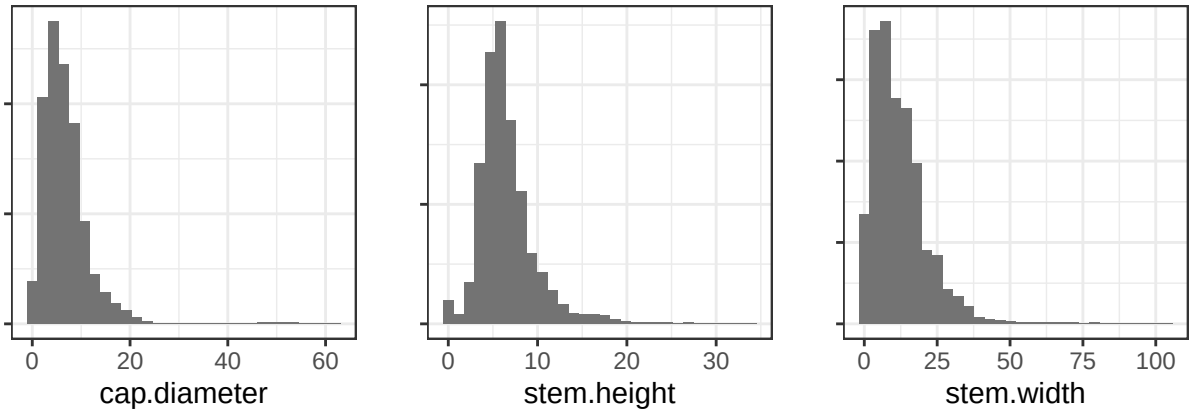

Figure 12: Distribution des diamètres de chapeau, longueur de stipe, diamètre de stipe

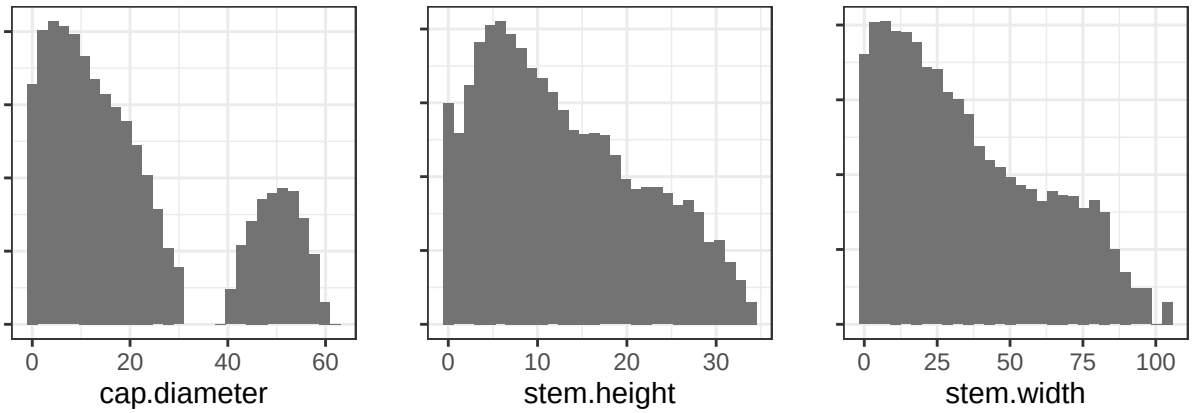

Figure 13: Distribution des diamètres de chapeau, longueur de stipe, diamètre de stipe (échelle logarithmique en ordonnée)

La distribution du diamètre du chapeau D_C a l'apparence d'une courbe en cloche, avec une longue queue à droite, mais est en réalité bimodale, avec un mode principal à 5 cm, et un mode secondaire beaucoup plus petit pour $D_C \approx 50$ cm. Cette taille exceptionnelle est attribuable à des variétés telles que *Polyporus squamosus*.[22]

La distribution de la longueur de stipe L_S a également une forme de courbe en cloche avec une longue queue à droite, un mode principal à 5 cm et un mode secondaire à 0 cm. Cette valeur

peut également sembler surprenante, mais certains champignons du lot n'ont pas de pied, ce qui explique cette valeur.

La distribution du diamètre de stipe D_S a aussi l'apparence d'une courbe en cloche avec une longueur queue à droite, et un pic à $D_S \approx$ 10-15 mm. Dans toutes ces distributions, la longue queue à droite peut probablement s'expliquer par l'utilisation, dans le *Secondary Mushroom Dataset*, d'une distribution normale pour chaque variété, associée à l'impossibilité d'avoir des valeur dimensionnelles négatives.

5.2 Optimisation et sélection des modèles

Il existe une grande variété de modèles exploitables pour bâtir un système d'apprentissage machine. Cette section expliquera la stratégie utilisée pour l'évaluation de certains de ces modèles, ainsi que pour l'exploration de l'espace de leurs hyperparamètres à fins d'optimisation et la mesure de leurs performances.

Les modèles sélectionnés pour cette étude sont de types variés :

- Analyse discriminante linéaire (LDA) : *Linear Discriminant Analysis* (lda2), *Penalized Discriminant Analysis* (pda)
- Modèle additif généralisé (GAM) : *Generalized Additive Model using LOESS* (gamLoess)
- Modèle arborescent : *Classification And Regression Tree* (CART) (rpart, rpartCost), *Single C5.0 Tree* (ctree)
- Forêt aléatoire : *Random Ferns* (rferns), *Random Forest* (ranger, Rborist)

5.2.1 Stratégie d'optimisation

Les algorithme d'apprentissage machine développés au cours de cette étude mettent en œuvre les méthodes présentées dans les sections précédentes afin d'effectuer automatiquement les tâches suivantes :

1. Découpage du lot de données en un jeu d'entraînement/optimisation et en un jeu de validation, avec adaptation des rapports de taille en fonction du volume de données du lot initial,[d]
2. Apprentissage sur le jeu d'entraînement, exploitant une validation croisée à k blocs, avec adaptation du nombre de blocs à la taille du lot de données,[e]
3. Exploration de l'ensemble de l'espace expérimental des hyperparamètres du modèle, via la méthode des hypercubes latins quasi-orthogonaux,\footnote{cf. section 4.4, page 31
4. Mesure des performances en exploitant une métrique adaptée,[f]
5. Modélisation des performances en fonction des hyperparamètres, via un modèle quadratique avec interactions,[f]
6. Sélection des hyperparamètres permettant d'optimiser les performances du modèle,
7. Mesure des performances de chaque modèle avec les hyperparamètres optimaux,
8. Sélection des modèles les plus performants pour prédiction et mesure finale des performances contre le lot d'évaluation,
9. Génération, sauvegarde et insertion automatique de tous les graphiques et données numériques dans le texte de la présente étude.

[d] cf. section 4.1, page 29
[e] cf. section 4.2, page 30
[f] cf. section 4.5, page 33

5.2.2 Modèles d'analyse discriminante

Les modèles d'analyse discriminante choisis pour cette étude sont lda2 (LDA : *Linear Discriminant Analysis*) et pda (*Penalized Discriminant Analysis*). Le modèle lda2 dispose d'un seul hyperparamètre (*dimen*, nombre de fonctions discriminantes). Le modèle pda a également un unique hyperparamètre (*lambda*, pénalité de réduction des coefficients).

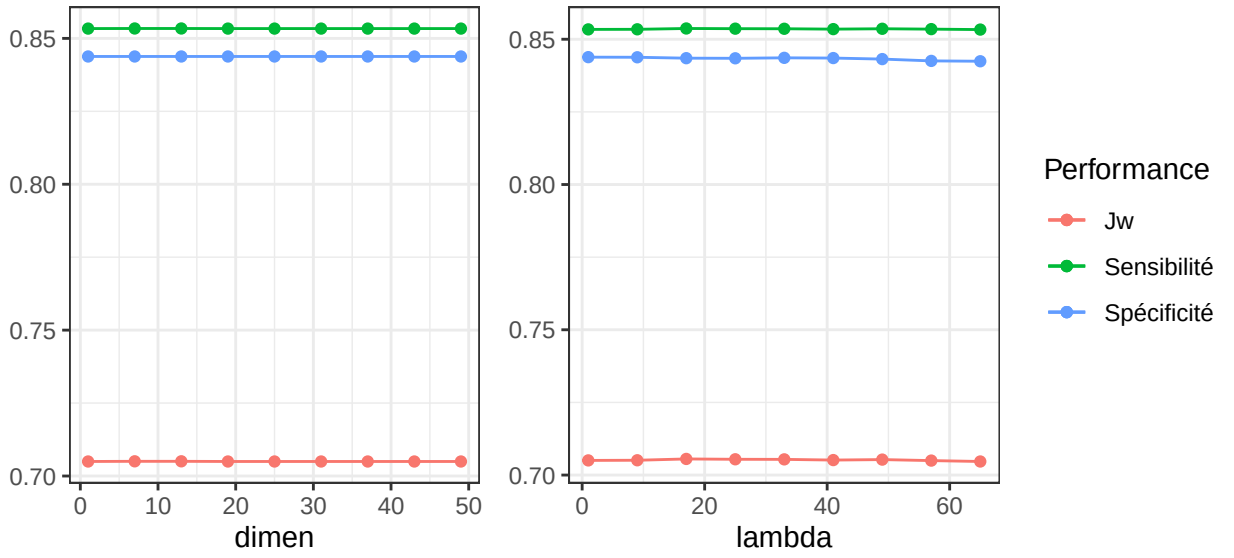

Figure 14: Performances des modèles lda2 (à gauche) et pda (à droite)

Comme l'illustre la figure 14, les performances des modèles PDA et LDA sont très proches, et relativement constantes sur la totalité de l'espace expérimental de leurs hyperparamètres.

Le paramètre *dimen* du modèle lda2 ne semble en effet pas avoir d'effet significatif sur ses performances, avec un index J de Youden pondéré relativement constant ($J_{w_{moy}} = 0.705$).

De même, le paramètre *lambda* du modèle pda n'impacte ses performances que de matière très marginale, avec des lambdas faibles donnant une légère amélioration des résultats ($J_{w_{max}} = 0.706$).

Toutefois, les performances de ces deux modèles restent malheureusement insuffisantes pour notre étude, aussi bien en sensibilité qu'en spécificité :

$$\begin{cases} Sen \approx 0.853 \\ Spe \approx 0.848 \end{cases}$$

Ces performances médiocres s'expliquent par le fonctionnement même des modèles d'analyse discriminante qui, s'ils peuvent analyser des données qualitatives à fins de classifications, ne peuvent le faire que si une quantification sous-jacente est possible, par exemple :

- Données binaires ou booléennes,
- Données catégorielles basées sur des données quantitatives.

L'inclusion de ces modèles, présentant ici des performances très modestes, a un intérêt essentiellement didactique, permettant de souligner l'intérêt d'une connaissance élémentaire des fondamentaux mathématiques et algorithmiques des modèles d'apprentissage machine mis en œuvre, pour en connaître les limites ou évaluer les besoins de nettoyage préalable des données avant déploiement de l'apprentissage machine, afin d'éviter de confronter certains modèles face à des problèmes de classification pour lesquels ils n'ont pas été conçus.

5.2.3 Modèle additif généralisé

Le seul modèle additif généralisé choisi pour cette étude est gamLoess (*Generalized Additive Model using Locally Weighted Linear Regression*). Le modèle gamLoess dispose de deux hyperparamètres : *span* (fraction de points utilisés dans l'environnement local) and *degree* (degré de linéarisation).

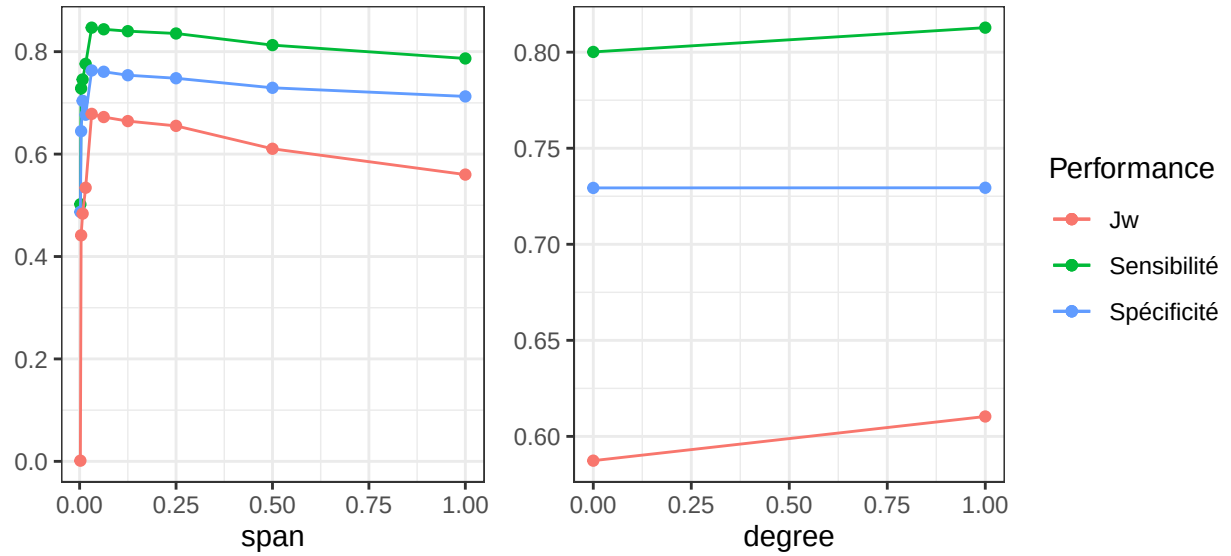

Figure 15: Performances de gamLoess en fonction du span (gauche) and du degré (droite)

L'hyperparamètre *degree* du modèle gamLoess a un impact mineur sur ses performances, avec un écart entre indices J de Youden maximal et minimal de $\Delta J_w = 0.023$.

