

RAPPORT DE STAGE

1^{er} Année BTS Analyse
Biotechnologique

Réalisé par: LOUIS DEBRAY
Maître de stage: PATRICE GARCIA

Du 1 juin au 31 août 2020



Forêt Cellulose Bois-construction Ameublement
ALLEE DE BOUTANT 33000 BORDEAUX

Objectif : Développement d'un outil de reconnaissance d'essences de bois en utilisant la spectroscopie infrarouge proche.

Mots-clefs : Prétraitement des données - Classification - Model de machine learning et deep learning - Réseau de neurone - K-nearest-neighbors

CONTENU

REMERCIEMENTS	4
INTRODUCTION :	6
I-L'ENTREPRISE FCBA	7
II-MON ACTIVITE AU FCBA.....	7
III-LA RECONNAISSANCE D'ESSENCE DE BOIS	8
1- Mise en place de la problématique	8
A- Le contexte.....	8
B- Programme de reconnaissance d'essences de bois par prise d'images	9
C- Ma contribution à l'axe de recherche :	9
a- Utilisation du SPIR (Spectre Infra-Rouge)	9
b- Définition d'un objectif	10
D- Intelligence artificiel: Machine Learning / Deep Learning	11
2- Les données.....	11
A- Caractéristiques	11
B- X et y trainset et X et y testset	13
3- Les modèles.....	14
A- Choix du modèle	14
B- Deep Learning / Réseaux de neurones.....	15
a- Fonctionnement d'un réseau de neurones	15
C- Machine Learning / Algorithme K-Nearest-Neighbors	18
a- Fonctionnement	18
4- Etudes des signaux.....	20
A- Comprendre le signal issu d'un spectromètre infra-rouge avec la transformé de fourrier	20
B- Spectroscopie infra-rouge proche (NIR).....	22
C- Les Prétraitements des spectres dans l'infra-rouge proche :.....	23
C-1 Correction du phénomène de diffusion	23
a- La ligne de base.....	24
b- La Dérivation	25
c- Le lissage de Savitsky-Golay (SG).....	26
d- La Normalisation (squatter-correction) :.....	27
e- Application des prétraitements aux spectres IR	28
f- Choix des gammes spectrales	29
C-2 Sélection de variables	30
a- Utilisation du SelectKBest	31
5- Optimisation et Résultats	33
A- Les hypers-paramètres.....	33

B-	Le validation set.....	34
C-	Randomized-Search-CV.....	34
D-	Recherche du réseau de neurone optimal	35
a-	Structure et hyperparamètres optimaux.....	35
b-	Optimisation du nombre d'époques.....	35
c-	Les Résultats.....	36
E-	Recherche du K-Nearest-Neighbors optimal.....	36
a-	Les hyperparamètres optimaux	36
b-	Les Résultats.....	37
IV-	CONCLUSION :	38

TABLE DES FIGURES

Figure 1. Matrice de confusion à deux classes	10
Figure 2. Représentation d'une image numérique.....	12
Figure 3. Représentation d'un signal numérique	12
Figure 4. Représentation des faces du bois.....	13
Figure 5. Tableau du Dataset composé des spectres d'Epicea, et de Sapin	13
Figure 6. Procédure du choix du modèle de Machine Learning.....	14
Figure 7. Schéma d'une propagation avant	16
Figure 8. Fonction sigmoïde.....	16
Figure 9. Différentes étapes de la rétropropagation	17
Figure 10. Répartition des points de deux classes suivant deux variables (Y-Axe et X-Axe).	18
Figure 11. Les Etapes de bases du KNN.....	19
Figure 12. Démonstration de l'influence du choix de K.....	19
Figure 13. Signal = somme de signaux.	20
Figure 14. Intégration du signal multiplié par plusieurs signaux sinusoïdaux.	20
Figure 15. Transformée de fourrier d'un signal périodique	21
Figure 16. Transformée de fourrier d'un signal non périodique	21
Figure 17. Transformée de fourrier d'un signal non périodique	22
Figure 18. Répartition des points de deux classes suivant deux variables (Y-Axe et X-Axe).	23
Figure 19. Spectre = somme spectre pur et ligne de base	24
Figure 20. Diffusion et réflexion sur le fond d'une onde.	24
Figure 22. Spectres simulés, perturbés par des lignes de bases.....	25
Figure 21. Simulation de deux bandes d'absorptions	25
Figure 23. Dérivation seconde des deux spectres	26
Figure 24. Lissage par Savitzky-Golay.....	26
Figure 25. Prétraitement appliqué à un spectre d'Epicéa	27
Figure 26. Différents prétraitements appliqués aux spectres infra-rouges proche	28
Figure 27. Pic d'absorption des vibrations de la molécule d'eau avec méthode LB/SNV	29
Figure 29. Pic d'absorption des vibrations de la molécule d'eau avec méthode SG2/SNV	29
Figure 28. Pic d'absorption des vibrations de la molécule d'eau avec méthode SG1/SNV	29
Figure 30. Courbe d'entraînement sans sélection de variables : classement des espèces aléatoirement	30
Figure 31. Exemple de Cross-validation de 25 % sur le Trainset.....	34
Figure 32. Courbe d'entraînement et de Validation pour illustrer l'overfitting	35

REMERCIEMENTS

Je tiens à remercier dans un premier temps Mr Patrice Garcia mon maître de stage, pour sa sympathie, son dynamisme, et la confiance qu'il a placé en moi tout au long du stage.

Grace à lui et ses qualités de pédagogue j'ai pu comprendre rapidement les premières bases théoriques me permettant de commencer mon programme.

Je suis également reconnaissant envers les ingénieurs et techniciens du laboratoire Mécanique du FCBA, pour m'avoir intégré dans leur équipe et m'avoir permis de garder le cap durant la conception du programme.

Ce stage fut pour moi très enrichissant et gratifiant dans toutes les tâches que j'ai réalisées et les échanges que j'ai pu avoir. J'en ressort grandit avec un bagage technique passionnant.

INTRODUCTION

Je devais réaliser mon stage de 2ème année de BTS analyse biotechnologique au Québec au Canada dans un laboratoire de lutte biologique, mais celui-ci a été annulé à cause de la crise sanitaire du covid-19.

J'ai finalement eu la possibilité de réaliser un stage dans l'entreprise FCBA (Foret Cellulose Bois-construction Ameublement) situé à Bordeaux. Ce stage, dont la plus grosse partie faisait appel aux sciences de l'ingénieur, allait bien au-delà de ce j'ai appris dans ma formation initiale (bien que dans BTS analyse-biotechnologie il y a aussi technologie). Cependant celui-ci m'a permis d'explorer le monde de l'ingénierie du bois, domaine qui m'intéressait tout particulièrement et dans lequel j'aimerais travailler dans le futur.

Durant cette période de 10 semaines, j'ai eu l'occasion de découvrir le monde de l'entreprise et plus particulièrement celui d'un laboratoire mécanique.

Au-delà d'enrichir mes connaissances dans des domaines variés (mathématiques, physiques, et informatique) cette période de stage m'a permis de découvrir le monde de la recherche et ses différents rouages en alliant, recherches de différentes solutions, innovation constante et tests pour l'amélioration du programme.

L'élaboration de ce rapport de stage a pour but de décrire mes contributions à l'axe de recherche en cours de réflexion. Il a pour principales sources les documents réalisés sur l'étude des signaux par les scientifiques de FCBA, mes recherches sur internet et sur différents sites spécialisés en open source, ainsi que les entretiens que j'ai eu l'occasion d'avoir avec les experts dans les domaines en question. Il s'agit ici d'un axe de recherche qui aurait mérité une durée de 6 mois pour atteindre des résultats complets. Vous trouverez donc ici un ensemble de voies de recherches qui ne sont pas à prendre comme définitives, ainsi que d'autres voies qui restent à envisager.

I-L'ENTREPRISE FCBA

Créé en 1952 à la demande des professionnels de la filière bois, FCBA (pour Forêt Cellulose Bois-construction Ameublement) est un Centre Technique Industriel de 340 collaborateurs. Son chiffre d'affaire est de 30 millions d'euros en 2018.

L'activité du FCBA couvre l'ensemble des domaines des entreprises de la forêt, du bois, de la cellulose, de la construction et de l'ameublement et s'appuie sur différents métiers : recherches, innovations, normalisation, diffusion de l'information, promotion de la filière, consultance, appui technique, certification, essais, formation ...

L'objectif étant de renforcer la compétitivité des entreprises françaises, face à la mondialisation et à la concurrence inter-matériaux.

Cet établissement est placé à la fois sous la tutelle du Ministère de l'Agriculture (chargé de la forêt) et du Ministère de l'Industrie. FCBA a un Conseil d'Administration constitué de représentants des diverses professions, des Ministères de l'Agriculture et de l'Industrie et de représentants des personnels techniques des entreprises.

FCBA assure une présence sur l'ensemble du territoire grâce à ses délégations territoriales (Limoges, Dijon, Grenoble, Cestas). J'ai réalisé mon stage dans le pôle « Laboratoire bois » de FCBA. Ce pôle est chargé de réaliser des essais sur les produits, matériaux et composants de la construction afin de caractériser l'aptitude à l'emploi et les performances des matériaux. Il intervient dans la validation du respect des exigences réglementaires et assure un appui à l'activité de certification par l'évaluation initiale des produits à certifier et leur suivi dans le temps. Il se divise en 4 laboratoires : « Physique », « Chimie », « Biologie » et « Mécanique ». C'est dans ce dernier que j'ai effectué mon stage.

II-MON ACTIVITE AU FCBA

Sous la tutelle de Mr Garcia le directeur du laboratoire mécanique de FCBA, j'ai eu le privilège de participer au projet de développement d'un outil de reconnaissance d'essence de bois. Ma contribution au projet a consisté à mettre en place un modèle permettant la différenciation des résineux Epicéa et Sapin par analyse de leur spectre dans le domaine de l'infra-rouge proche.

Mon activité au sein de l'entreprise pouvait être assimilé au travail d'un chercheur dans différents domaines :

Premièrement, il m'a fallu réaliser une étude sur les signaux issus d'un spectromètre infra-rouge. En passant par la compréhension des bases mathématiques se cachant derrière ces signaux et les traitements réalisés sur ces derniers.

Dans un deuxième temps, il m'a fallu partir à la recherche des concepts théoriques me permettant de comprendre la structure et le mécanisme du programme d'un réseau de neurones.

Et finalement entrant dans le vif du sujet, j'ai dû apprendre par moi-même à maîtriser le langage python permettant la réalisation de ce programme tout en échangeant avec d'autres professionnels afin de garder le cap.

Ces travaux de recherche rassemblent à mon goût deux facettes qui me plaisent particulièrement. Ils allient à la fois « un esprit de doute » (à travers la lecture de différents sites/ articles ou le visionnage de vidéos pédagogiques, vulgarisatrices) nous permettant de mettre en question nos choix et ainsi de prendre les meilleures décisions afin d'avancer le plus efficacement possible dans le temps imparti. Et

également « un esprit d'innovation » constant, basé sur l'ensemble des notions apprises au préalable afin de concevoir le programme informatique et d'optimiser ce dernier.