L'hyperparamètre *span* affecte marginalement la sensibilité de gamLoess, avec une valeur optimale de $span = 0.03125$, aboutissant à un index J de Youden maximal de $J_{w_{max}} = 0.679$.

Les performances de ce modèle n'atteignent pas le critère posé pour notre étude. Le modèle additif généralisé s'est avéré inférieur aux modèles d'analyse discriminante linéaire, aussi bien en en sensibilité (0.847 vs 0.853) qu'en spécificité (0.763 vs 0.844).

Les performances médiocres de ce modèle s'expliquent par les mêmes raisons que celles des modèles d'analyse discriminante.\footnote{cf. section 5.2.2)

5.2.4 Modèles d'arbres de décision

Les modèles basés sur des arbres de décision ont un intérêt tout particulier pour cette étude, pour deux raisons majeures :

- La logique en arbre de décision est habituellement usitée pour la classification manuelle des champignons,
- Les arbres de décision obtenus peuvent être tracés, et facilement interprétés par l'humain.

Les premiers modèles présentés dans le cadre de notre étude sont deux modèles CART (*Classification And Regression Tree*). Le modèle CART le plus simple proposé dans notre étude (rpart) ne dispose que d'un seul hyperparamètre : *cp* (complexité).

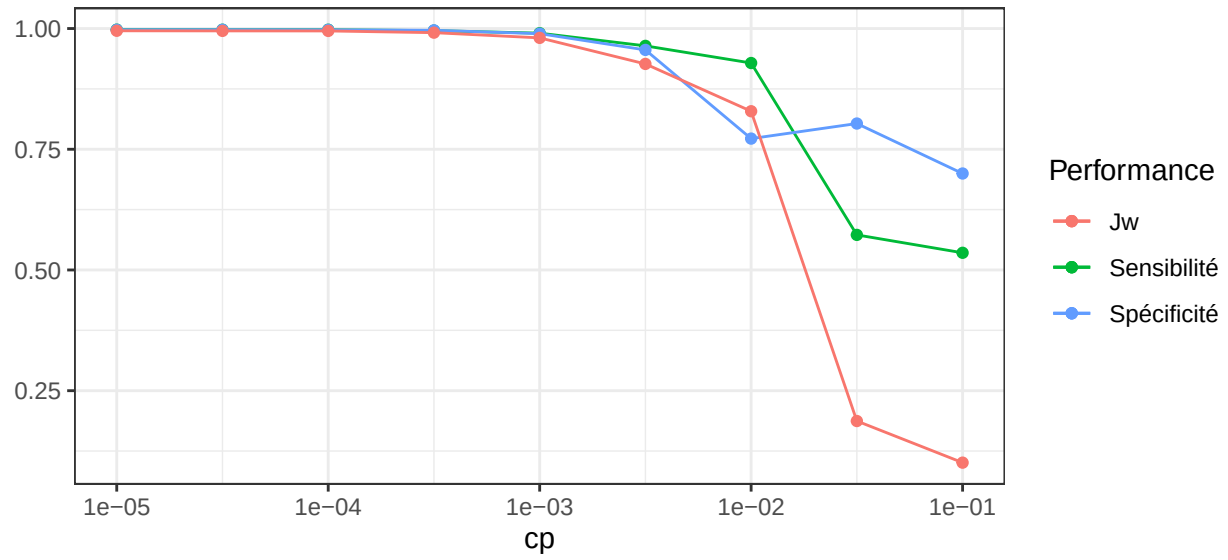

Figure 16: Performances du modèle rpart en fonction du paramètre de complexité (cp)

Le modèle CART le plus simple n'atteint jamais les performances requises, le critère étant $Jw \geq 0.999$. Toutefois, ce modèle s'en approche, et donne de bons résultats globaux, avec :

$$\begin{cases} J_{w_{max}} = 0.9954 \\ Spe_{Jw_{max}} = 0.9974 \\ Sen_{Jw_{max}} = 0.9977 \end{cases}$$

Le second modèle CART utilisé dans cette étude (rpartCost) associe des hyperparamètres de complexité (*cp*) et de coût (*Cost*). Les graphiques de sensibilité et de spécificité en fonction des hyperparamètres (figure 17) illustrent bien, dans leur partie supérieure ($cp \geq 0.05$) la notion classique de compromis entre sensibilité et spécificité : dans cette zone, toute amélioration de la sensibilité se fera inévitablement au détriment de la spécificité, et réciproquement.

En pratique, pour $cp \geq 0.05$, notre modèle d'IA basé sur ce type d'arbre de décision se montrera soit excessivement sévère, rejetant un nombre considérable de champignons comestibles (quadrant supérieur gauche, $cost \leq 1.5$), soit au contraire excessivement laxiste, admettant un nombre important de champignons non-comestibles (quadrant supérieur droit, $cost \geq 1.5$).

C'est dans la section inférieure de ces graphiques ($cp \leq 0.025$) que le modèle montre une performance acceptable tant en sensibilité qu'en spécificité.

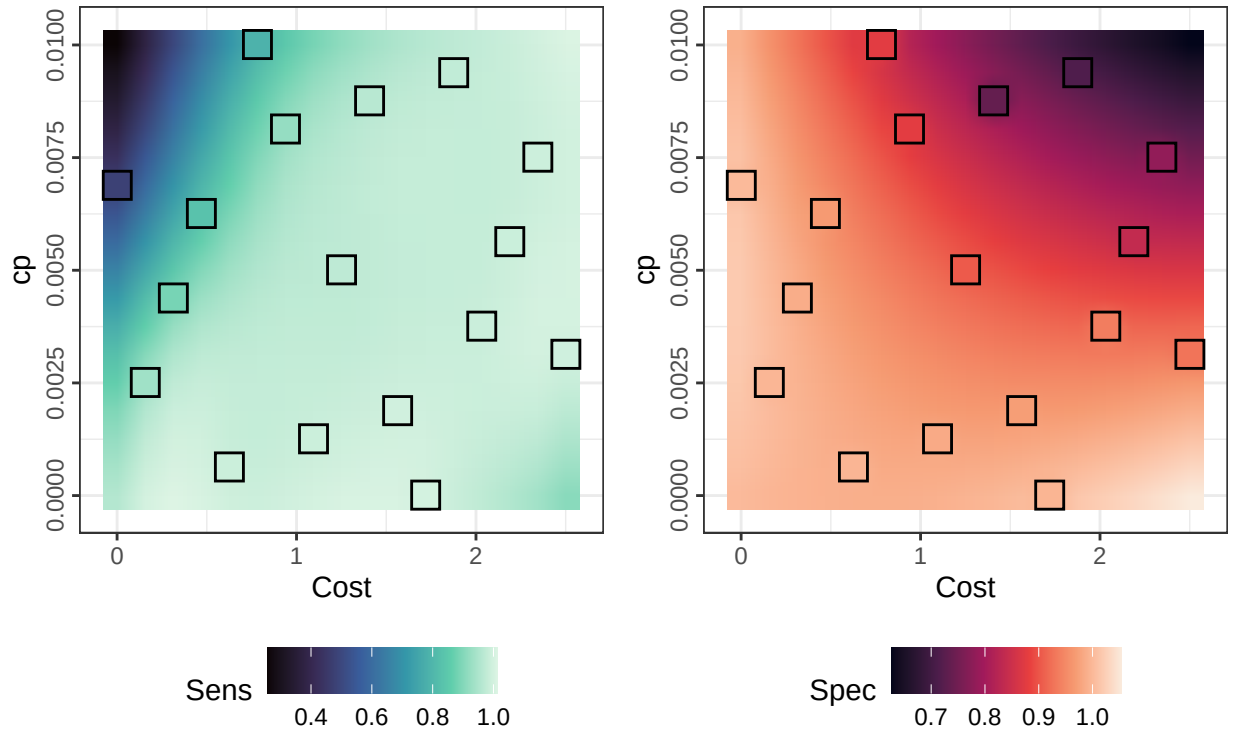

Figure 17: Sensibilité (à gauche) et spécificité (à droite) de rpartCost en fonction de la complexité et du coût (interpolation quadratique, points expérimentaux encadrés en noir)

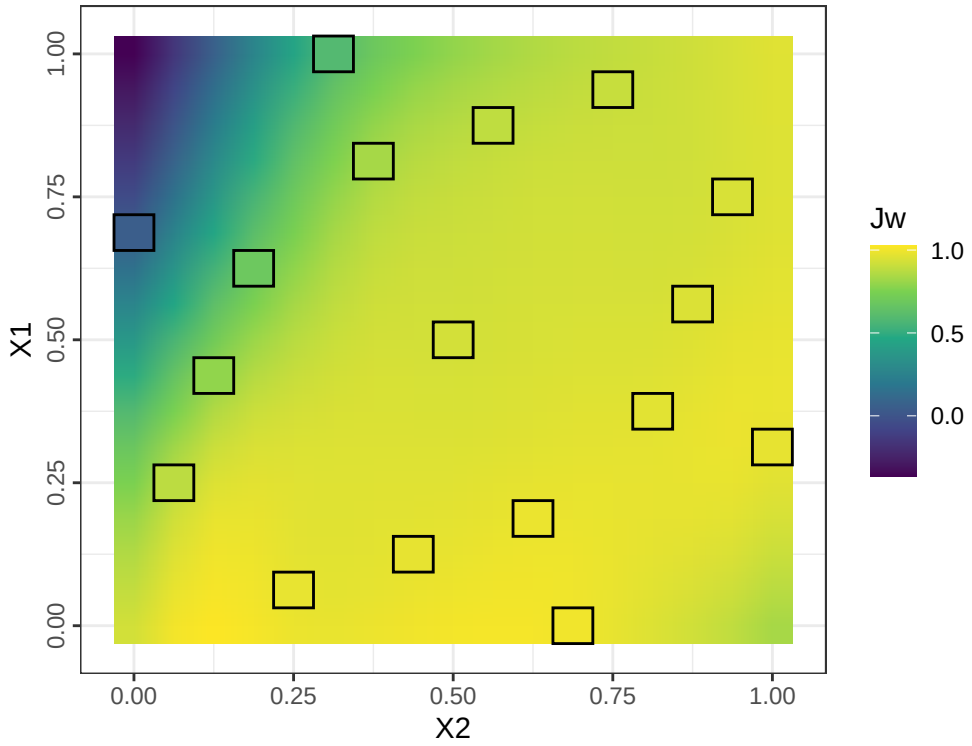

Figure 18: Performances (index J de Youden pondéré 10:1) de rpartCost en fonction des hyperparamètres réduits de complexité X1 et de coût X2 (interpolation quadratique, points expérimentaux encadrés en noir)

En posant comme facteurs réduits :

- $X_1 \in [0;1]$ pour le paramètre *minNode*,
- $X_2 \in [0;1]$ pour le paramètre *predFixed*,

Nous pouvons modéliser la réponse Y (J_w) par un modèle quadratique avec interaction[g] :

$$Y = b_0 + b_1.X_1 + b_2.X_2 + b_{12}.X_1.X_2 + b_{11}.X_1^2 + b_{22}.X_2^2$$

Avec Y l'indice J de Youden pondéré, X_1 le facteur réduit dans la plage [0;1] associé à l'hyperparamètre de complexité (*cp*), X_2 le facteur réduit associé à l'hyperparamètre de coût (*Cost*) et b_n les coefficients des effets. La modélisation permet de calculer les effets suivants:

$$\begin{cases} b_0 = 0.813 \\ b_1 = -0.7723 \\ b_2 = 1.2453 \end{cases} \qquad \begin{cases} b_{12} = 1.3354 \\ b_{11} = -0.3065 \\ b_{22} = -1.2955 \end{cases}$$

Les performances maximales seront ici atteintes pour :

$$\begin{cases} X_1 = 0 & soit \quad cp = 1e-05 \\ X_2 = 0.5 & soit \quad Cost = 1.25 \end{cases}$$

[g] cf. section 4.4, page 31

Ces hyperparamètres optimaux permettent au modèle d'atteindre :

$$\begin{cases} J_{w_{max}} = 0.995 \\ Spe_{Jw_{max}} = 0.9966 \\ Sen_{Jw_{max}} = 0.9976 \end{cases}$$

Les performances du modèle rpartCost, bien qu'excellentes, ne permettent pas d'atteindre l'index J de Youden requis.