III-LA RECONNAISSANCE D'ESSENCE DE BOIS

1- Mise en place de la problématique

A- Le contexte

En Asie du Sud-Est la forêt tropicale est constituée d'environ 3000 espèces de bois. Les principales ressources financières de la plupart des pays d'Asie du Sud-Est sont issues de l'exportation des produits du bois. La Malaisie est l'un des principaux exportateurs de produits du bois avec un total de plus de 10 milliards de dollars en 2006.

Chaque essence possède ses propres caractéristiques et donc ses propres usages. Par exemple, pour construire des toits relativement fiables, on utilise des bois avec une résistance acceptable comme le « *Neobalanocarpus heimii* » ou le nom local « chengal » et d'autre part pour la fabrication de meubles, on utilise un bois moins cher, « *Hevea brasiliensis* » ou simplement connu sous le nom de bois d'hévéa.

Ainsi, le besoin de reconnaître le bois est nécessaire car les prix varient considérablement entre les différentes essences de bois. L'identité d'un arbre dans la forêt peut être facilement connue en examinant ses fleurs, fruits et feuilles. Cependant, une fois abattu, l'identification devient plus difficile et se base sur ses caractéristiques physiques macroscopiques et microscopiques.

Il existe donc différentes méthodes pour différencier les essences de bois après qu'il ait été abattu. Ces méthodes sont basées sur les propriétés de base des matériaux.

- La Masse Volumique : Détermination par l'intermédiaire d'une pesée classique.
- La Couleur des bois : Que j'ai eu la chance de comparer grâce à la vaste Xylothèque de FCBA (Niveaux de vert, bleu, et rouge).
- Humidité : Mesure de l'humidité des bois à l'état vert (bois non sec)
- Mesure du PH : Mesure de la coloration en bout de planche à l'aide d'un réactif colorimétrique chimique (ex : Bleu de Bromothymol ou Bromophénol)

Ces méthodes généralement rapides à appliquer, présentent cependant quelques inconvénients :

-Manque de précisions : Les propriétés mesurées, hormis le pH, sont très variables sur une même essence. Cette variabilité est due aux nombreux facteurs entrant en jeu dans ces paramètres. Par exemple la masse volumique est dépendante de la localisation du bois dans l'arbre mais aussi de l'arbre dans le peuplement, des conditions du site de croissance et aussi de ses paramètres génétiques.

-Mesure non immédiate : La détermination de ces propriétés a beau être rapide (quelques dizaines de secondes), elle n'est pas instantanée et demande de la manipulation. C'est assez problématique par rapport aux chaînes de production toujours plus mécanisées.

-Dégradation des pièces : Ces tests ne sont pas destructifs mais peuvent avoir un impact, comme par exemple la mesure du PH qui génère une coloration en bout de planche.

- Approvisionnement en produit chimique : L'étude du PH nécessite par exemple un approvisionnement en réactifs chimiques.

Il existe un dernier inconvénient, le fait que toutes ces mesures doivent être effectuées par des personnes expertes dans des laboratoires. On peut, à titre d'exemple citer les attentes des viticulteurs qui font expertiser les fûts de vin dont ils se servent pour la production des AOP.

Un autre but de l'identification du bois est le contrôle de la fraude car certains commerçants de bois ont tendance à mélanger différents types de bois afin d'augmenter leur profit.

En raison de la pression des organisations non gouvernementales liées à l'environnement, tel que l'Agence Internationale de l'Environnement (AIE), de nombreux pays ont interdit l'exportation des espèces telles que le « Gonystylus Bancanus » ou localement connu sous le nom de Ramin ayant été largement sur-exploité.

Le FCBA a mis en place deux partenariats avec des chercheurs (américains et malaisiens) qui développent actuellement des programmes de classification autonome d'essence de bois par prise d'image. Ces programmes ont pour but d'être disponible sur smartphone et utilisable par des douaniers lors de différents contrôles de fraudes.

B- Programme de reconnaissance d'essences de bois par prise d'images

L'équipe d'FCBA a souhaité disposer de sa propre application afin d'étendre ses capacités de reconnaissance à des bois tempérés (non tropicaux) car ne possédant pas les mêmes caractéristiques. De ce fait, des premiers exemples de programme de classification d'images ont été conçu par des ingénieurs chercheurs de FCBA.

Ces programmes fonctionnent grâce à une technologie de réseaux de neurones (explications à venir à la page 19), à travers lesquels on donne en entrée des données : différentes caractéristiques extraites des images d'essences de bois (Châtaignier, Chêne européen, Epicéa bleu, Hêtre et Pin laricio).

C- Ma contribution à l'axe de recherche :

a- Utilisation du SPIR (Spectre Infra-Rouge)

Le contexte expliqué, je peux maintenant vous présenter ma contribution à l'axe de recherche. Mon maître de stage Patrice Garcia m'a demandé de créer un nouveau programme en langage python et permettant de classer les résineux Epicéa et Sapin avec une analyse de leurs spectres dans le domaine de l'infra-rouge proche. L'idée ici, est de créer un nouveau modèle prédictif utilisant de nouvelles bases de données (spectres infra-rouges) et permettant de compléter les modèles de reconnaissance des essences par prise d'image.

Ainsi, si on combine deux modèles prédictifs, l'un utilisant la prise d'image sur smartphone, et l'autre l'analyse spectrale dans le proche-infrarouge les chances de converger vers une bonne prédiction seront plus optimales.

b- Définition d'un objectif

Dans un premier temps, j'ai dû définir un objectif mesurable. L'objectif du programme fut d'obtenir la prédiction d'un spectre d'une essence de bois appartenant à une classe de Sapin (classe 0) ou d'Epicéa (classe 1).

Afin d'obtenir les performances de notre modèle, nous avons utilisé certaines métriques.

Par exemple si l'on veut une exactitude sur nos prédictions de 90%, cela veut dire que sur 100 spectres, on veut obtenir 90 bonnes réponses.

Sachant que les populations d'échantillons possèdent souvent des classes déséquilibrées, par exemple sur 100 échantillons, 90 % d'entre eux sont dans la classe 1 et 10 % d'entre eux sont dans la classe 0. Si l'on prend comme métrique « l'exactitude » on rencontre un certain problème. En effet, il est assez facile d'atteindre 90% de performances avec un modèle mal entraîné qui aurait été développé à prédire constamment des spectres d'Epicéa. La métrique d'exactitude ne prend pas en compte ce qu'on appelle les « faux positifs ».

Il existe alors différentes métriques comme la précision, le « recall » (sensibilité) et le score F1 (moyenne pondérée entre la précision et le « recall »).

La précision et le « recall » sont des mesures de performances calculées à partir des éléments que l'on trouve dans une matrice de confusion.

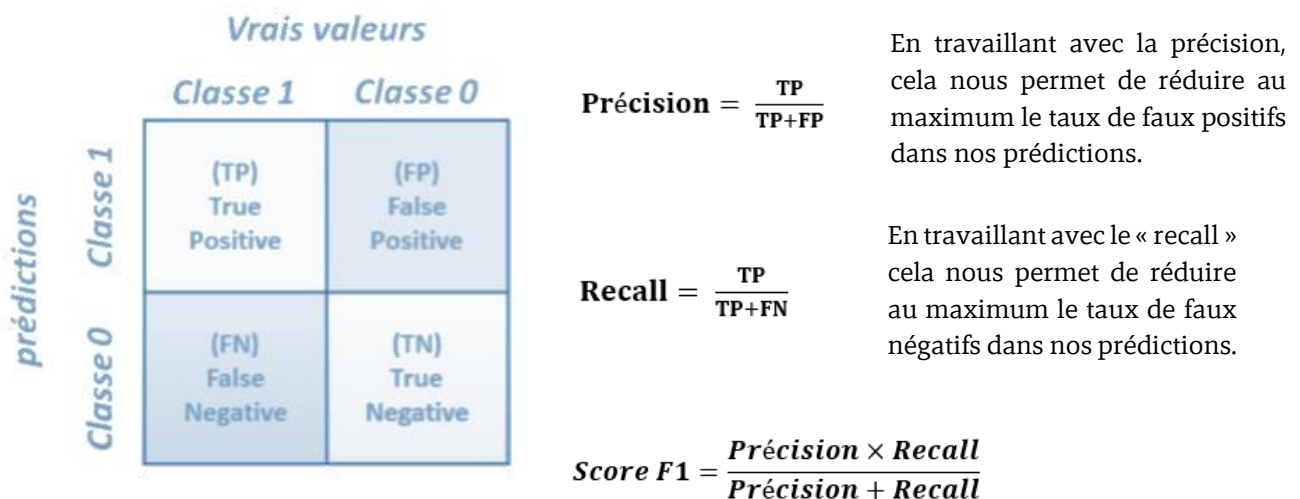


Figure 1. Matrice de confusion à deux classes

L'objectif que je me suis fixé est alors le suivant :

Obtenir la prédiction d'un spectre d'une essence de bois appartenant à une classe de Sapin ou d'Epicéa avec un score F1 = 70%.

D- Intelligence artificiel: Machine Learning / Deep Learning

Durant mon stage j'ai eu l'occasion de découvrir l'intelligence artificielle (IA) et deux de ses branches (Machine Learning et Deep Learning).

Premièrement qu'est-ce que l'IA ?

On la définit dans l'industrie par des algorithmes plus ou moins évolués qui imitent des actions humaines.

L'IA utilise la logique IF...THEN...ELSE... dans un programme sans qu'elle soit forcément intelligente. Par exemple une IA peut réaliser des tâches très simple comme déterminer le poids d'une personne en fonction de sa taille et son âge. Il existe cependant des intelligences artificielles dont le résultat se veut un minimum « intelligent ».

On en arrive au machine Learning. Vous trouverez le site d'où j'ai tiré l'exemple en bas de page (je le trouve très pédagogique) :

Imaginons que vous vouliez créer une IA qui vous donne le prix d'un appartement à partir de sa superficie.

Dans les années 1950, vous auriez fait un programme du type "si la superficie est inférieure à 20m², le prix vaut 60 000€, si elle est entre 20m² et 30m², le prix vaut 80 000€, etc...", ou peut-être "prix = superficie*3 000".

Si vous avez un ami statisticien, il pourrait alors vous dire que ces approximations ne sont pas satisfaisantes, et qu'il suffirait de constater le prix d'un grand nombre d'appartements dont on connaît la superficie pour estimer le prix d'un nouvel appartement de taille non-référencée ! Votre ami vient de donner naissance au machine Learning (qui est donc un sous-domaine de l'intelligence artificielle).

En effet apparu dans les années 1980, le Machine Learning est l'application de méthodes statistiques aux algorithmes pour les rendre plus intelligents. Malheureusement le désavantage du Machine Learning est qu'un expert humain doit, au préalable, faire un tri dans les données.