Le dernier modèle d'arbre décisionnel proposé dans notre étude est C5.0 tree (c50tree). Ce modèle ne dispose d'aucun hyperparamètre.

Les performances obtenues sont :

$$\begin{cases} J_w = 0.9978 \\ Spe = 0.9988 \\ Sen = 0.9989 \end{cases}$$

De manière assez surprenante, bien que ne disposant d'aucun hyperparamètre, ce modèle a donné d'excellents résultats sans aucune optimisation nécessaire. Toutefois, le modèle C5.0 tree n'a pas rempli l'objectif posé par le critère $J_w \geq 0.999$.

Les modèles d'arbres de classification sont particulièrement adaptés aux problèmes de classification avec variables quantitatives et surtout qualitatives, et ont pu s'illustrer dans cette étude en fournissant des résultats très acceptables ($Spe \geq 0.994$, $Sen \geq 0.998$), mais n'atteignant pas pour autant les exigences imposées par le critère de performance que nous avons défini pour les classifieurs binaires de notre étude.

5.2.5 Forêts aléatoires

Le premier modèle de forêt aléatoire évalué dans notre étude est le modèle de fougères aléatoires rFerns (*Random Ferns*). Ce modèle ne possède qu'un seul hyperparamètre, la profondeur (*depth*).

Quoique très efficient sur le plan calculatoire, le modèle de fougères aléatoires a fourni des résultats assez peu satisfaisants, avec :

$$\begin{cases} J_{w_{max}} = 0.561 \\ Spe_{Jw_{max}} = 0.803 \\ Sen_{Jw_{max}} = 0.778 \end{cases}$$

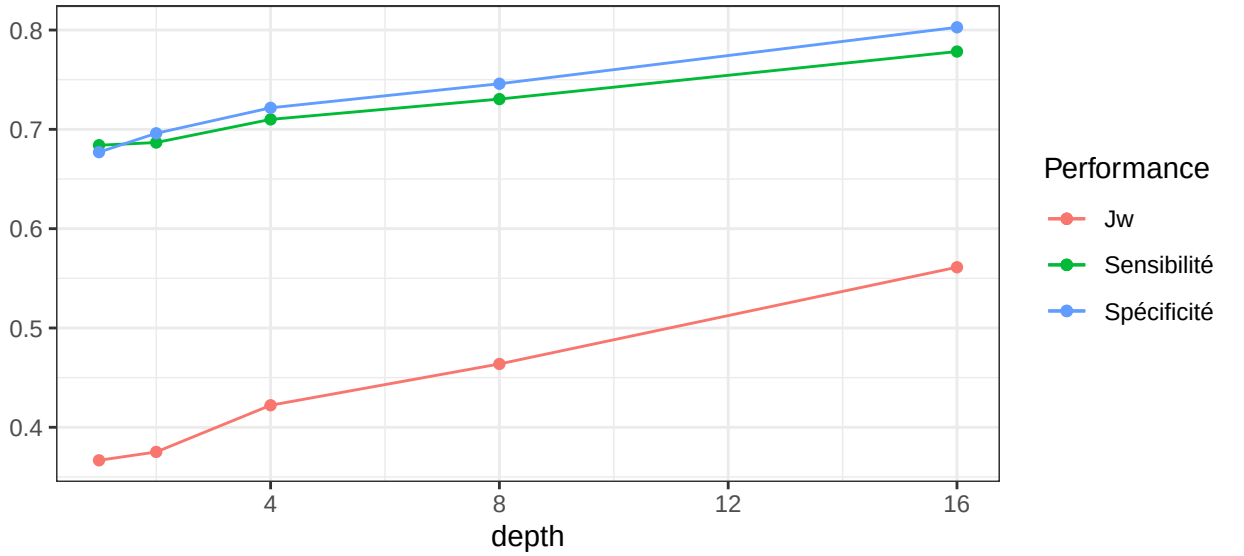

Figure 19: Performances du modèle de fougères aléatoires

Le second modèle de forêt aléatoire évalué dans cette étude est Rborist. Deux hyperparamètres régissent ce modèle : le nombre de prédicteurs testés pour une scission (*predFixed*) et le nombre minimal de lignes-références distinctes avant de scinder un nœud (*minNode*).

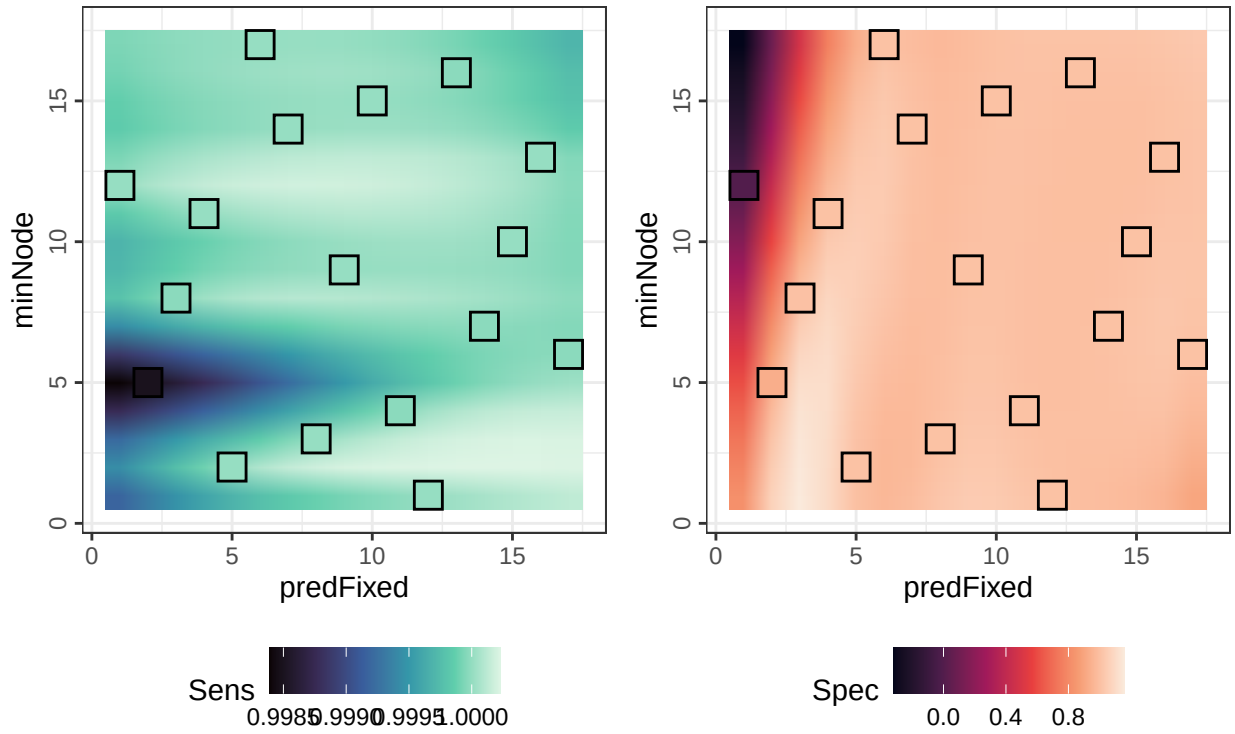

Figure 20: Sensibilité (à gauche) et spécificité (à droite) du modèle Rborist en fonction de ses deux hyperparamètres (interpolation quadratique, points expérimentaux encadrés en noir)

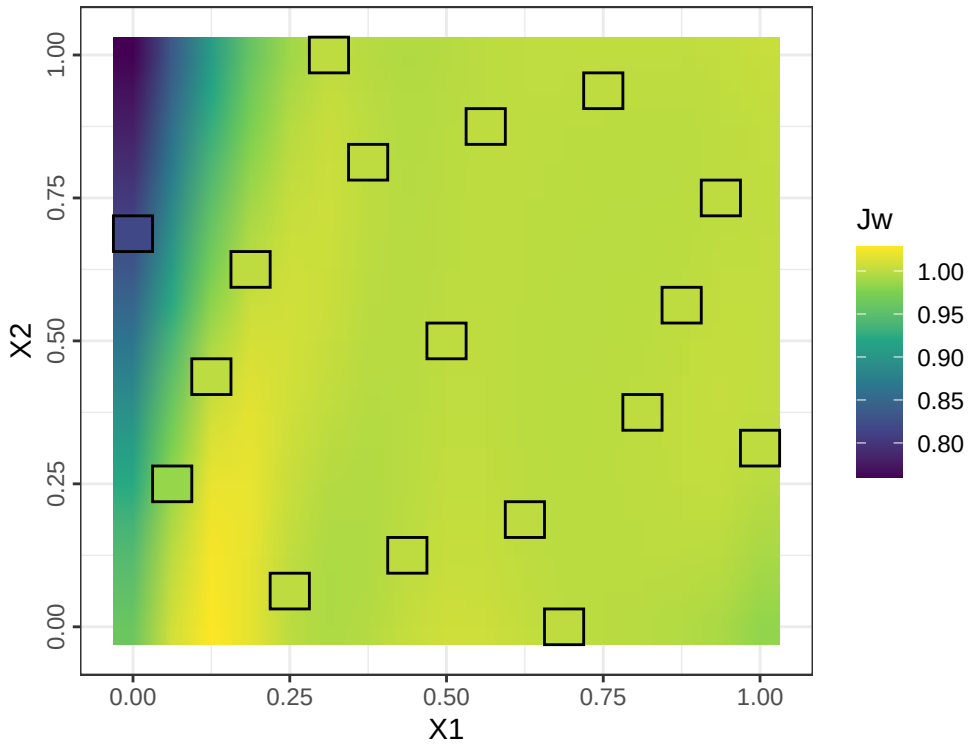

Figure 21: Performance (indice J de Youden pondéré) du modèle Rborist en fonction de ses deux hyperparamètres réduits de nombre de prédicteurs X1 et de nombre de références avant scission X2 (interpolation quadratique, points expérimentaux encadrés en noir)

Nous pouvons modéliser la réponse par un modèle quadratique avec interaction[h] :

$$Y = b_0 + b_1.X_1 + b_2.X_2 + b_{12}.X_1.X_2 + b_{11}.X_1^2 + b_{22}.X_2^2$$

Avec Y l'indice J de Youden pondéré J_w, X_1 le facteur réduit dans la plage [0;1] associé à l'hyperparamètre de nombre de prédicteurs testés par scission (*predFixed*), X_2 le facteur réduit associé à l'hyperparamètre de nombre minimal de références distinctes avant scission (*minNode*) et b_n les coefficients des effets. La modélisation permet de calculer les effets suivants:

Le calcul numérique nous permet d'obtenir les estimation des effets :

$$\begin{cases} b_0 = 0.9238 \\ b_1 = 0.421 \\ b_2 = -0.1161 \end{cases} \qquad \begin{cases} b_{12} = 0.1851 \\ b_{11} = -0.3877 \\ b_{22} = -0.0143 \end{cases}$$

Ce modèle quadratique avec interactions permet d'évaluer les hyperparamètres optimaux permettant de maximiser l'indice J de Youden pondéré ($minNode = 1$ et $predFixed = 10$) afin de lancer la prédiction sur un modèle optimisé. Les performances obtenues sont excellentes :

$$\begin{cases} J_{w_{max}} = 1 \\ Spe_{Jw_{max}} = 1 \\ Sen_{Jw_{max}} = 1 \end{cases}$$

Le dernier modèle de forêt aléatoire que nous évaluons dans cette étude est le modèle ranger. Celui-ci dispose de trois hyperparamètres : la taille minimale de nœud (*min.node.size*), le nombre de caractéristiques à séparer à chaque nœud (*mtry*) et la règle contrôlant cette séparation (*splitrule*).