Par exemple, pour notre appartement, si vous pensez que l'âge du propriétaire n'a pas d'incidence sur le prix, il n'y a aucun intérêt à donner cette information à l'algorithme, car si vous lui en donnez trop, il pourrait voir des relations là où il n'y en a pas...

Ainsi le Deep Learning est né vers 2010 et s'inspire des réseaux de neurones biologiques reliés au cortex visuel.

Le Deep Learning repose donc sur les réseaux de neurones artificiels. Un réseau de neurones artificiel est un outil informatique permettant le traitement de données par l'apprentissage. Son architecture (dont j'explique le fonctionnement dans la partie 3.B.a) permet de se passer d'un expert humain pour faire le tri dans des données, puisque l'algorithme trouvera lui-même les corrélations.

2- Les données

A- Caractéristiques

L'étape clef de la reconnaissance d'un signal ou d'une image que ce soit biologiquement ou numériquement, réside dans l'extraction de caractéristiques pour les images et le prétraitement des données pour les signaux. Ce phénomène est si automatique pour un humain que nous n'y prêtons aucune attention. Pourtant nous sommes capables de différencier une forme d'une autre et un son d'un autre.

Mettons-nous dans la peau d'un jeune enfant qui apprend en maternelle à différencier un chat d'un chien. Il va au début essayer de trouver les différences propres à chaque animal (le chat à un petit museau, des petites pattes, des oreilles pointus etc... et le chien lui a un museau plus long, des grosses canines, de longues oreilles etc...) idem pour les sons (papa a la voix plus grave, plus rauque, tandis que celle de maman est plus douce et aigue).

Lorsque l'enfant reçoit sur sa rétine une information lumineuse ou un signal sonore à travers son système auditif, ses organes traduisent le message en signaux électriques à travers des nerfs.

A force d'apprentissage, le réseau de neurone va isoler les détails de l'image (contours, contrastes couleurs etc...) ou ceux du sons (la tonalité, l'intensité etc...) Et les associer à ce qu'il reconnaît d'après ses connaissances.

Maintenant nous pouvons nous demander quelle est la différence entre une image numérique et un signal (dans notre cas le spectre infra-rouge) ?

Premièrement qu'est-ce qu'une image numérique ? La définition d'une image s'exprime en pixels, ces pixels sont des petits carrés semblables à des carrés de mosaïque, l'image se forme grâce à cet ensemble ordonné de carrés ou pixels. Tous les pixels d'une image peuvent être attribué à une valeur afin d'identifier sa couleur (d'autres caractéristiques sont possibles ex : la transparence). Une fois numérisés les pixels sont classés dans une matrice comme ceci :

Voilà à quoi ressemble notre image numérique, prête à être analysée puis classée par un programme :

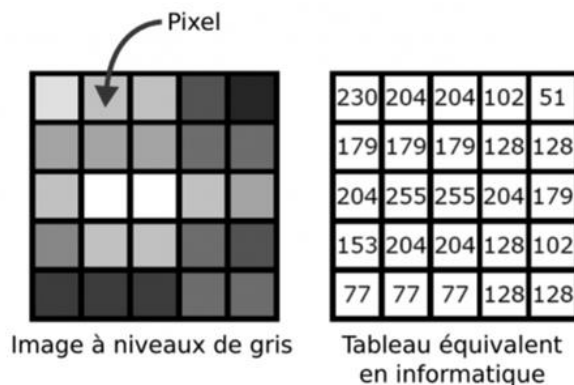


Figure 2. Représentation d'une image numérique

Après numérisation du signal du spectre infra-rouge, nous obtenons le même résultat, un vecteur composé de différents chiffres correspondant à l'absorbance de l'échantillon sur une plage de longueur d'onde spécifique.

Voilà à quoi ressemble notre spectre IR, prêt à être analysé puis classé par un programme.

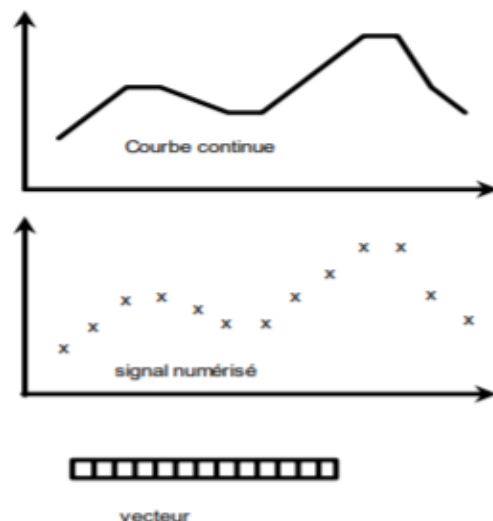
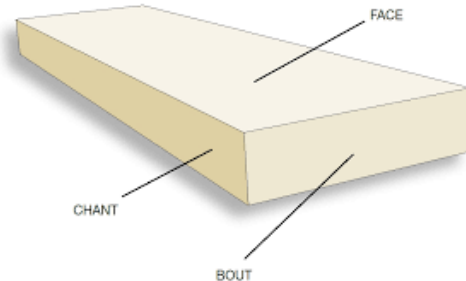


Figure 3. Représentation d'un signal numérique

On peut alors conclure qu'au niveau de la structure, elles sont presque identiques, cependant les valeurs discrètes du vecteur correspondant au spectre IR ont une allure continue, tandis que celles de la matrice correspondant à l'image ont un certain ordre mais n'ont pas une allure continue.

Par conséquent ce ne seront pas les mêmes caractéristiques que nous extrairons dans ces deux bases de données.

A l'aide d'un spectrophotomètre, nous avons prélevé les valeurs de transmittance d'une population de 286 spectres sur 96 éprouvettes en réalisant des prélèvements sur le bout et les champs de celle-ci.



Puis les spectres dont la plage de nombre d'onde va de **4000 à 28571 cm⁻¹** ont été associés à leur étiquette (l'Essence) puis classé dans un tableau appelé le Dataset :

Figure 4. Représentation des faces du bois

Etiquettes valeurs			Transmittance valeurs					
FACES	Essence		28571.428571428572	28490.02849002849	28409.090909090908	28328.611898017	28248.58757062147	28169.014084507045
File Name								
548_B1.asd	Bout	Epicea	0.088904	0.079458	0.076541	0.072554	0.063079	0.060449
548_B2.asd	Bout	Epicea	0.116825	0.108165	0.102722	0.097590	0.090969	0.089948
548_B3.asd	Bout	Epicea	0.103146	0.092693	0.087122	0.081760	0.073010	0.071344
548_C1.asd	Chant	Epicea	0.103750	0.095278	0.091988	0.088062	0.080067	0.077616
548_C2.asd	Chant	Epicea	0.108544	0.099469	0.100163	0.096249	0.081149	0.080983
...
156_C2.asd	Chant	Sapin	0.102667	0.094728	0.094464	0.091214	0.079913	0.079094
156_C3.asd	Chant	Sapin	0.115185	0.104814	0.100757	0.097165	0.089585	0.085422
156_F1.asd	Face	Sapin	0.081991	0.072520	0.071598	0.070766	0.063552	0.059123
156_F2.asd	Face	Sapin	0.097453	0.092445	0.086704	0.082150	0.078820	0.072601
156_F3.asd	Face	Sapin	0.101478	0.090062	0.088689	0.086557	0.076884	0.075393

286 rows × 2153 columns

Figure 5. Tableau du Dataset composé des spectres d'Epicea, et de Sapin

B- X et y trainset et X et y testset

Lorsque nous voulons évaluer les performances d'un modèle on n'utilise jamais les données qui ont servi à l'entraînement du modèle. Pourquoi ?

Imaginons que nous entraînons une machine à reconnaître des photos de chats puis nous l'évaluons sur ces mêmes photos. La machine va directement prédire que ce sont des chats puisqu'elle a déjà vu ces photos. En revanche ce qu'il nous intéresse c'est d'entraîner la machine sur des données jamais vues. Le Dataset va donc être divisé aléatoirement en deux parties : le trainset et le testset.

X et y trainset

228 rows × 2153 columns

X et y testset

58 rows × 2153 columns

3- Les modèles

A- Choix du modèle

Dans cette partie, nous allons explorer les différents critères permettant de réaliser le choix d'un modèle. Un réseau de neurones peut être considéré comme un modèle sophistiqué pour les raisons suivantes :

- Il s'agit d'un modèle consommant beaucoup de données pour être entraîné.
- Il met beaucoup de temps à être entraîné.
- il est complexe à régler.

Nous devons alors vérifier si un modèle plus simple ne suffirait pas pour réaliser nos prédictions. Travaillant sur ce problème de classification, j'ai alors identifié des modèles spécialisés dans ce domaine tels que LogisticRegression, LinearSVC ou KNeighborsClassifier. Maintenant lequel choisir ?

Critère 1 : La quantité de données :

Certains modèles peuvent travailler avec de gros Datasets (+100, 000 données)

- Régression Logistique
- Réseaux de Neurones

D'autres sont très lent pour manipuler tant de données (-100 000 données)

- Support Vector Machines
- K-Nearest Neighbours

On explique cette différence par le fonctionnement de ces algorithmes. Par exemple le K-Nearest Neighbours retient la totalité du dataset pour ensuite effectuer ses prédictions en calculant la distance entre le nouveau point et tous les points se trouvant dans le dataset. L'explication en détail de ce modèle sera faite dans la partie 18.

Tandis qu'un réseau de neurones afin d'être performant nécessitera une grosse quantité de données.

Critère 2 : Structure des données :

-Les données non structurées : Comme les images, le texte, ou le son.

Les réseaux de neurones, sont très efficaces sur ce genre d'application.

-Les données structurées : Les données que l'on classe dans des fichiers Excel (le poids, la taille, la superficie, ou bien la longueur d'onde).

Les modèles de machine Learning classique fonctionnent très bien avec ces applications.