[h] cf. section 4.4, page 31

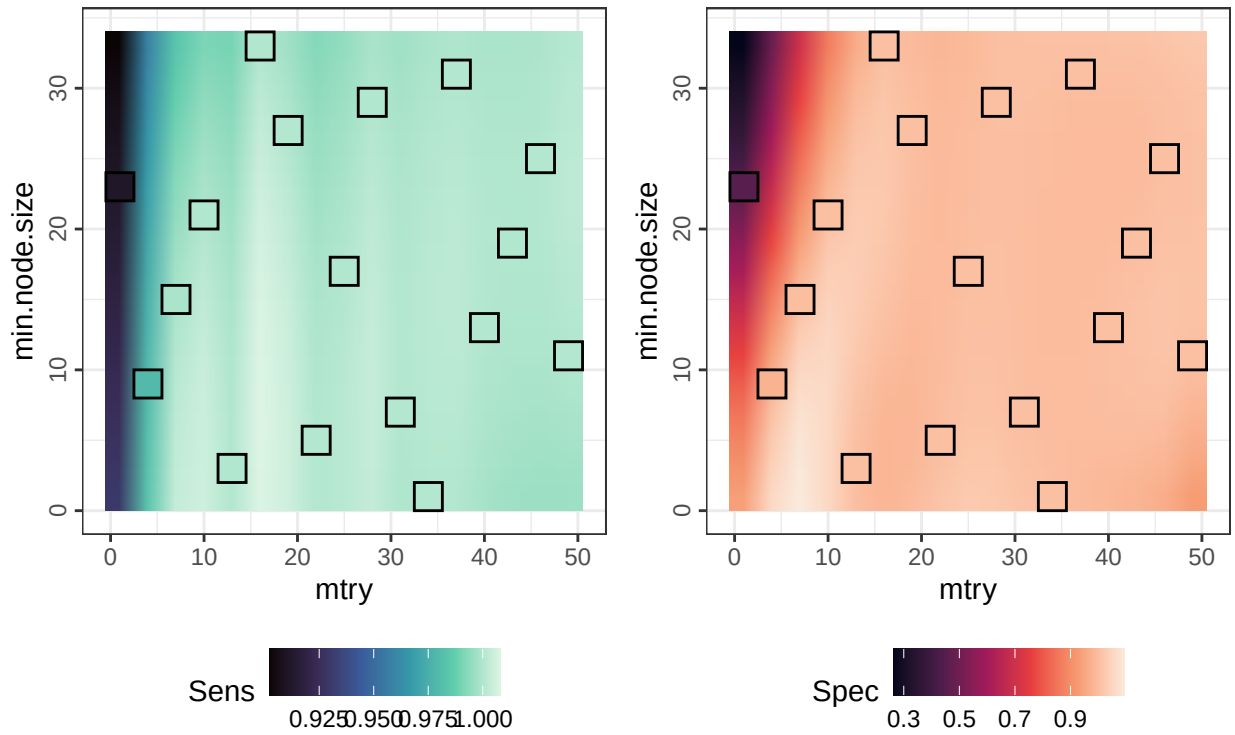

Figure 22: Sensibilité (à gauche) et spécificité (à droite) du modèle Ranger, en fonction des 2 hyperparamètres (algorithme de scission : gini)

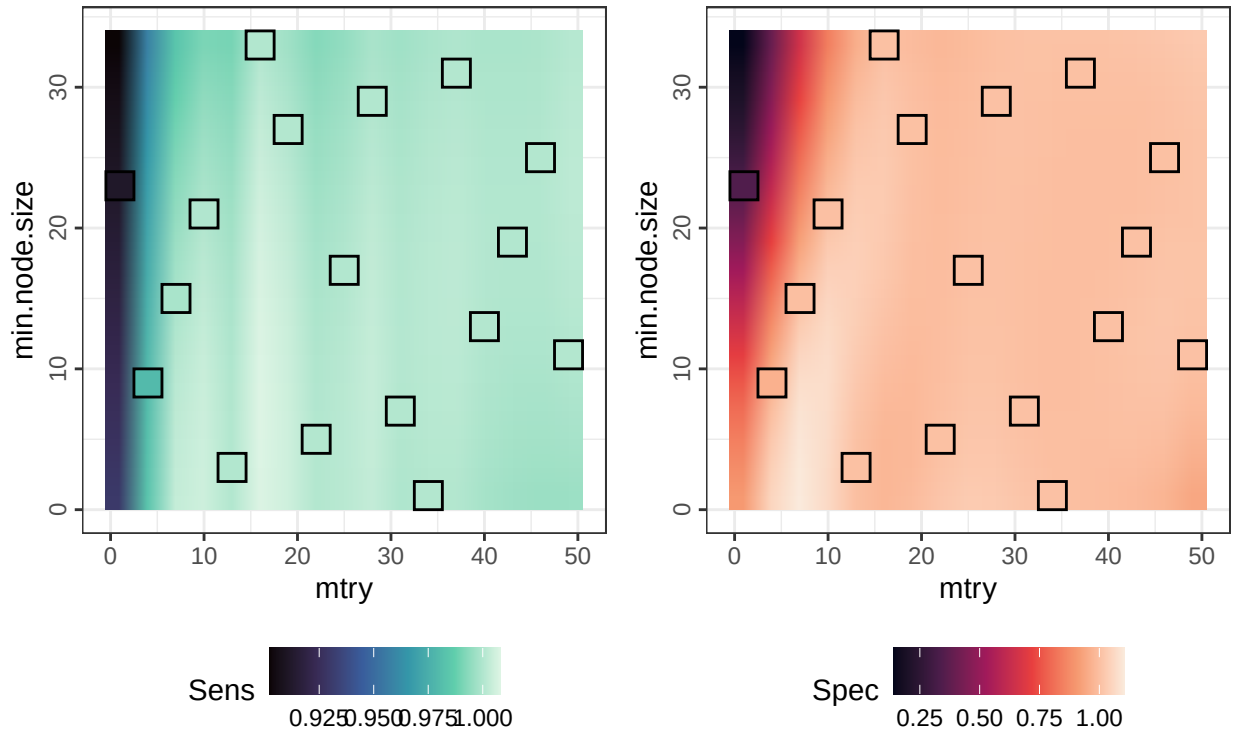

Figure 23: Sensibilité (à gauche) et spécificité (à droite) du modèle Ranger, en fonction des 2 hyperparamètres (algorithme de scission : extratrees)

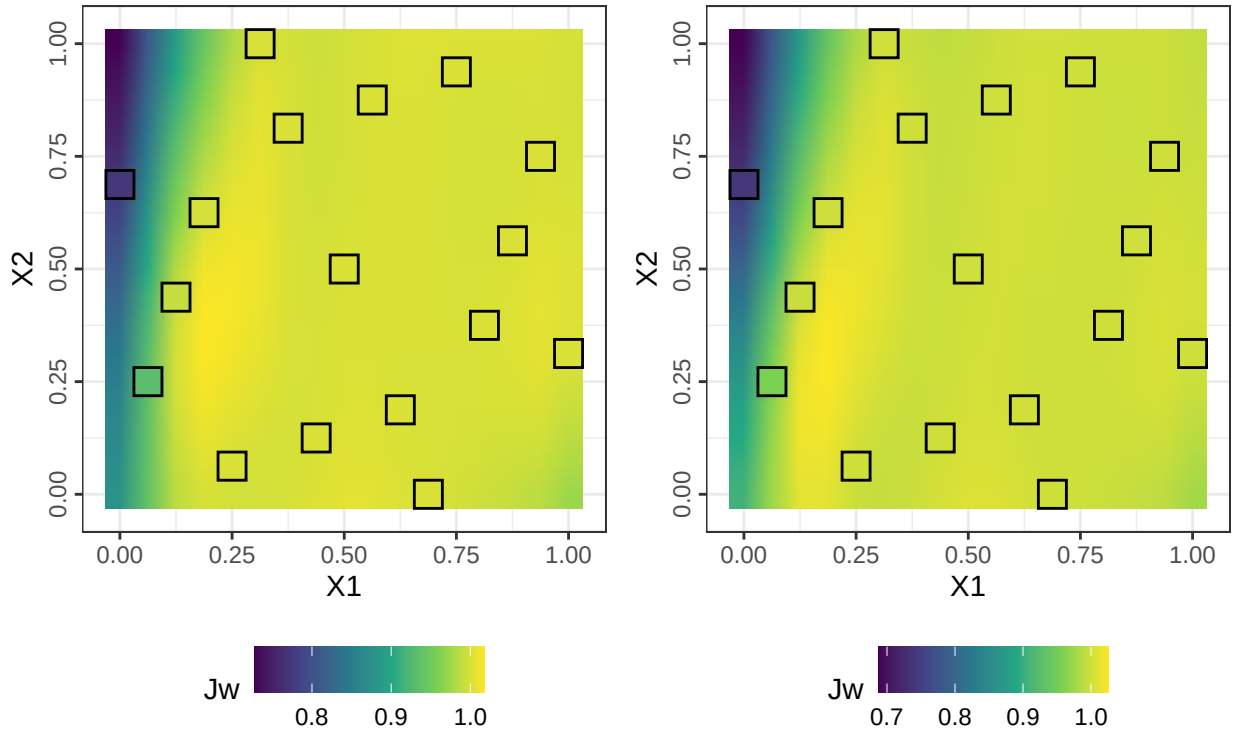

Figure 24: Performances du modèle Ranger en fonction de l'algorithme de scission (extratrees à gauche, gini à droite) et des hyperparamètres réduits : caractéristiques à séparer X1 et taille minimale de noeud X2 (interpolation quadratique, points expérimentaux encadrés en noir)

Nous pouvons proposer pour ce modèle la modélisation quadratique suivante :

$$Y = b_0 + b_1.X_1 + b_2.X_2 + b_3.X_3 + b_{12}.X_1.X_2 + b_{23}.X_2.X_3 + b_{13}.X_1.X_3 + b_{11}.X_1^2 + b_{22}.X_2^2$$

Avec Y l'indice J de Youden pondéré, X_1 le facteur réduit associé au paramètre de taille minimale de nœud (*min.node.size*), X_2 le facteur réduit associé au paramètre de nombre de caractéristiques à séparer à chaque nœud (*mtry*), X_3 le facteur régissant la règle de séparation (*splitrule*, la valeur 0 étant attribuée à *gini*, 1 à *extratrees*) et b_n les estimations des coefficients des effets. Le facteur X_3 n'ayant que deux niveaux, il est évidemment impossible de lui attribuer une composante quadratique.

La modélisation permet de calculer les effets suivants :

$$\begin{cases} b_0 = 0.9224 \\ b_1 = 0.3981 \\ b_2 = -0.0618 \\ b_3 = -0.0036 \end{cases} \qquad \begin{cases} b_{12} = 0.1829 \\ b_{23} = 0.0081 \\ b_{13} = -8 \times 10^{-4} \\ b_{11} = -0.3957 \\ b_{22} = -0.0536 \end{cases}$$

Le modèle ranger semble déjà, par simple interprétation graphique (voir figures 22, 23 et 24), donner de bons résultats sur une très large plage de l'espace expérimental de ses hyperparamètres, même avec un nombre réduit d'arbres ($n = 6$). Une optimisation des hyperparamètres grâce à

la modélisation quadratique ($min.node.size = 20$, $mtry = 32$ et $splitrule = extratrees$) a donné d'excellents résultats :

$$\begin{cases} J_{w_{max}} = 1 \\ Spe_{Jw_{max}} = 1 \\ Sen_{Jw_{max}} = 1 \end{cases}$$

Le modèle Ranger a donné des résultats similaires à ceux du modèle Rborist, avec une sensibilité, une spécificité et un indice de Youden excellents.

Ces résultats soulignent un fait intéressant : tous les modèles de forêts aléatoires ne sont pas égaux. Notre étude montre une différence considérable en sensibilité et spécificité entre les forêts aléatoires de type rFerns, Ranger et Rborist. Lors des étapes préliminaires de cette étude, d'autres algorithmes de forêts aléatoires ont montré de grandes disparités d'efficience sur le plan calculatoire, ce qui nous a conduit à écarter certains modèles pour des raisons pratiques, alors que d'autres se sont avérés sensiblement plus rapides et ont donc pu être retenus pour notre étude.

5.3 Résultats

5.3.1 Protocole d'évaluation

Les modèles ayant atteint les performances requises ($J_w \geq 0.999$) lors de l'étape d'optimisation ont été choisis pour l'évaluation. Les deux modèles retenus sont deux modèles de type forêt aléatoire :

- Forêt aléatoire avec algorithme de type Ranger,
- Forêt aléatoire avec algorithme de type Rborist.

Tous les modèles ont été entraînés sur le jeu de données d'apprentissage, après application des hyperparamètres optimaux obtenus précédemment[i] par modélisation des performances via un modèle quadratique avec interactions. Les performances de nos modèles face au jeu de données d'évaluation, auquel ils n'ont encore jamais été exposés[j], seront évaluées avec le même critère que précédemment : $J_w \geq 0.999$

5.3.2 Performances des modèles de forêts aléatoires

La matrice de confusion du modèle ranger (table 2) donne les résultats détaillés de ses prédictions.

[i] cf. section 5.2.5

[j] cf. section 4.2, 30

Table 2: Matrice de confusion du modèle Ranger (prédictions à gauche, référence en haut)

	toxique	comestible
toxique	1994	0
comestible	0	1599

La forêt aléatoire de type Ranger a donné d'excellents résultats, sa précision finale étant égale à 1, avec un intervalle de confiance à 95% de [0.999 ; 1], le tout en un temps raisonnable (24.02min), preuve de son efficience calculatoire.

La forêt aléatoire de type Rborist a donné des résultats similaires, atteignant une précision finale égale à 1, avec un intervalle de confiance à 95% de [0.999 ; 1]. Le modèle Rborist, donnant des résultats sensiblement identiques à Ranger, s'est de plus avéré extrêmement efficient sur le plan calculatoire (3.36min).

Table 3: Performances des modèles Ranger et Rborist (jeu d'évaluation)

	Sensibilité	Spécificité	J de Youden	Durée (min)
Ranger	1	1	1	24.02
Rborist	1	1	1	3.36

6 Apprentissage machine et classification multiclasse

Brouillon, le lot de données utilisé ici est un lot synthétique créé par moi-même (avec algorithme de création fonctionnel), mais à partir des données primaires du Secondary Mushroom Dataset de D.Wagner.

Étant données les performances qu'ont montré les différents modèles lors de la classification binaire, seuls les modèles basés sur les arbres décisionnels et les forêts aléatoires seront évalués dans cette section.

6.1 Classification par familles

6.1.1 Modèles d'arbres de décision

Brouillon, à étoffer et finir, peut-être avec plus de modèles.

Le modèle d'arbre de décision présenté dans cette partie est rpart, le plus simple des modèles CART.

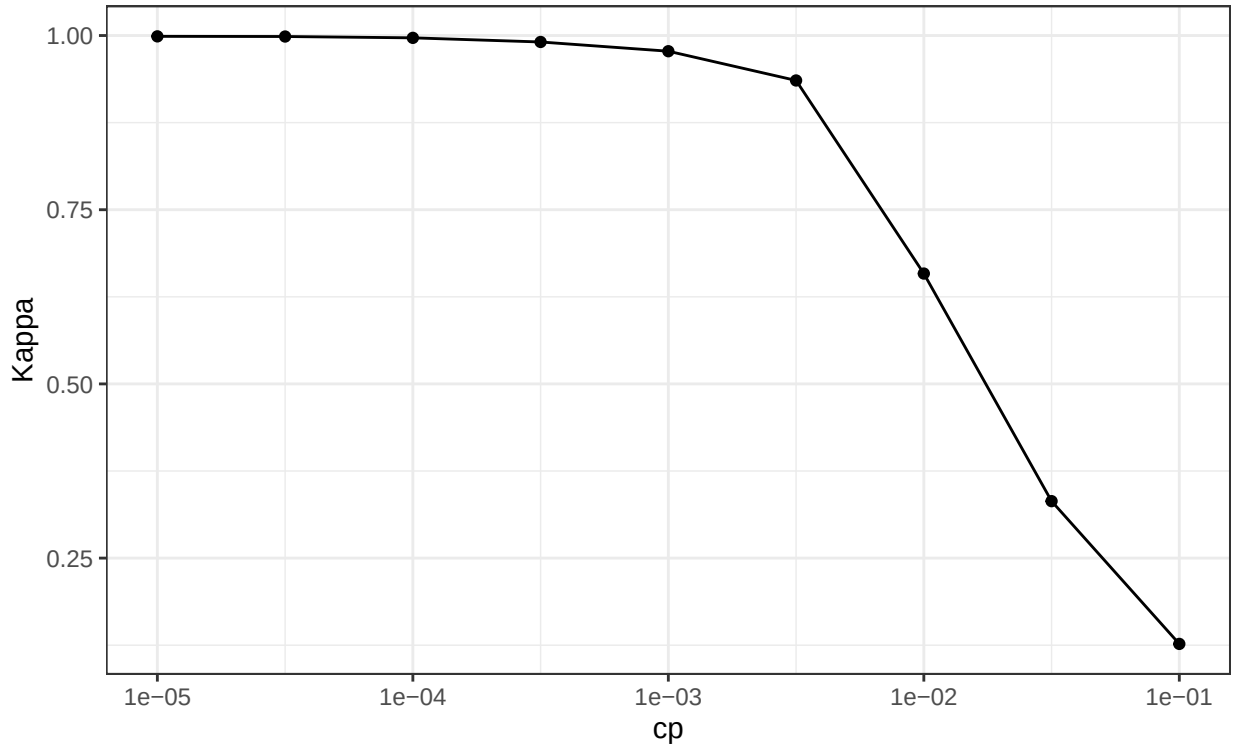

Figure 25: Performances du modèle CART (rpart) en fonction du paramètre de complexité

Ce modèle, pourtant très simple, donne déjà de très bons résultats globaux, avec ($\kappa_{max} = 0.999$ et une précision $R_{max} = 0.999$).

6.1.2 Forêts aléatoires

Le premier modèle de forêt aléatoire évalué dans cette partie est ranger, que nous avons déjà présenté précédemment. Les graphiques des performances en fonction des hyperparamètres laissent entrevoir d'excellentes caractéristiques sur une large plage d'hyperparamètres.

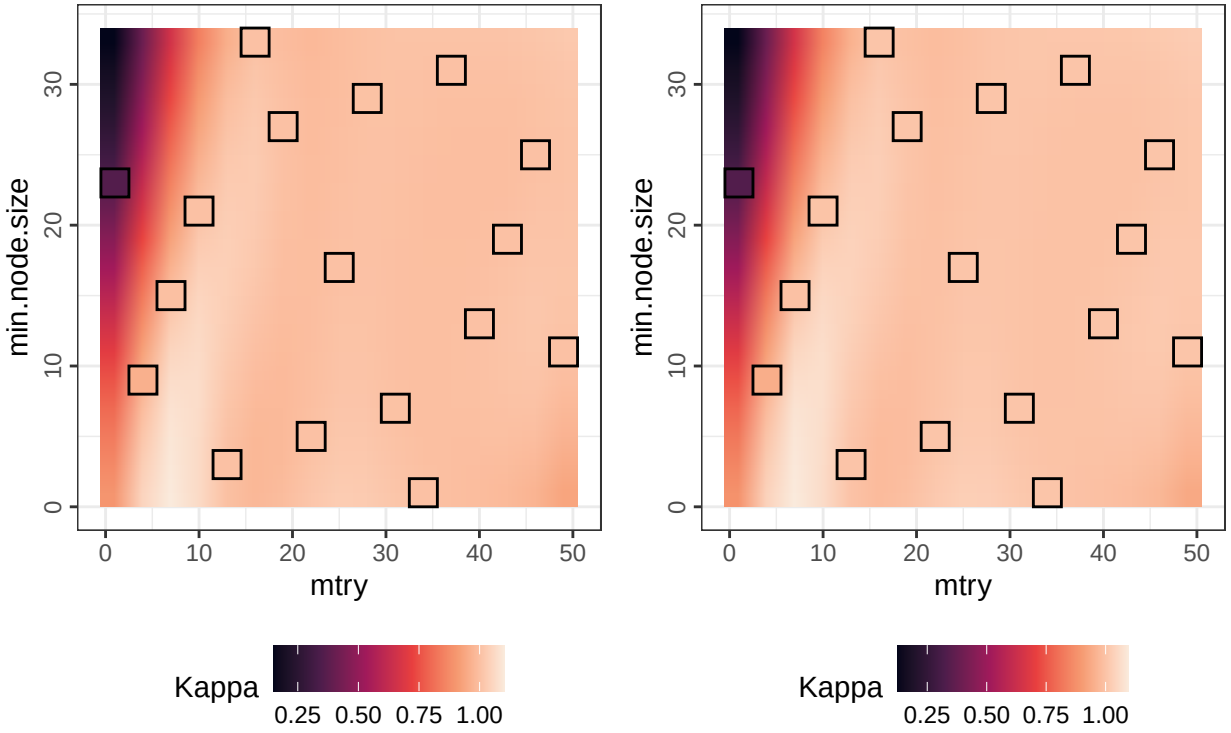

Figure 26: Performances du modèle Ranger, en fonction de ses 2 hyperparamètres (algorithme de scission : gini à gauche, extratrees à droite)

Après optimisation des hyperparamètres (*min.node.size* = 11, *mtry* = 49 et *splitrule* = extratrees), ce modèle a donné d'excellents résultats.

Table 4: Performances du modèle Ranger (hyperparamètres optimaux)

mtry	min.node.size	splitrule	Accuracy	Kappa
49	11	extratrees	0.99997	0.99997

Le dernier modèle de forêt aléatoire est Rborist.

Avec des paramètres optimaux (*predFixed* = 8 et *minNode* = 3), la performance est estimée à :

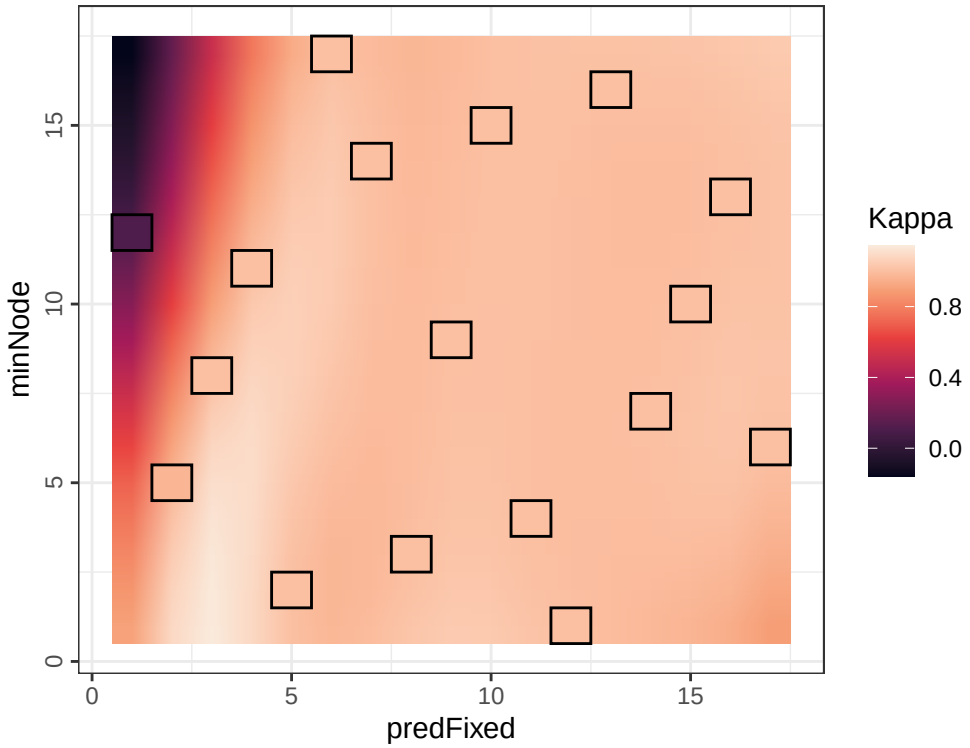

Figure 27: Performances du modèle Rborist en fonction de ses deux hyperparamètres (interpolation quadratique, points expérimentaux encadrés en noir)

Table 5: Performances du modèle Rborist (hyperparamètres optimaux)

predFixed	minNode	Accuracy	Kappa
8	3	0.99999	0.99999

Le modèle Rborist a donné des résultats similaires à ceux du modèle Ranger, avec d'excellentes performances.

6.1.3 Résultats

Les critères et le protocole de l'évaluation sont les mêmes que ceux évoqués précédemment.

L'évaluation finale du modèle ranger donne la matrice de confusion de la table 6.

La précision finale est égale à $R = 1$, avec un intervalle de confiance à 95% de [0.9996 ; 1]. La forêt aléatoire de type Ranger a donné d'excellents résultats, en un temps très raisonnable (1.74 min).

La forêt aléatoire de type Rborist a donné des résultats similaires, avec une précision finale égale à 1, avec un intervalle de confiance à 95% de [0.9996 ; 1]. Le modèle Rborist, donnant des résultats sensiblement identiques à Ranger, s'est avéré plutôt efficient sur le plan calculatoire (9.66 min).