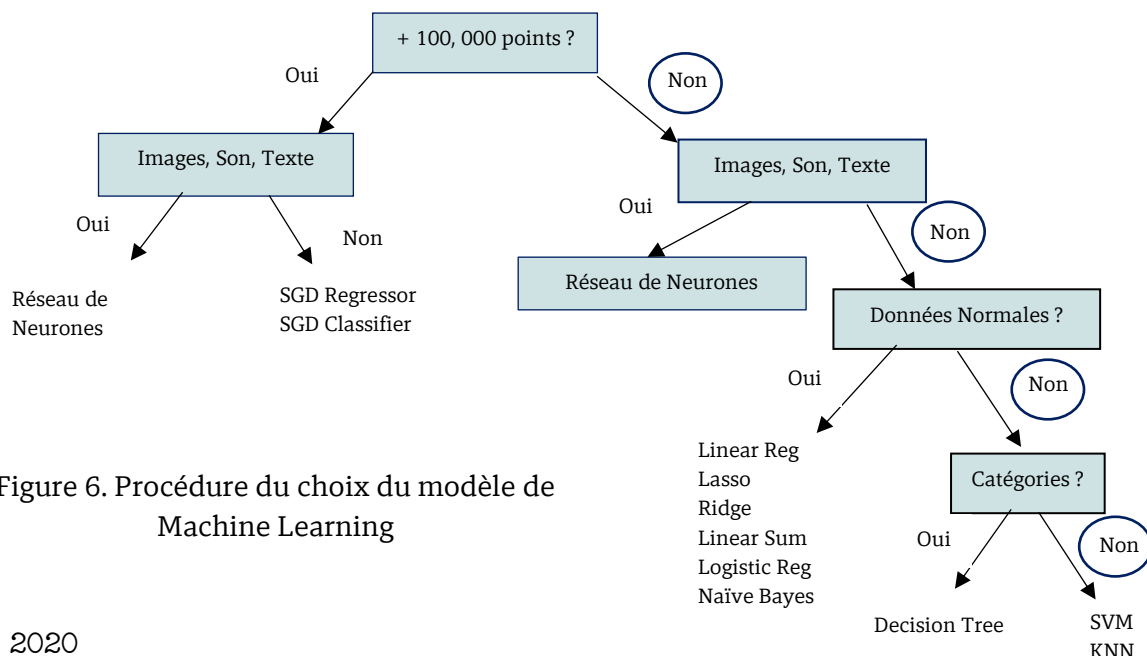


Figure 6. Procédure du choix du modèle de Machine Learning

Possédant une base de données de 228 spectres (ce sont nos points), qui sont à la fois des données structurées (car ce sont des données continues contrairement aux pixels d'une image par exemple) et avec une seule variable catégorique (faces sur lesquelles sont prélevés les spectres : Face, Bout, Champs), un algorithme de K-Nearest-Neighbors (KNN) ou un Support-Vector-Machine (SVM) sont les algorithmes préconisés dans notre situation.

Mais le seul moyen de savoir quel est le meilleur modèle, c'est de s'en tenir aux faits ! C'est pour cela qu'il est judicieux de tester plusieurs modèles et de choisir celui qui a la meilleure performance.

Même si les modèles de machines Learning semblent être plus adaptés à cette situation, j'ai tout de même réalisé un réseau de neurones sur la demande de mon maître de stage.

B- Deep Learning / Réseaux de neurones

a- Fonctionnement d'un réseau de neurones

Les caractéristiques représentent alors nos données d'entrée de notre réseau de neurones.

Ce type de réseau de neurones, s'appelle un perceptron multicouche. Il est composé d'une couche de neurones d'entrée, d'une couche cachée (ou intermédiaire), et d'une couche de neurones de sortie.

Un perceptron multicouche peut avoir une multitude de conceptions (nombre de couches différentes, nombre de neurones différents, qui sont des caractéristiques essentielles de l'optimisation de notre réseau de neurones).

Afin de comprendre le fonctionnement, nous allons nous concentrer sur un réseau de neurones composé de trois couches une couche d'entrée, deux couches cachées et une couche de sortie (les entrées ne comptent pas comme couches dans la terminologie ANN).

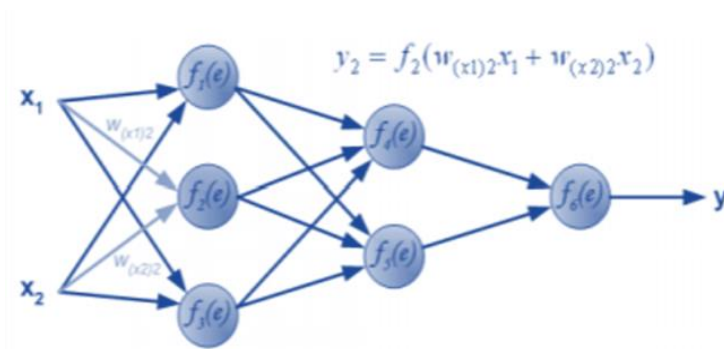
Couche d'entrée : représente les valeurs spectrales enregistrées ; dans le cas des spectres de l'infra-rouge proche des résineux épicéa et sapin, nous disposons d'enregistrements pour tous les 1 cm^{-1} de 350 cm^{-1} à 2500 cm^{-1} ce qui correspond à 2150 données, mais nous allons voir dans la partie 4.C-2.a que seulement 10 longueurs d'ondes sont sélectionnées après le pré-traitement et utilisé dans le réseau de neurone, ce qui correspond à 10 données d'entrées.

Couches intermédiaires ou couches cachées : leur nombre peut varier et leur composition en nombre de neurones également. Ces paramètres dépendront de l'optimisation du modèle.

Couche de sortie : la couche de sortie correspond aux différentes classes. Dans notre cas nous avons deux classes Epicéa et Sapin, la couche de sortie sera donc composée d'une couche d'un seul neurone. Cherchant à prédire en pourcentage à quelle classe appartient le spectre mis en entrée. (Par exemple : le neurone va prédire que le spectre appartient à la classe 1 (Epicéa) si le pourcentage est proche de 1 (Epicéa à 80%) et il prédira qu'il appartient à la classe 2 (Sapin) si le pourcentage est proche de 0 (Sapin à 20%).

L'entraînement du réseau de neurones se décompose en deux parties, la propagation et la rétropropagation.

Propagation : Durant cette première phase, les poids (w_i) sont initialisés aléatoirement.



Chaque neurone de la couche cachée va réaliser la somme des valeurs des données d'entrée multipliées par les poids de sa synapse (jonction entre deux neurones) des chemins précédant. La valeur obtenue sera transformée par le biais d'une fonction non-linéaire appropriée.

Figure 7. Schéma d'une propagation avant

Voici la formule mathématique :

$$y(t) = f\left[\sum_{i=0}^n w_i(t)x_i(t)\right]$$

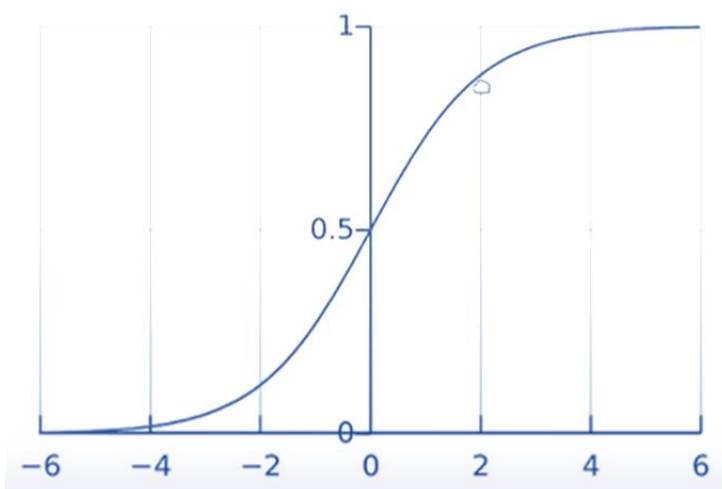
Avec ;

$w_i(t)$ = le poids d'un neurone

$x_i(t)$ = l'entrée du neurone

et $f(t)$ = une fonction

Il existe une multitude de fonction de transfert non linéaires pouvant être utilisées à des fins différentes afin de fournir des solutions optimales dans des applications particulières.



La fonction sigmoïde réduit la valeur d'entrée (qui va de $+\infty$ à $-\infty$), entre 0 et 1. Cette fonction nous permet donc d'avoir une probabilité en sortie. Quand les valeurs d'entrée sont inférieures à 0, l'activation est inférieure à 0,5 et quand la valeur supérieure à 0 elle va être supérieure à 0,5.

Nous retournant une probabilité, cette fonction peut très bien être placée au niveau de notre couche de sortie réalisant ainsi la prédiction de l'essence en pourcentage (c'est un sapin à 90 %).

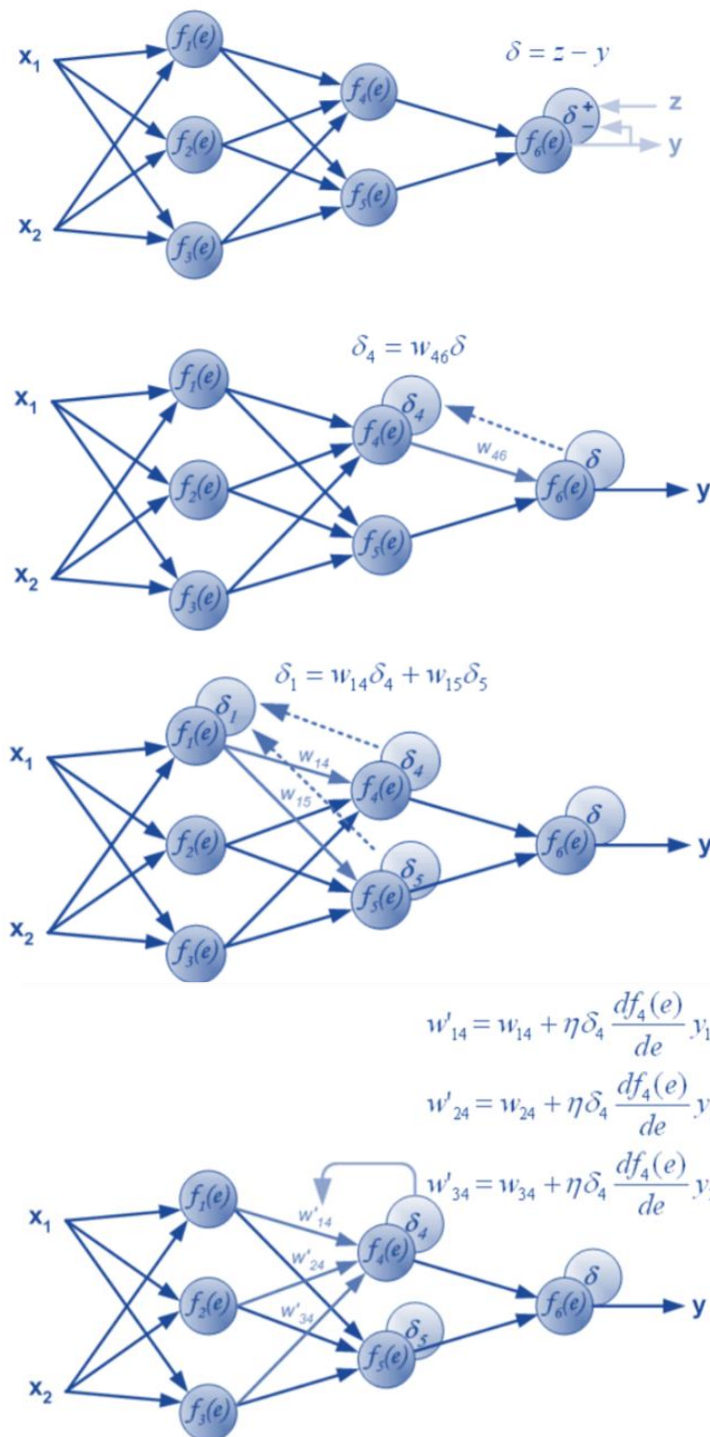
Figure 8. Fonction sigmoïde

Il est donc très important de bien choisir ces fonctions de transfert pour chaque neurone car elles permettront de bien faire converger les données d'entrée vers la bonne prédiction avec la meilleure précision.

La rétro-propagation :

Nous pouvons voir les choses de cette manière : à l'instar des neurones biologiques qui à force d'apprentissage, renforcent leurs liaisons synaptiques, les réseaux de neurones artificiels, lors de l'entraînement, modifient leurs poids, afin de permettre aux valeurs d'entrée de converger vers une bonne prédiction (suite à des multiplications, additions et transformation par fonction de transfert).

Figure 9. Différentes étapes de la rétropropagation



Le signal de sortie du réseau y est comparé à la valeur de sortie souhaitée (la cible qui est égale à 1 pour le sapin ou 0 pour l'épicéa), qui se trouve dans l'ensemble de données d'apprentissage. La différence est appelée signal d'erreur δ du neurone de la couche de sortie.

L'idée est de propager le signal d'erreur δ (calculé en une seule étape d'enseignement) à tous les neurones.

Un signal d'erreur est créé pour chaque neurone en multipliant la valeur des coefficients précédents par le poids des chemins précédents.

Lorsque le signal d'erreur pour chaque neurone est calculé, les poids de chaque nœud d'entrée de neurone peuvent être modifiés.

Dans les formules ci-contre, $df(e)/de$ représente le dérivé de la fonction d'activation des neurones (dont les poids sont modifiés).

Le coefficient η affecte la vitesse d'apprentissage du réseau. Il existe quelques techniques pour sélectionner ce paramètre.

Afin de vérifier la validité de notre modèle, on réalise ensuite une phase de test sur l'échantillon de test. Le principe est le même que la phase d'entraînement, à la différence qu'ici les poids ne changent pas et que l'on regarde juste le résultat. Si les résultats lors de cette phase ne sont pas équivalents à la phase d'entraînement alors il y a un phénomène de sur-apprentissage. Ce phénomène est expliqué dans la partie 5-D-b.

C- Machine Learning / Algorithme K-Nearest-Neighbors

a- Fonctionnement

Avant de commencer à construire un modèle de réseau de neurones qui est complexe à régler, j'ai commencé par utiliser un modèle KNN (K-Nearest-Neighbor).

Il s'agit d'un algorithme d'apprentissage automatique non paramétrique, ce qui signifie que la structure du modèle est déterminée à partir de l'ensemble de données.

C'est un algorithme qui n'a besoin d'aucune donnée d'entraînement pour la génération du modèle. Toutes les données d'entraînement sont utilisées dans la phase de test. Cela rend la discrimination des points plus rapide mais le KNN a besoin de plus de temps pour analyser tous les points de données et cette analyse nécessitera plus de mémoire pour stocker les données d'entraînement.

Comment fonctionne l'algorithme KNN ?

Dans KNN, K est le nombre de voisins les plus proches.

Le nombre de voisins est le principal facteur décisif. K est généralement un nombre impair si le nombre de classes est 2.

Lorsque $K = 1$ supposons que P1 soit le point pour lequel on doit prédire l'étiquette (appartenance à un groupe ou à un autre).

Tout d'abord, l'algorithme trouve le point le plus proche de P1, puis donne l'étiquette du point le plus proche à P1.

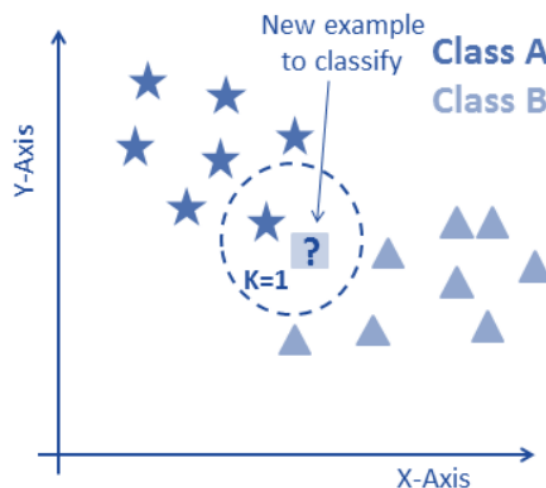


Figure 10. Répartition des points de deux classes suivant deux variables (Y-Axe et X-Axe).

KNN a les étapes de base suivantes :

- 1- Calcul des distances entre le point pour lequel on doit prédire l'étiquette et les points les plus proches. On utilise la distance Euclidienne, la distance de Hamming, la distance de Manhattan ou la distance de Minkowski.
- 2- Trouver les K voisins les plus proches.
- 3- Attribuer l'étiquette au point par simple vote majoritaire ou un vote pondéré par les distances.

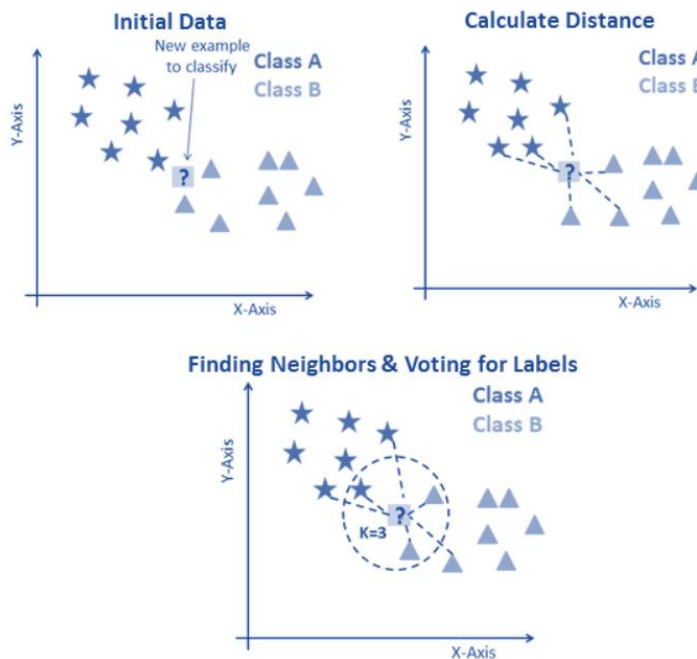


Figure 11. Les Etapes de bases du KNN

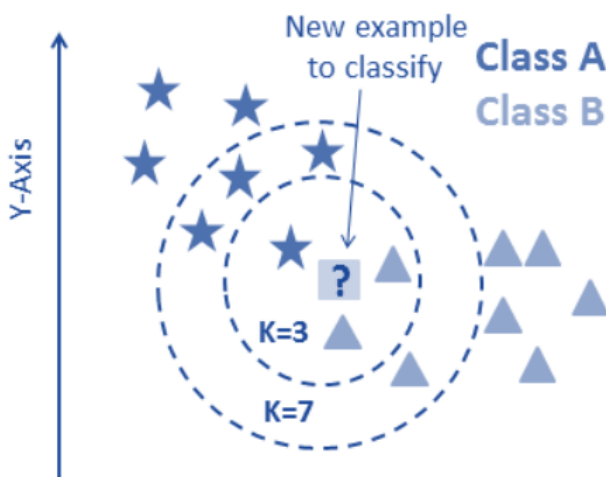


Figure 12. Démonstration de l'influence du choix de K.

Comment choisir le nombre de voisin ?

La recherche a montré qu'aucun nombre optimal de voisins ne convient à tous les types d'ensembles de données. Chaque ensemble de données a ses propres exigences. Dans le cas d'un petit nombre de voisins, le bruit aura une influence plus élevée sur le résultat, et un grand nombre de voisins le rendra coûteux en calcul.

En règle générale, les scientifiques spécialistes du traitement des données choisissent un nombre impair si le nombre de classes est pair.

Mais dans notre cas nous avons généré le modèle sur différentes valeurs de k et vérifié leurs performances.

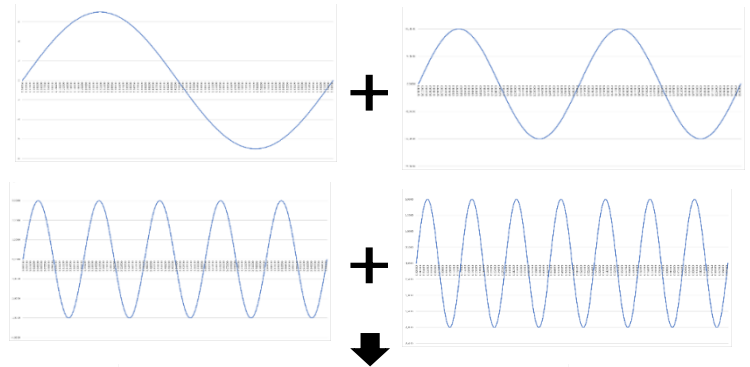
4- Etudes des signaux

A- Comprendre le signal issu d'un spectromètre infra-rouge avec la transformé de fourrier

Afin de comprendre l'ensemble de mon axe de recherche, j'ai eu l'occasion de découvrir ce qu'il se cachait mathématiquement derrière un spectre infra-rouge. Je vous propose alors ici de découvrir comment à partir d'un signal, nous parvenons à obtenir un spectre infra-rouge.

Nous avons créé plusieurs signaux sinusoïdaux de fréquence 1Hz, 2Hz, 5Hz et 7Hz sur un temps de 1sec échantillonnés avec un pas de 1/500 sur Excel :

En réalisant la somme de ces signaux, on obtient un signal temporel quelconque.
Un signal est donc une somme de fonctions sinusoïdales de fréquences connues.



$$X(t) = a \times \sin(2\pi f_1 t) + b \times \sin(2\pi f_2 t) \\ + c \times \sin(2\pi f_3 t) + d \times \sin(2\pi f_4 t)$$

Avec a, b, c et d : amplitude des 4 sinus composites de fréquences respectives f1, f2, f3 et f4.

Figure 13. Signal = somme de signaux.