Table 6: Matrice de confusion du modèle Ranger

	Amanita	Bolbitius	Bolete	Bracket_Fungi	Chanterelle	Cortinarius	Crepidotus	Ear_Pick	Entoloma	Hydnum	Ink_Cap	Jelly_Discs	Lepiota	Morel	Mushroom	Oyster_Mushroom	Paxillus	Pluteus	Russula	Saddle_Cup	Stropharia	Tricholoma	Wax_Gill
Amanita	381	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bolbitius	0	142	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bolete	0	0	666	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bracket_Fungi	0	0	0	335	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chanterelle	0	0	0	0	142	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cortinarius	0	0	0	0	0	524	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Crepidotus	0	0	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ear_Pick	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Entoloma	0	0	0	0	0	0	0	0	334	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hydnum	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0
Ink_Cap	0	0	0	0	0	0	0	0	0	0	621	0	0	0	0	0	0	0	0	0	0	0	0
Jelly_Discs	0	0	0	0	0	0	0	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0	0
Lepiota	0	0	0	0	0	0	0	0	0	0	0	0	143	0	0	0	0	0	0	0	0	0	0
Morel	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0
Mushroom	0	0	0	0	0	0	0	0	0	0	0	0	0	0	238	0	0	0	0	0	0	0	0
Oyster_Mushroom	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	0	0	0	0	0	0	0
Paxillus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	143	0	0	0	0	0	0
Pluteus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96	0	0	0	0	0
Russula	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1285	0	0	0	0
Saddle_Cup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	47	0	0	0
Stropharia	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	381	0	0
Tricholoma	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2046	0
Wax_Gill	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	382

Nous pouvons noter que le modèle ranger s'est ici avéré plus rapide que Rborist.

Table 7: Performances des modèles Ranger et Rborist (évaluation)

	Précision	Kappa	Durée (min)
Ranger	1	1	1.74
Rborist	1	1	9.66

6.2 Classification par espèce

A faire, en reprenant les recettes de la classification par familles. . .

7 Robustesse de la classification

Partie facultative, assez ambitieux, voir si cette partie est compatible (en temps) avec la création du lot de données, qui va être assez chronophage. . .

Evaluation de la robustesse, avec deux stratégies envisageables :

1. Robustesse "intrinsèque" : lot d'entraînement standard, lot d'évaluation avec variations. Robustesse des modèles seuls.

2. Robustesse "intégrée" : intégration de la robustesse dès le départ, avec lot d'entraînement intégrant des variations représentatives du lot d'évaluation.

7.1 Robustesse face aux déviations

Robustesse face à des données hors-normes, sans être extrêmes pour autant.

Facile à coder pour valeurs quantitatives (simplement augmenter la SD du générateur. . .)

DIFFICILE à coder pour les valeurs qualitatives (i.e. trouver des proximités, soit manuellement, soit via clustering / KNN)

Robustesse "intrinsèque" et "intégrée"

7.2 Robustesse face aux erreurs

Robustesse face à des données complètement aberrantes.

Facile à coder. Robustesse "intrinsèque" uniquement.

8 Références bibliographiques

1. Schlimmer J. Mushroom Data Set. University of California. 1987; Disponible sur: https://archive.ics.uci.edu/ml/datasets/Mushroom
2. Wagner D, Heider D, Hattab G. Mushroom data creation, curation, and simulation to support classification tasks. Scientific Reports [Internet]. avr 2021 [cité 10 déc 2022];11(1):8134. Disponible sur: https://www.nature.com/articles/s41598-021-87602-3
3. Wickham H. tidyverse: Easily Install and Load the Tidyverse [Internet]. 2022. Disponible sur: https://CRAN.R-project.org/package=tidyverse
4. Mersmann O. microbenchmark: Accurate Timing Functions [Internet]. 2021. Disponible sur: https://github.com/joshuaulrich/microbenchmark/
5. Ripley B. MASS: Support Functions and Datasets for Venables and Ripley's MASS [Internet]. 2023. Disponible sur: http://www.stats.ox.ac.uk/pub/MASS4/
6. Kuhn M. caret: Classification and Regression Training [Internet]. 2022. Disponible sur: https://github.com/topepo/caret/
7. Schloerke B, Cook D, Larmarange J, Briatte F, Marbach M, Thoen E, et al. GGally: Extension to ggplot2 [Internet]. 2021. Disponible sur: https://CRAN.R-project.org/package=GGally
8. Vakayil A, Joseph R. twinning: Data Twinning [Internet]. 2022. Disponible sur: https://CRAN.R-project.org/package=twinning
9. Therneau T, Atkinson B. rpart: Recursive Partitioning and Regression Trees [Internet]. 2022. Disponible sur: https://CRAN.R-project.org/package=rpart
10. Kuhn M, Quinlan R. C50: C5.0 Decision Trees and Rule-Based Models [Internet]. 2023. Disponible sur: https://topepo.github.io/C5.0/
11. Hothorn T, Hornik K, Strobl C, Zeileis A. party: A Laboratory for Recursive Partytioning [Internet]. 2022. Disponible sur: http://party.R-forge.R-project.org
12. Wright MN, Wager S, Probst P. ranger: A Fast Implementation of Random Forests [Internet]. 2022. Disponible sur: https://github.com/imbs-hl/ranger
13. Kursa MB. rFerns: Random Ferns Classifier [Internet]. 2021. Disponible sur: https://gitlab.com/mbq/rFerns
14. Seligman M. Rborist: Extensible, Parallelizable Implementation of the Random Forest Algorithm [Internet]. 2022. Disponible sur: https://CRAN.R-project.org/package=Rborist
15. Allaire J, Xie Y, McPherson J, Luraschi J, Ushey K, Atkins A, et al. rmarkdown: Dynamic Documents for R [Internet]. 2023. Disponible sur: https://CRAN.R-project.org/package=rmarkdown
16. Xie Y. knitr: A General-Purpose Package for Dynamic Report Generation in R [Internet]. 2022. Disponible sur: https://yihui.org/knitr/
17. Kassambara A. ggpubr: ggplot2 Based Publication Ready Plots [Internet]. 2023. Disponible sur: https://rpkgs.datanovia.com/ggpubr/

18. Franco J, Dupuy D, Roustant O, Kiener P, Damblin G, Iooss. B. DiceDesign: Designs of Computer Experiments [Internet]. 2021. Disponible sur: http://dice.emse.fr

19. Dupuy D, Helbert C. DiceEval: Construction and Evaluation of Metamodels [Internet]. 2022. Disponible sur: https://CRAN.R-project.org/package=DiceEval

20. Xie Y. bookdown: Authoring Books and Technical Documents with R Markdown [Internet]. 2023. Disponible sur: https://CRAN.R-project.org/package=bookdown

21. Courtecuisse R. Clé de détermination macroscopique des champignons supérieurs des régions du Nord de la France. Société mycologique du Nord de la France; 1986.

22. Courtecuisse R, Duhem B. Champignons de France et d'Europe. Delachaux et Niestlé; 2013. (Guides Delachaux).

23. Courtecuisse R, Moreau PA, Welti S. Initiation à la reconnaissance des champignons du Nord de la France - Clé pour la détermination des espèces les plus fréquentes. Département des Sciences Végétales et Fongiques, Faculté de Pharmacie de Lille; 2020.

24. Money NP. Insights on the mechanics of hyphal growth. Fungal Biology Reviews [Internet]. 2008 [cité 11 févr 2023];22(2):71-6. Disponible sur: https://www.sciencedirect.com/science/article/pii/S1749461308000195

25. Porter DL, Naleway SE. Hyphal systems and their effect on the mechanical properties of fungal sporocarps. Acta Biomaterialia [Internet]. juin 2022 [cité 11 févr 2023];145:272-82. Disponible sur: https://www.sciencedirect.com/science/article/pii/S1742706122002161

26. Johnson NL. Continuous univariate distributions, volume 2. 2nd ed. New York [etc: John Wiley & sons; 1995. (Wiley series in probability et mathematical statistics Applied probability et statistics; vol. 2).

27. R Core Team. R: A Language and Environment for Statistical Computing [Internet]. Vienna, Austria: R Foundation for Statistical Computing; 2021. Disponible sur: https://www.R-project.org/

28. Brownlee J. What is the Difference Between Test and Validation Datasets? [Internet]. MachineLearningMastery.com. 2017 [cité 14 févr 2023]. Disponible sur: https://machinelearningmastery.com/difference-test-validation-datasets/

29. Joseph VR. Optimal ratio for data splitting. Statistical Analysis and Data Mining: The ASA Data Science Journal [Internet]. 2022 [cité 15 févr 2023];15(4):531-8. Disponible sur: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11583

30. Mak S, Joseph VR. Support points. The Annals of Statistics [Internet]. déc 2018 [cité 15 févr 2023];46(6A):2562-92. Disponible sur: https://projecteuclid.org/journals/annals-of-statistics/volume-46/issue-6A/Support-points/10.1214/17-AOS1629.full

31. Joseph VR, Vakayil A. SPlit: An Optimal Method for Data Splitting. Technometrics [Internet]. avr 2022 [cité 15 févr 2023];64(2):166-76. Disponible sur: https://doi.org/10.1080/00401706.2021.1921037

32. Vakayil A, Joseph VR. Data Twinning. Statistical Analysis and Data Mining: The ASA Data Science Journal [Internet]. 2022 [cité 12 mars 2023];15(5):598-610. Disponible sur: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11574

33. Santiago J, Claeys-Bruno M, Sergent M. Construction of space-filling designs using WSP algorithm for high dimensional spaces. Chemometrics and Intelligent Laboratory Systems [Internet]. avr 2012 [cité 4 mars 2023];113:26-31. Disponible sur: https://linkinghub.elsevier.com/retrieve/pii/S0169743911001195

34. Youden WJ. Index for rating diagnostic tests. Cancer [Internet]. 1950 [cité 4 mars 2023];3(1):32-5. Disponible sur: https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-0142%281950%293%3A1%3C32%3A%3AAID-CNCR2820030106%3E3.0.CO%3B2-3

35. Rücker G, Schumacher M. Summary ROC curve based on a weighted Youden index for selecting an optimal cutpoint in meta-analysis of diagnostic accuracy. Statistics in Medicine [Internet]. 2010 [cité 4 mars 2023];29(30):3069-78. Disponible sur: http://onlinelibrary.wiley.com/doi/abs/10.1002/sim.3937

36. Cohen J. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement [Internet]. avr 1960 [cité 12 mars 2023];20(1):37-46. Disponible sur: https://doi.org/10.1177/001316446002000104

37. Landis JR, Koch GG. The Measurement of Observer Agreement for Categorical Data. Biometrics [Internet]. 1977 [cité 12 mars 2023];33(1):159-74. Disponible sur: http://www.jstor.org/stable/2529310

A Annexe : développement d'un algorithme de génération de lot synthétique

Plus de détails concrets sur la méthode utilisée en pratique, avec extraits de code...

Pour l'instant, l'algorithme est écrit et fonctionne, à partir des données primaires du Secondary Dataset de Dennis Wagner. C'est cet algorithme qui a servi à créer le lot de données de la classification multiclasse.

Pour avoir mes propres lots de données, il ne me reste plus qu'à avoir mes propres données primaires, c'est à dire entrer manuellement les caractéristiques clés de mes 400+ champignons, un par un...

La seule bibliothèque utilisée lors de la création du lot de données synthétique est le *tidyverse*[3], collection de bibliothèques spécialisées dans le domaine de la *data science* et notamment dédiées au traitement, au nettoyage et à la visualisation de données.

```
library(tidyverse)
```

Notre algorithme charge ensuite le fichier zip, incluant le fichier csv contenant les caractéristiques typiques des macromycètes (type de sporophore, dimensions maximales du stipe, du chapeau, type de lames, couleur de sporée, etc.), qui est lu et attribué à un *dataframe*. Les lignes commentées correspondent à l'utilisation d'un fichier identique hébergé à distance sur un dépôt GitHub.

```
fichier_data <- tempfile()
#URL <- "https://github.com/EKRihani/mushrooms/raw/master/MushroomDataset.zip"
#download.file(URL, fichier_data)
fichier_data <- "~/projects/champis/MushroomDataset.zip"
fichier_data <- unzip(fichier_data, "MushroomDataset/primary_data.csv")
data_champis <- read.csv(fichier_data,
                         header = TRUE,
                         sep = ";",
                         stringsAsFactors = TRUE)
```

etc etc.

B Annexe : outils d'analyse exploratoire des données (EDA)

Code EDA... A développer une fois le jeu de données primaires créé...

C Annexe : développement des algorithmes d'apprentissage machine

Nous détaillerons ici principalement les algorithmes utilisés pour le classifieur binaire. Les particularités d'intérêt des classifieurs multiples seront évoquées brièvement lors des développements de cette section.