En multipliant le signal crée avec des sinus de fréquences 1Hz, 2Hz, 5Hz, 7Hz, 3Hz et 6 Hz, on obtient les signaux suivants :

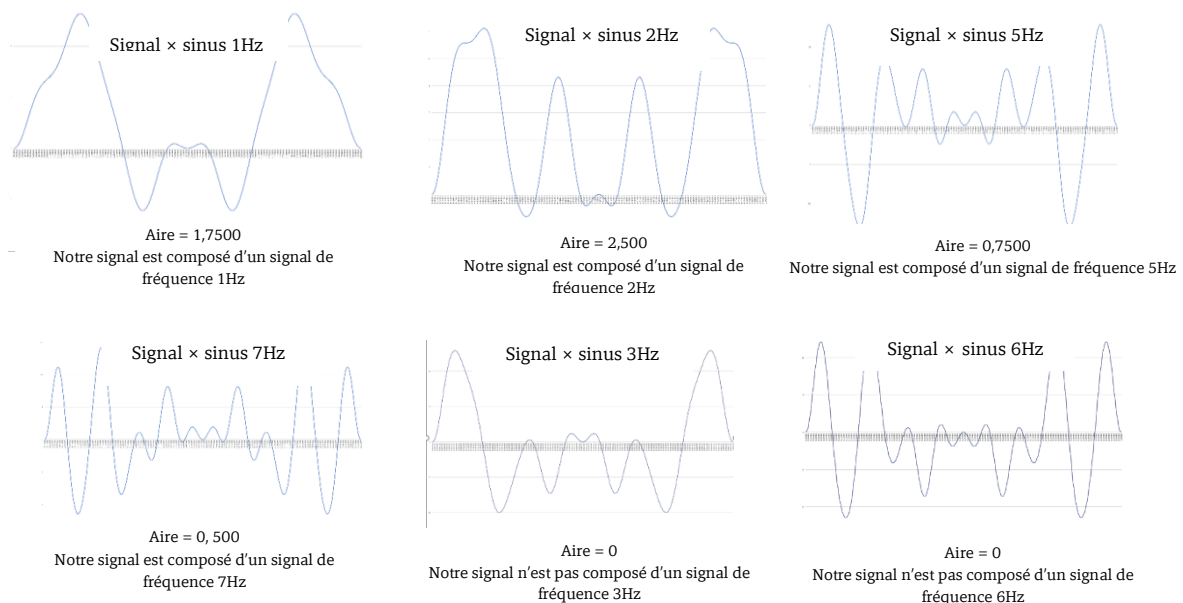


Figure 14. Intégration du signal multiplié par plusieurs signaux

On se rend compte que l'aire sous la courbe des signaux ci-dessus multipliés par un sinus qui appartient au signal est égale à 0.

Ainsi, on va répéter cette opération avec des sinus de différentes fréquences et on saura quelles sont toutes les fréquences présentes dans notre signal.

Nous venons de voir comment fonctionne le principe d'une transformée de fourrier et qu'elle revient donc à faire un calcul d'aire (une intégration). Voici la formule complète :

On voit dans la formule que le calcul de la transformée de fourrier fait intervenir les nombres complexes (à cause du i), mais je vous épargne les détails de cette partie.

$$y(f) = \int_{-\infty}^{\infty} x(t)e^{2\pi ift} dt = \int_{-\infty}^{\infty} x(t)(\cos 2\pi ft + i \sin 2\pi ft) dt$$

Pour conclure on peut dire qu'elle permet le passage du domaine temporel à un domaine fréquentiel.

Suite à une transformée de fourrier un signal périodique va présenter un spectre d'amplitude discret dont les différents traits représentent les différents termes :

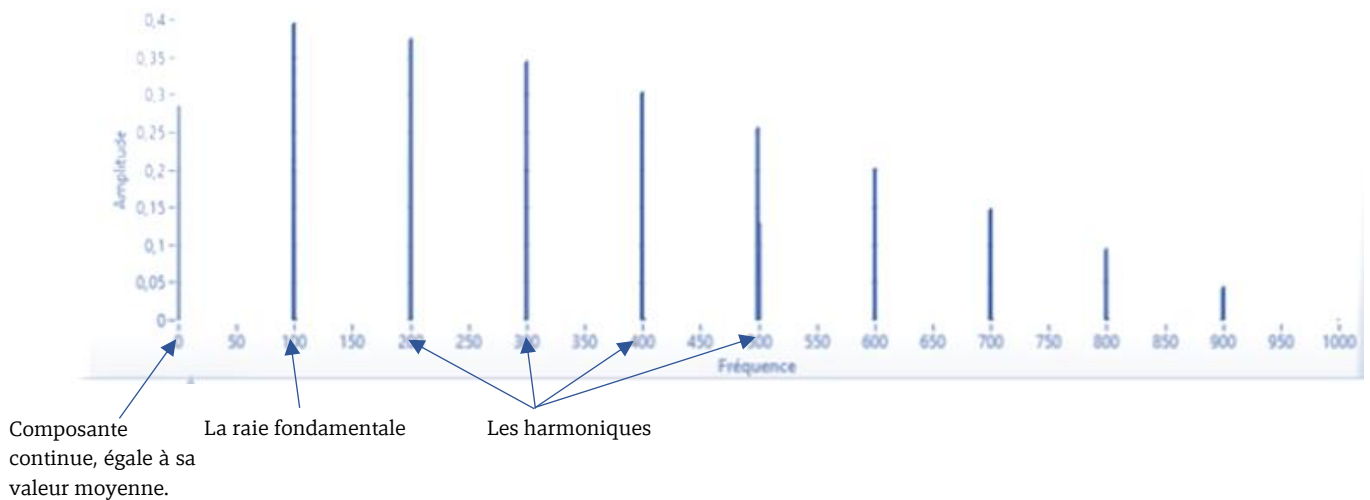


Figure 15. Transformée de fourrier d'un signal périodique

Un signal non périodique va présenter un spectre d'amplitude non discret :

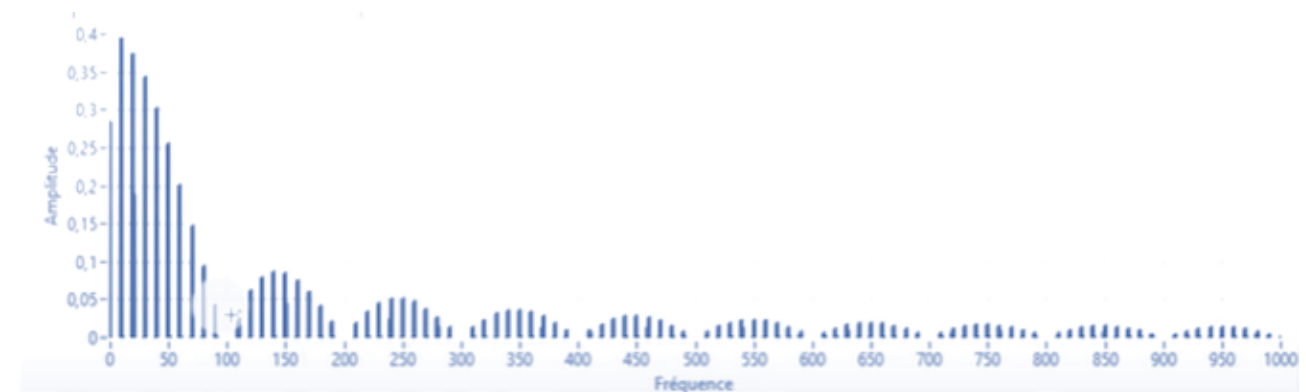


Figure 16. Transformée de fourrier d'un signal non périodique

B- Spectroscopie infra-rouge proche (NIR)

La spectroscopie infra-rouge proche est basée sur l'absorption du rayonnement par le matériel analysé pour des longueurs d'onde allant de 350 nm à 2500 nm soit en nombre d'onde de 4000 cm^{-1} à 28571 cm^{-1} . L'absorption se fait en fonction des vibrations des liaisons chimiques de l'échantillon. Dans le domaine de l'infra-rouge proche, les absorptions ne sont pas dues aux vibrations fondamentales des molécules mais à des vibrations harmoniques et des vibrations de combinaisons. Il en résulte des bandes d'absorption très larges et un fort chevauchement des bandes d'absorption des différentes liaisons chimiques. Il est donc difficile d'assigner des bandes d'absorption caractéristiques chimiques précises de l'échantillon.

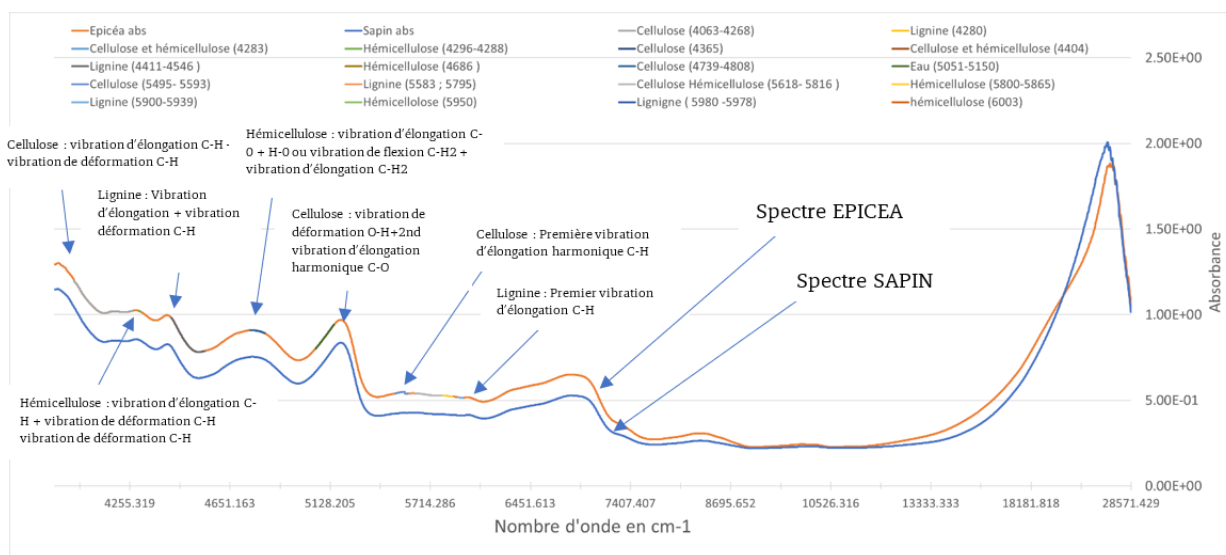


Figure 17. Transformée de fourrier d'un signal non périodique

Sur la figure ci-dessus on observe les variations de bandes spectrales que nous avons réussi à identifier qui résultent des vibrations harmoniques et des vibrations de combinaisons pour les deux échantillons de bois, Sapin et Epicéa (les bandes d'absorption se chevauchant dans l'infra-rouge proche, l'identification des liaisons reste approximative). Nous remarquons que les bandes spectrales les plus importantes se situent essentiellement dans la gamme 4000 à 8000 cm^{-1} . Le reste du spectre concernant la gamme allant au-delà de 8000 cm^{-1} n'apportant aucune information sur les liaisons des molécules des essences, risque de fausser la prédiction, il est alors intéressant de la supprimer.

La spectroscopie NIR est couramment utilisée dans l'industrie. Elle sert à l'analyse quantitative et qualitative des produits agricoles, pharmaceutiques, polymères et l'analyse biologique. Elle permet notamment de connaître la composition chimique des aliments et des matières premières.

La spectroscopie NIR fortement connu pour être une technique rapide et non destructive pour l'analyse qualitative et quantitative des produits agricoles allant des fourrages aux céréales en passant par le tabac et le bois. Elle a été utilisée pour évaluer la prédiction des composés de la biomasse lignocellulosiques (canne à sucre, miscanthus, fourrages, maïs, etc.).

Vous trouverez en annexe le tableau présentant les bandes spectrales qui correspondent aux groupes fonctionnels chimiques rencontrés dans les composés organiques identifiés par la spectroscopie NIR.