C.1 Initialisation

Les bibliothèques utilisées lors des étapes d'apprentissage machine sont :

- tidyverse[3], collection de bibliothèques spécialisées dans le domaine de la *data science*,
- DiceDesign[18], bibliothèque spécialisée dans la création de plans d'expériences hypercubiques,
- DiceEval[19], bibliothèque spécialisée dans la modélisation des résultats de plans d'expériences hypercubiques,
- caret[6], collection d'outils dédiés à l'apprentissage machine.
- twinning[8], outils dédiés à la génération de jeux de données d'entraînement, optimisation, validation équilibrés.

```
library(tidyverse)
library(DiceDesign)
library(DiceEval)
library(caret)
library(twinning)
```

Le chargement des données s'effectue de la même façon que lors des sections précédentes. L'argument *stringsAsFactors* = *TRUE* revêt une importance particulière, car la classe *factor* est essentielle au bon fonctionnement des classifieurs. Dans le cadre d'une classification binaire, nous définissons arbitrairement, à l'aide de la fonction *relevel*, la classe *"toxique"* comme étant la valeur positive. Cette définition n'est pas nécessaire pour les classifieurs multiclasses.

```
fichier_data <- tempfile()
fichier_data <- "~/projects/champis/MushroomDataset.zip"
fichier_data <- unzip(fichier_data, "MushroomDataset/secondary_data.csv")
dataset <- read.csv(fichier_data,
                    header = TRUE,
                    sep = ";",
                    stringsAsFactors = TRUE)
dataset$class <- recode_factor(dataset$class, e = "comestible", p = "toxique")
# /!\ recode utilisé uniquement pour le Wagner !!!
dataset$class <- relevel(dataset$class, ref = "toxique")
```

C.2 Création des jeux d'entraînement, optimisation et évaluation

La création du jeu d'évaluation s'effectue en deux étapes. La première est la définition des rapports des dichotomies entre jeux d'entraînement et optimisation d'une part, et d'évaluation d'autre part. Cette définition implique l'évaluation du nombre d'individus, le calcul du nombre de coefficients p, puis du rapport de dichotomie $f = \sqrt(p) + 1$.[k]

```
BI_n_champis <- nrow(dataset)
BI_split_p <- sqrt(BI_n_champis)
BI_split_facteur <- round(sqrt(BI_split_p)+1)
```

La seconde partie consiste à effectuer la scission proprement dite. Cette scission implique la définition d'une liste d'index, de fraction 1 : f du nombre d'individus, qui servira via inclusion (jeu d'évaluation) ou exclusion (jeu d'apprentissage et d'évaluation) booléennes des lignes correspondantes, à de constituer chaque jeu de données. La fonction *set.seed* assure la reproductibilité.

```
set.seed(7)
index1 <- twin(data = dataset, r = BI_split_facteur)
BI_lot_appr_opti <- dataset[-index1,]
BI_lot_evaluation <- dataset[index1,]
```

Les lots ainsi obtenus seront ensuite utilisés pour l'entraînement, l'optimisation et l'évaluation finale des performances des modèles.

C.3 Entraînement et optimisation des modèles

Cette partie ne prétend pas à l'exhaustivité, elle se limitera à la présentation de l'entraînement, l'optimisation et la génération de graphiques pour deux modèles : un modèle CART (rpart) et un modèle RF (Rborist). Un certain nombre de tâches telles que l'entraînement du modèle ou la génération de graphiques sont en réalité attribuées à des fonctions créées *adhoc* dans le but de clarifier l'organisation du code de l'algorithme, car elles sont effectuées à de nombreuses reprises. Nous décrirons ici le code source sans faire appel à ces fonctions.

C.3.1 Arbre de classification et régression

La première étape est de définir l'indice J de Youden,[l] et les pondérations respectives de la sensibilité et de la spécificité. Cette définition n'est pas nécessaire pour les classifieurs multiclasse, le kappa (κ) et l'indice de Rand (R) étant des métriques évaluées nativement par la librairie *caret*.

```
BI_w <- 10
BI_RatioSens <- 2*BI_w/(BI_w+1)
BI_RatioSpec <- 2*(1-BI_w/(BI_w+1))
```

[k] cf. section 4.1, page 29.
[l] cf. section 4.5, page 33

La seconde définition à préciser est celle de l'espace expérimental des hyperparamètres. En l'espèce le seul hyperparamètre du modèle rpart est la variable *cp*.

```
BI_grid_rpart_cp <- data.frame(cp = 10^seq(from = -5, to = -1, by = .5))
```

L'étape suivante est de définir les paramètres d'entraînement et d'évaluation des performances du modèle en vue de son optimisation. La fonction *trainControl* permet ici de préciser les principaux paramètres régissant cette étape :

- *classProbs*, *[TRUE indispensable au fonctionnement...]*
- *summaryFunction*, afin d'indiquer que les métriques de performance à utiliser sont celles d'un classifieur binaire (l'argument *multiClassSummary* sera utilisé pour un classifieur multiclasse)
- *method*, afin de préciser la méthode de construction des jeux d'entraînement et d'optimisation, ici validation croisée (*CV: cross-validation*)
- *number*, afin d'indiquer le nombre de blocs de la validation croisée, calculé précédemment.

Ici aussi, la fonction *set.seed* assure la reproductibilité du processus.

```
set.seed(1)
tr_ctrl <- trainControl(classProbs = TRUE,
                        summaryFunction = twoClassSummary,
                        method = "cv",
                        number = BI_split_facteur)
```

L'entraînement du modèle peut avoir lieu. En l'espèce, le modèle mathématique retenu est l'attribution d'une prédiction sur *class* en fonction de toutes les autres variables (*class* $\sim$.). Les arguments *data*, *trControl*, *tuneGrid* font appel aux éléments décrits dans les paragraphes qui précèdent.

```
BI_fit_rpart_cp <- train(class ~ .,
                         method = "rpart",
                         data = BI_lot_appr_opti,
                         trControl = tr_ctrl,
                         tuneGrid  = BI_grid_rpart_cp)
```

L'objet résultant est d'une structure relativement complexe. Notre algorithme peut notamment en extraire les résultats relatifs aux performances du modèle, et y adjoindre le calcul du J de Youden pondéré J_w. Dans le cadre des classifieurs multiclasses, ce calcul est inutile, l'objet généré à l'étape précédente contenant déjà les indicateurs de performance que nous utilisons : kappa (κ) et indice de Rand (*R*, *accuracy*).

```
BI_fit_rpart_cp_resultats <- BI_fit_rpart_cp$results %>%
   mutate(Jw = Sens*BI_RatioSens + Spec*BI_RatioSpec - 1)
```

Table 8: Tableau des résultats de l'entraînement de rpart

cp	ROC	Sens	Spec	ROCSD	SensSD	SpecSD	Jw
0.0000100	0.9992475	0.9977113	0.9974202	0.0005176	0.0014069	0.0014071	0.9953696
0.0000316	0.9992344	0.9975859	0.9974593	0.0005374	0.0014608	0.0012481	0.9951487
0.0001000	0.9992092	0.9975545	0.9973029	0.0005223	0.0014356	0.0012967	0.9950632
0.0003162	0.9987400	0.9956733	0.9963646	0.0007393	0.0026099	0.0012678	0.9914724
0.0010000	0.9972132	0.9904371	0.9894068	0.0007871	0.0031615	0.0027640	0.9806869
0.0031623	0.9848444	0.9641001	0.9557501	0.0042545	0.0104682	0.0095868	0.9266819
0.0100000	0.8983138	0.9286069	0.7720671	0.0141701	0.0192717	0.0380337	0.8287520
0.0316228	0.7011303	0.5726145	0.8032593	0.0142681	0.0320282	0.0150998	0.1871645
0.1000000	0.6176520	0.5355555	0.6997485	0.0111304	0.0146717	0.0167098	0.1009643

L'objet de type *dataframe* ainsi créé peut être appelé afin d'en extraire des résultats d'intérêt ou d'en inclure le tableau dans le rapport :

L'algorithme génère également un graphique synthétisant les performances du modèle (sensibilité, spécificité, J_w, κ, R ou autre indicateur d'intérêt) en fonction de son hyperparamètre :

- *ggplot* est la fonction de génération du graphique, et permet d'appeler l'objet servant à générer le graphique, ainsi que certains paramètres complémentaires via *aes*. Ici, la variable servant d'abscisse.
- *geom_point* permet de tracer le nuage de points. Ici encore, *aes* permet de préciser, pour chaque nuage de points, la variable d'ordonnée (*Sens*, *Spec* ou *Jw*), ainsi que la légende associée à la couleur des points (*Sensibilité*, *Spécificité* ou *Jw*).
- *geom_line* permet de tracer les lignes correspondant au nuage de points.
- *labs* permet de légender correctement l'attribut *color* de notre légende.
- *ylab* permet de définir la légende l'axe des ordonnées. Ici, de la supprimer, car nous avons trois variables différentes en ordonnées.
- *scale_x_log10* nous permet ici de définir un axe logarithmique décimal en abscisse.
- *theme_bw* attribue le thème(couleur de fond, d'axes, grilles) de type *bw* (*black and white*) à notre graphique.

```
BI_fit_rpart_cp_graphe <- ggplot(data = BI_fit_rpart_cp_resultats, aes(x = cp)) +
   geom_point(aes(y = Sens, color = "Sensibilité")) +
   geom_line(aes(y = Sens, color = "Sensibilité")) +
   geom_point(aes(y = Spec, color = "Spécificité")) +
   geom_line(aes(y = Spec, color = "Spécificité")) +
   geom_point(aes(y = Jw, color = "Jw")) +
   geom_line(aes(y = Jw, color = "Jw")) +
   labs(color = "Performance") +
   ylab(NULL) +
   scale_x_log10() +
   theme_bw()
```

Le graphique ainsi généré peut être intégré dans notre rapport :

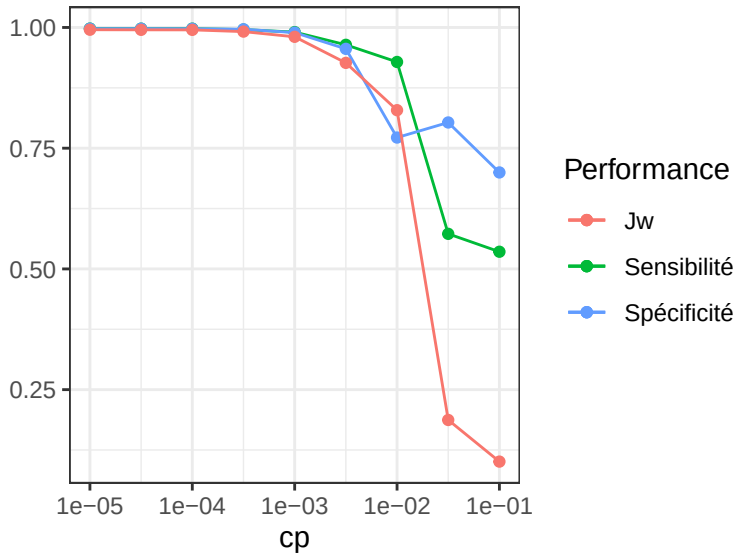

Figure 28: Graphique des performances de rpart

C.3.2 Rborist

La première étape est, comme précédemment, de définir l'espace expérimental. Ici, s'agissant d'un modèle à plusieurs hyperparamètres, l'algorithme utilisera un plan d'expériences basé sur les hypercubes latins. La fonction *nolhDesign* permet de créer un hypercube latin quasi-orthogonal (NOLH : *Near Orthogonal Latin Hypercube*), ici paramétré avec 2 dimensions, dans l'espace $[0;1]^2$. Le plan d'expérience en est extrait, puis inséré dans un objet de type *dataframe*, avec colonnes nommées d'après nos variables réduites X_1 et X_2.

```
BI_LHS <- nolhDesign(dimension = 2, range = c(0, 1))$design
BI_LHS <- data.frame(BI_LHS)
colnames(BI_LHS) <- c("X1", "X2")
```

L'hypercube latin des hyperparamètres (*predFixed* et *minNode*) est généré à partir de l'hypercube latin des paramètres réduits. Ces hyperparamètres sont des valeurs entières. L'hypercube latin quasi-orthogonal de dimension 2 possédant 17 expériences équitablement réparties dans l'espace $[0;1]^2$, il apparaît souhaitable, pour des raisons d'homogénéité dans l'espace expérimental, que $F_n = k \times X_n$ avec k multiple de 16. En pratique, nous n'avons jamais rencontré d'erreurs d'arrondi lors de cette étape mais l'usage de la fonction *round* constitue une précaution supplémentaire garantissant que le produit sera bien un entier.

```
BI_grid_Rborist <- data.frame(BI_LHS) %>%
   mutate(predFixed = round(1+X1*16,0)) %>%
   mutate(minNode = round(1+X2*16,0))
```

Table 9: Plan d'expériences d'entraînement et optimisation du modèle Rborist

X1	X2	predFixed	minNode
0.3125	1.0000	6	17
0.0625	0.2500	2	5
0.1250	0.4375	3	8
0.1875	0.6250	4	11
0.7500	0.9375	13	16
1.0000	0.3125	17	6
0.6250	0.1875	11	4
0.5625	0.8750	10	15
0.5000	0.5000	9	9
0.6875	0.0000	12	1
0.9375	0.7500	16	13
0.8750	0.5625	15	10
0.8125	0.3750	14	7
0.2500	0.0625	5	2
0.0000	0.6875	1	12
0.3750	0.8125	7	14
0.4375	0.1250	8	3

L'entraînement du modèle se déroule de la même façon que pour le modèle rpart.[m]

```
set.seed(1)
tr_ctrl <- trainControl(classProbs = TRUE,
                        summaryFunction = twoClassSummary,
                        method = "cv",
                        number = BI_split_facteur)
BI_fit_Rborist <- train(class ~ .,
                        method = "Rborist",
                        data = BI_lot_appr_opti,
                        trControl = tr_ctrl,
                        tuneGrid  = BI_grid_Rborist[c('predFixed', 'minNode')])
```

L'algorithme extrait les résultats relatifs aux performances du modèle et y adjoint le calcul de J_w (ou κ pour les classifieurs multiclasses), comme précédemment. L'objet obtenu ne contenant que les facteurs expérimentaux (non réduits) des hyperparamètres, il convient d'y adjoindre les facteurs réduits. La table *BI_grid_Rborist* générée précédemment contient toutes les informations qui permettent, via une jonction, de lier facteurs réduits et facteurs expérimentaux.