Dans notre axe de recherche, nous ne cherchons pas à réaliser une prédiction sur la composition des essences de bois Sapin et Epicéa, mais à différencier ces deux classes à partir d'un spectre prélevé de d'un échantillon. Ce sont alors les différences de variations de bandes d'absorptions correspondant à certaines liaisons chimiques de ces deux classes qui vont permettre au modèle de différencier le Sapin de l'Epicéa.

Malheureusement le fort chevauchement des bandes spectrales des spectres infra-rouges et la présence d'interférences rend la prédiction des deux classes plus compliquée et moins précise. Alors afin d'augmenter le taux de précision de notre modèle, différentes méthodes de prétraitement peuvent être appliquées avant l'incorporation des spectres dans le réseau de neurones. C'est ce que nous allons voir dans la partie suivante.

C- Les Prétraitements des spectres dans l'infra-rouge proche :

C-1 Correction du phénomène de diffusion

Etant donné que plusieurs spectres ont été obtenu sur plusieurs échantillons de bois, extraits de différentes parcelles, à différentes périodes, il est logique de considérer que les prétraitements les plus adaptés pour la gamme spectrale fournissent des classes les plus compactes et les plus différenciées après le passage des données dans le modèle prédictif.

En effet les échantillons n'étant pas indéniablement identiques du fait des différentes variations présentées précédemment, les deux classes ne peuvent donc pas se rassembler sur deux points distincts, néanmoins elles devraient se répartir en deux nuages de points distincts l'un de l'autre comme nous pouvons le voir sur le schéma ci-contre.

Nous cherchons deux classes : les + compactes et les + distinctes l'une de l'autre.

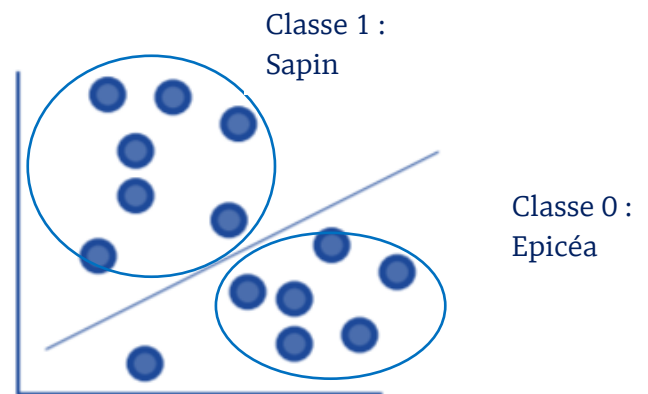


Figure 18. Répartition des points de deux classes suivant deux variables (Y-Axe et X-Axe).

Les spectres infra-rouges sont affectés par certaines interférences : les hétérogénéités ou les impuretés de l'échantillon, des perturbations rencontrées sur le chemin optique, la lumière extérieure, ou bien l'humidité de l'échantillon etc.

Afin d'éliminer ou de diminuer en partie l'influence de ces interférences, il est indispensable d'appliquer des prétraitements mathématiques sur ces spectres. Il en existe deux grandes catégories : les prétraitements de correction de dispersion (scatter-correction) et ceux de dérivation spectrale.

L'objectif de cette partie est donc de présenter les trois prétraitements les plus utilisés pour traiter des spectres NIR afin d'obtenir de meilleures prédictions.

a- La ligne de base

Avant de vous présenter les différents prétraitements, je vais vous présenter la notion de ligne de base.

Souvent apparaissent sur les spectres infra-rouges des déviations d'intensités qui vont être à l'origine de la ligne de base. La ligne de base est un signal qui s'ajoute au signal utile (contenant les raies du spectres) et elle est principalement due à des phénomènes de diffusions :



Figure 19. Spectre = somme spectre pur et ligne de base

Explication du phénomène de diffusion à l'origine de la ligne de base :

Lorsqu'un signal rencontre une interface rugueuse, une partie de l'énergie incidente est dispersée et génère des ondes secondaires dans toutes les directions. Ces ondes secondaires sont diffusées dans les directions qui dépendent des irrégularités du fond. C'est le phénomène de diffusion

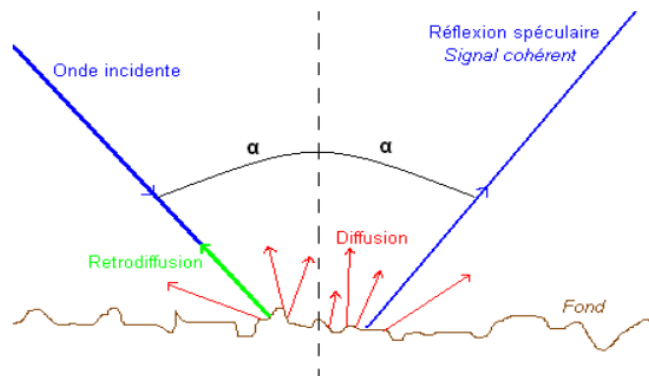


Figure 20. Diffusion et réflexion sur le fond d'une onde.

Par exemple, si l'échantillon a une surface rugueuse ou contient de la poudre de composé inorganique, la diffusion de la lumière infrarouge à la surface de l'échantillon ou à l'intérieur sera plus importante pour les longueurs d'ondes les plus courtes.

En outre, comme la ligne de base provient de lumière intrinsèque ou extérieure à l'expérience, elle constitue un signal qui vient s'ajouter au spectre.

La correction de cette ligne de base basé sur des méthodes numériques, consiste donc à soustraire la ligne de base au spectre mesuré. Il en résultera une élimination de l'influence de ces phénomènes perturbateurs et donnera accès à un signal plus interprétable.

b- La Dérivation

La dérivation de spectres est un prétraitement très couramment appliqué en spectroscopie IR. Elle permet de séparer des pics ou des bandes d'absorption qui se chevauchent alors que le bruit de fond basse-fréquence des spectres devient proche de zéro. La ligne de base se trouve de ce fait corrigée. Malheureusement, l'opération de différenciation amplifie le bruit haute fréquence et augmente la complexité des spectres.

Afin de démontrer l'effet de la dérivation, on suppose que les spectres sont déformés par l'ajout d'une ligne de base de la forme :

$$x_i = a + b\lambda_i + z_i \quad \text{Avec } z_i \text{ qui représente la partie « informative » du spectre. Et } \lambda_i \text{ le vecteur des nombres d'ondes}$$

La dérivation première donne :

$$\frac{dx_i}{d\lambda_i} = b + \frac{dz_i}{d\lambda_i} \quad \text{Le terme } a, \text{ représentatif de la « hauteur » de la ligne de base a disparue}$$

Et la dérivation seconde donne :

$$\frac{d^2x_i}{d\lambda_i^2} = \frac{d^2z_i}{d\lambda_i^2} \quad \text{Cette relation démontre que quelle que soit la ligne de base, perturbant la mesure spectrale, son effet disparaît lorsque les spectres sont mis sous la forme de leurs dérivés secondes.}$$

Afin d'illustrer les effets de la dérivée première, voici une démonstration graphique réalisé sur Excel :

On simule deux bandes d'absorption à l'aide de deux courbes de gauss.

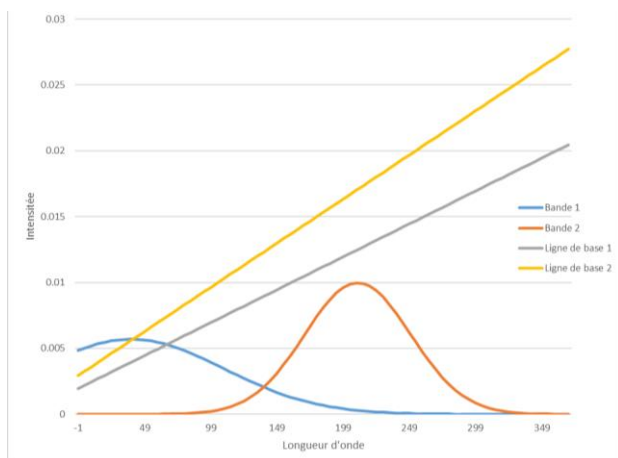


Figure 21. Simulation de deux bandes d'absorptions

On réalise l'enregistrement de deux spectres (somme des deux courbes de gauss) perturbés par l'ajout de deux lignes de base différentes variant au niveau de leurs paramètres.

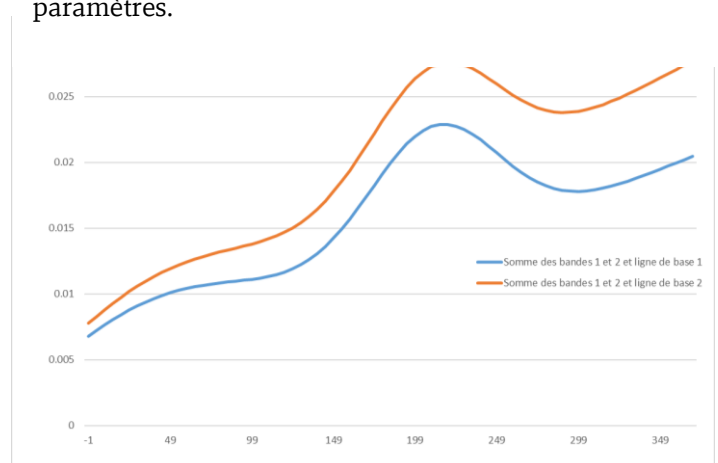


Figure 22. Spectres simulés, perturbés par des lignes de bases

Après dérivation seconde des deux spectres, ils se superposent. Prouvant alors la suppression de la ligne de base des deux spectres.

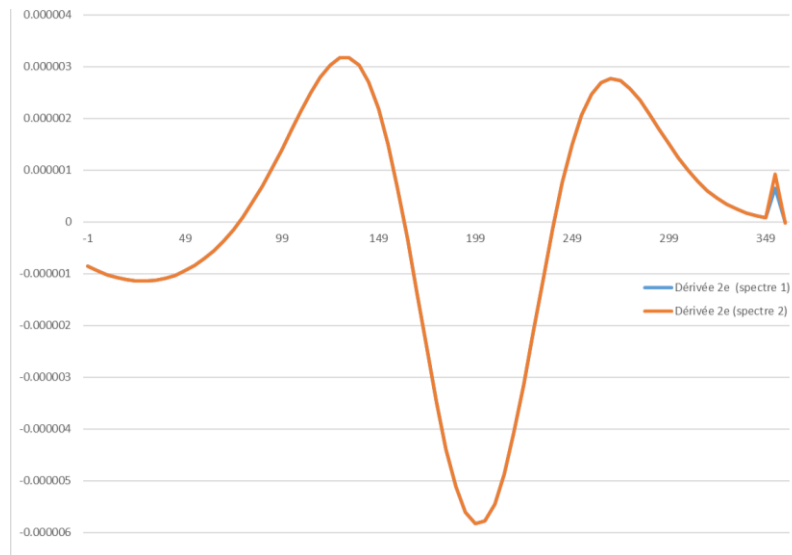


Figure 23. Dérivation seconde des deux spectres

c- Le lissage de Savitzky-Golay (SG)

Les spectres infra-rouges sont parfois chargés de bruits aléatoires. Un lissage permettra de diminuer ce bruit. Il existe différents algorithmes de lissage, le plus simple est la méthode des moyennes glissantes :

-Elle consiste à considérer une fenêtre, un intervalle, i comme étant $[i - m ; i + m]$, l'intervalle de milieu i de largeur $2m+1$. Avec m la demi-largeur (nombre de points).