[m] cf. section C.3.1, page 65.

```
BI_fit_Rborist_resultats <- BI_fit_Rborist$results %>%
   mutate(Jw = Sens*BI_RatioSens + Spec*BI_RatioSpec - 1) %>%
   left_join(x = .,
             y = BI_grid_Rborist,
             by = c("predFixed", "minNode"))
```

Nous chargeons ensuite notre algorithme de calculer le modèle quadratique avec interactions permettant d'évaluer, à partir des résultats obtenus suite à l'entraînement, la réponse J_w en fonction des X_1 et X_2, suivant la formule :

$$Y = b_0 + b_1.X_1 + b_2.X_2 + b_{12}.X_1.X_2 + b_{11}.X_1^2 + b_{22}.X_2^2$$

```
BI_mod_Rborist_jw <-  modelFit(X = BI_fit_Rborist_resultats[,c("X1", "X2")],
                               Y = BI_fit_Rborist_resultats$Jw,
                               type="Kriging",
                               formula= Y ~ X1 + X2 + X1:X2 + I(X1^2) + I(X2^2))
```

Ces résultats permettent notamment de modéliser les performances sur la totalité de l'espace expérimental des hyperparamètres. A cette fin, l'algorithme est chargé de générer l'ensemble des couples de valeurs (X_1, X_2) possibles, à l'aide de la fonction *expand.grid*, avant de calculer la valeur J_w correspondante à chaque couple de points.

```
CodBI_pred_Rborist <- expand.grid(CodBI_fit_Rborist_resultats[,c("X1","X2")]) %>%
   mutate(Jw = modelPredict(CodBI_mod_Rborist_jw, .[,c("X1","X2")]))
```

L'ensemble des données expérimentales et modélisées obtenues permettent de générer un graphique bidimensionnel des performances en fonction des hyperparamètres. Pour des raisons didactiques, nous séparerons ici le graphique résultant de l'expérimentation de celui résultant de la modélisation quadratique.

La génération du graphique reprend des principes similaires à ceux présentés précédemment pour le graphique des performances de rpart, notamment au niveau des arguments utilisés dans *aes*. Les seules fonctions appelant à commentaires sont l'utilisation de *geom_raster* pour la modélisation, à laquelle nous superposons le graphique généré via *geom_tile* pour les points expérimentaux. L'utilisation de la gamme de couleurs proposée par *viridis* permet une visualisation plus aisée des résultats obtenus.

```
BI_pred_Rborist %>% ggplot() +
   geom_raster(data = BI_pred_Rborist,
               aes(x = X1, y = X2, fill = Jw), interpolate = TRUE) +
   geom_tile(data = BI_fit_Rborist_resultats,
             aes(x = X1, y = X2, fill = Jw), color = 'black', linewidth =.5) +
   scale_fill_viridis_c(option = "D", direction = 1) +
```

```
  theme(axis.text.y = element_text(angle=90, vjust=.5, hjust=.5)) +
  theme_bw() +
  theme(legend.position='bottom')
```

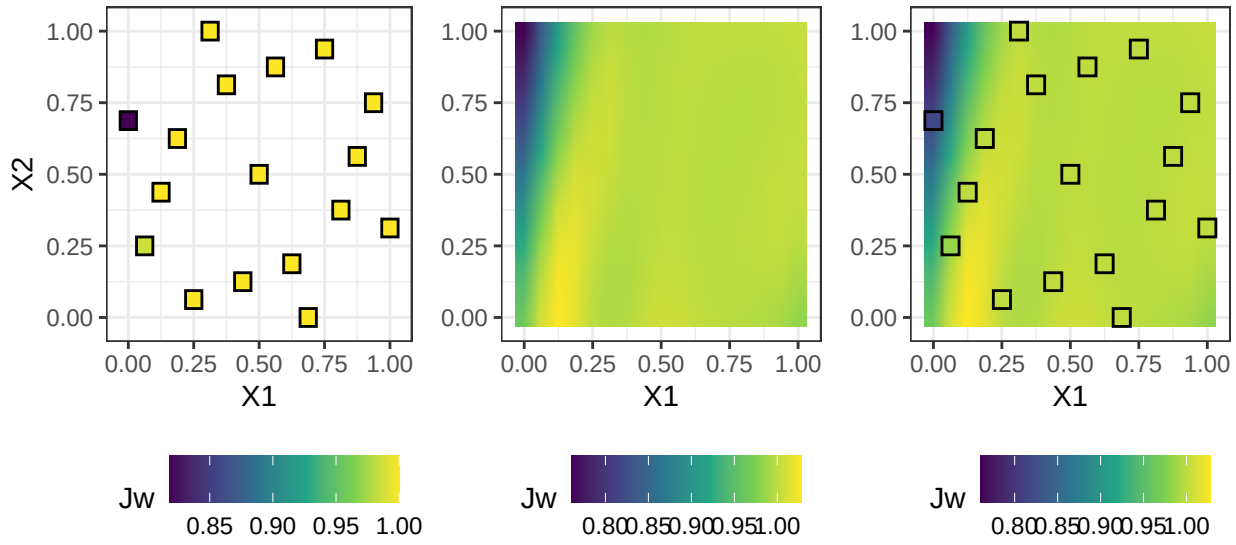

Figure 29: Points expérimentaux (à gauche), modélisation (au centre) et superposition (à droite) des résultats obtenus avec Rborist

L'étape suivante est d'exploiter ce modèle quadratique pour évaluer les hyperparamètres permettant de maximiser J_w. A cet effet, une table de l'ensemble des points de l'espace expérimental des facteurs réduits est générée, puis la fonction quadratique évaluée précédemment est appliquée, afin d'obtenir une prédiction approximative de J_w. Les hyperparamètres sont également calculés, afin de pouvoir être appliqués par la suite.

```
BI_modelquad_Rborist <- expand.grid(X1 = seq(from = 0, to = 1, length.out = 17),
                                    X2 = seq(from = 0, to = 1, length.out = 17)) %>%
  mutate(Jw = BI_mod_Rborist_jw$model@trend.coef[1] +
            BI_mod_Rborist_jw$model@trend.coef[2]*X1 +
            BI_mod_Rborist_jw$model@trend.coef[3]*X2 +
            BI_mod_Rborist_jw$model@trend.coef[4]*X1^2 +
            BI_mod_Rborist_jw$model@trend.coef[5]*X2^2 +
            BI_mod_Rborist_jw$model@trend.coef[6]*X1*X2) %>%
  mutate(predFixed = round(1+X1*16,0)) %>%
  mutate(minNode = round(1+X2*16,0))
```

Le J_w théorique maximal est ensuite évalué, ainsi que les hyperparamètres qui y sont associés.

```
BI_modelquad_Rborist_top <- BI_modelquad_Rborist[which.max(BI_modelquad_Rborist$Jw),
                                                 c("predFixed", "minNode")]
```

Suite du code : A TESTER !

Il est ensuite possible de relancer un entraînement, comme précédemment, avec les paramètres optimisés, et en extraire les indicateurs de performances (J_w).

```
set.seed(1)
BI_fit_Rborist_best <- train(class ~ .,
                  method = "Rborist",
                  data = BI_lot_appr_opti,
                  trControl = tr_ctrl,
                  tuneGrid  = BI_modelquad_Rborist_top[c('predFixed', 'minNode')])

BI_fit_Rborist_best_resultats <- BI_fit_Rborist_best$results %>%
      mutate(Jw = Sens*CodBI_RatioSens + Spec*CodBI_RatioSpec - 1)
```

C.4 Évaluation des performances des modèles

L'étape finale est celle de l'évaluation des performances du modèle, ici présentée pour Rborist. Cette évaluation commence par l'extraction des valeurs réelles de comestibilité – c'est à dire des réponses attendues de la part de notre modèle – à partir du jeu de données d'évaluation, ainsi que leur conversion en valeurs booléennes, à fins de comparaison avec les valeurs qui seront prédites. Cette étape n'est évidemment pas nécessaire lors d'une classification multiclasse.

```
BI_evaluation <- BI_lot_evaluation %>%
   mutate(reference = as.factor(case_when(class == "toxique" ~ TRUE,
                                          class == "comestible" ~ FALSE)))
```

Les performances du modèle peuvent être évaluées, en termes d'efficience calculatoire, par son temps d'éxécution. Un moyen de mesurer le temps d'éxécution de n'importe quelle portion de code est de mesurer l'heure de début et de fin d'éxécution du code à l'aide de *Sys.time*, puis d'en mesurer la différence via la fonction *difftime*.

C.4.0.1 Le code éxécuté correspond ici à l'entraînement du modèle (cf. *supra*) et à la prédiction sur le lot d'évaluation, enregistré dans un objet dédié.

```
chrono_debut <- Sys.time()
BI_fit_Rborist_final <- train(class ~ .,
                   method = 'Rborist',
                   data = BI_lot_appr_opti,
                   trControl = tr_ctrl,
                   tuneGrid  = BI_modelquad_Rborist_top[c('predFixed', 'minNode')])
BI_pred_Rborist_final <- predict(object = BI_fit_Rborist_final,
                                 newdata = BI_lot_evaluation)
chrono_fin <- Sys.time()
CodBI_temps_Rborist <- difftime(chrono_fin, chrono_debut) %>%
```

```
   as.numeric %>%
   round(.,2)
```

L'objet correspondant aux prédictions est ensuite comparé aux valeurs références que notre modèle devait prédire, ce qui permet notamment d'obtenir la matrice de confusion associée aux prédictions.

```
BI_CM_Rborist_final <- confusionMatrix(data = BI_pred_Rborist_final,
                                       reference = BI_lot_evaluation$class)
BI_CM_Rborist_final$table
```

Nous pouvons également extraire de cette comparaison des indicateurs de performances : sensibilité, spécificité, J_w dans le cas d'une classification binaire, kappa et indice de Rand pour la classification multiclasse, ainsi que le temps de calcul.

```
BI_resultats_Rborist <- BI_CM_Rborist_final$byClass %>%
   t(.) %>%
   as.data.frame(.) %>%
   select(c(Sensitivity, Specificity)) %>%
   mutate(Jw = Sensitivity*BI_RatioSens + Specificity*BI_RatioSpec - 1) %>%
   mutate(temps = BI_temps_Rborist)
```

Enfin, les principaux objets volumineux qui n'ont pas vocation à être exploités par la suite – c'est-à-dire les jeux de données, ainsi que les objets issus des fonctions *train* – sont retirés de l'environnement, qui est ensuite sauvegardé. Cette sauvegarde contient les graphiques, tableaux et valeurs d'intérêt, à fins d'insertion automatisée dans le fichier Rmarkdown qui constitue le corps de texte de cette thèse.

```
rm(dataset, BI_evaluation, BI_lot_appr_opti, BI_lot_evaluation,
   BI_fit_rpart_cp, BI_fit_Rborist, BI_fit_Rborist_best, BI_fit_Rborist_final)

save.image(file = "EKR-Champis-CodeSourceBi.RData")
```

Université de Lille

FACULTE DE PHARMACIE DE LILLE

DIPLOME D'ETAT DE DOCTEUR EN PHARMACIE

Année Universitaire 2022/2023

Nom : RIHANI
Prénom : Emir Kaïs

Titre de la thèse : Application de modèles d'intelligence artificielle à la classification des macromycètes

Mots-clés : intelligence artificielle, apprentissage machine, *machine learning*, classification, mycologie

---

Résumé : L'IA c'est génial !

Membres du jury :

Président : Nom, Prenom, titre et lieu de fonction

Assesseur(s) : Nom1, Prenom1, titre et lieu de fonction
Nom2, Prenom2, titre et lieu de fonction
Nom3, Prenom3, titre et lieu de fonction

Membre(s) extérieur(s) : Nom1, Prenom1, titre et lieu de fonction
Nom2, Prenom2, titre et lieu de fonction
Nom3, Prenom3, titre et lieu de fonction