-Considérer un polynôme de degré d , avec $d < 2m+1$. Pour chaque intervalle i , on effectue une régression pour déterminer le polynôme P_i minimisant l'erreur au sens des moindres carrés.

La valeur du spectre lissé est définie comme :

$$y_{\text{liss}} = P_i(x(i)).$$

La figure ci-contre montre le processus de formation du lissage par Savitzky-Golay (SG) pour un spectre $x(i)$. La courbe formée par les points bleu représente le polynôme P_i estimé sur m points. Les paramètres de la méthode sont : le nombre de points m (4 sur la figure), le degré du polynôme d (3 sur la figure).

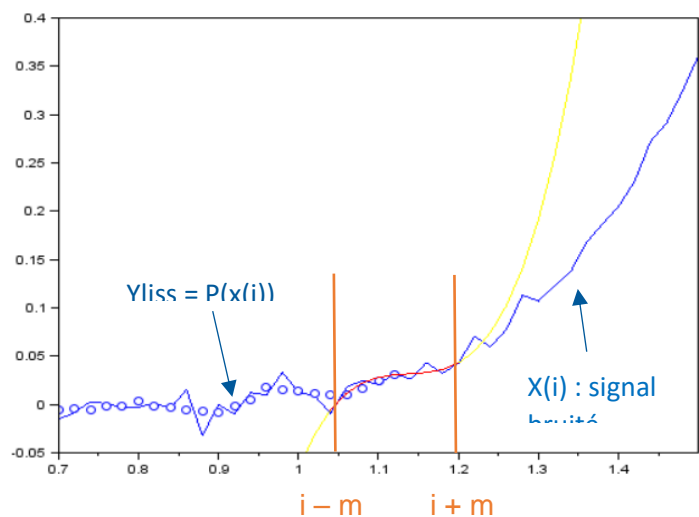


Figure 24. Lissage par Savitzky-Golay

Le polynôme étant au moins de degré 1, on peut déterminer la dérivée première :

$$y'_{\text{liss}} = P'_i(x(i)).$$

Le polynôme étant au moins de degré 2, on peut déterminer la dérivée seconde.

Cette méthode de prétraitement conduit alors à la suppression des effets de la variation incontrôlée (le bruit), suivit d'une dérivation on supprime alors la ligne de base du spectre.

d- La Normalisation (squatter-correction) :

La normalisation (Standard Normal Variate en anglais, SNV) étant l'une des méthodes de prétraitement les plus utilisées en spectroscopie infrarouge pour corriger des problèmes de dispersion.

Cette méthode permet de centrer le spectre sur sa moyenne et réajuster l'ensemble de l'intensité grâce à l'écart type du spectre. Dans une application de spectroscopie IR, nous pouvons construire des spectres corrigés par SNV. Le spectre corrigé par SNV est donné par la relation suivante :

$$xcor_j = \frac{x_j - \bar{x}}{\sqrt{\frac{\sum_{k=1}^p (x_k - \bar{x})^2}{p-1}}}$$

Soustraction de la moyenne du spectre et division par son écart-type.

La figure suivante représente le résultat obtenu avec la méthode de prétraitement SNV sur le spectre d'un échantillon d'Epicéa.

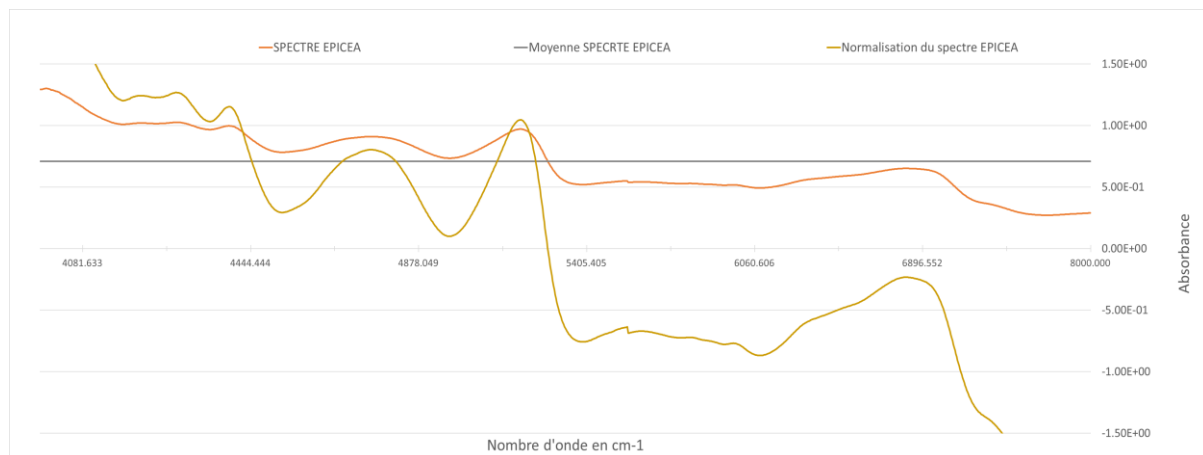


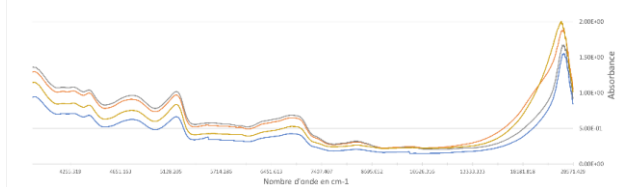
Figure 25. Prétraitement appliqué à un spectre d'Epicéa

La normalisation SNV est très utilisée pour les spectres infra-rouges car elle permet de réduire les effets de diffusion. En effet, la diffusion peut être responsable d'une réduction de l'intensité des spectres pour certaines longueurs d'ondes. Celle-ci peut varier considérablement entre deux échantillons (la surface sur laquelle on prélève l'échantillon, peut posséder des interfaces plus ou moins rugueuse etc.). Cette correction qui met l'ensemble des spectres centré sur 0 supprime ces déviations verticales d'intensité.

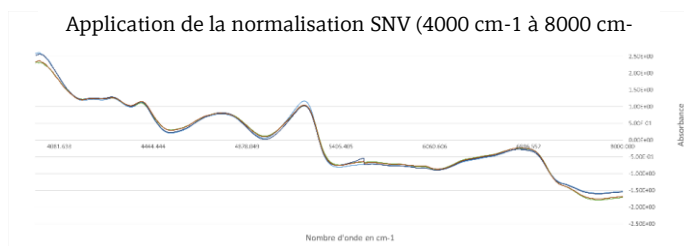
e- Application des prétraitements aux spectres IR

Après avoir expliqué dans les étapes précédentes le fonctionnement des différentes méthodes de prétraitement, dans cette section nous allons appliquer ces méthodes de prétraitement sur les spectres infra-rouge proche qui ont été recueillis sur les échantillons des deux essences Sapin et Epicéa. Nous procéderons ensuite à l'analyse de ces spectres.

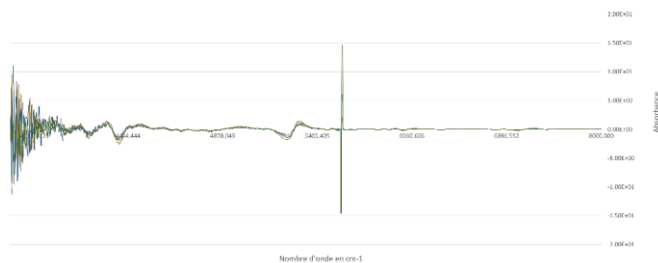
Spectres bruts de Sapin et d'Epicéa (4000 cm⁻¹ à 28490 cm⁻¹)



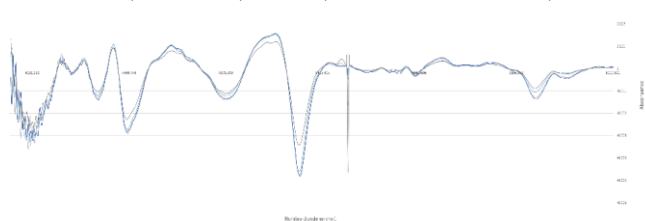
Application de la normalisation SNV (4000 cm⁻¹ à 8000 cm⁻¹)



LB (dérivée seconde) et SNV (4000 cm⁻¹ à 8000 cm⁻¹)



SG1 (dérivée 1er) et SNV (4000 cm⁻¹ à 8000 cm⁻¹)



SG2 (dérivée 2e) et SNV (4000 cm⁻¹ à 8000 cm⁻¹)

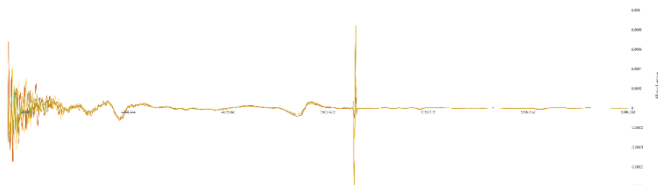


Figure 26. Différents prétraitements appliqués aux spectres infra-rouges proche

Analyse :

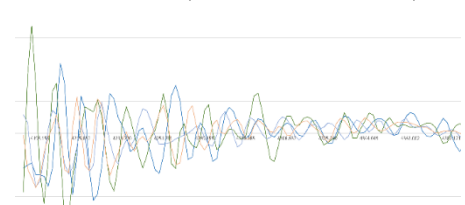
L'utilisation de la SNV permet de centrer tous les spectres sur l'axe des abscisses, supprimant ainsi les différences d'intensité.

Les méthodes LB suivie de la SNV puis SG d'ordre 1 suivie de la SNV et SG d'ordre 2 suivie de la SNV permettent d'obtenir un nombre plus important de pics spectraux surtout dans la gamme spectrale allant de 4000 à 4400 cm⁻¹. Ces pics pourraient alors apporter des informations supplémentaires pour différencier nos deux essences.

SG1 et SNV (4000 cm⁻¹ à 4200 cm⁻¹)



SG2 et SNV (4000 cm⁻¹ à 4200 cm⁻¹)



On observe également que la méthode SG d'ordre 1 suivie de la SNV aboutit à un spectre de bonne qualité, tout en apportant un nombre plus important de pics spectraux. On pourrait alors supposer que ce prétraitement soit le plus efficace pour nos prédictions.

f- Choix des gammes spectrales

Les spectres NIR ont été enregistré sur des nombres d'ondes allant de 4000 à 28490 cm^{-1} . Le reste du spectre concernant la gamme allant au-delà de 8000 cm^{-1} n'apportant aucune information sur les liaisons des molécules des essences, risque de fausser la prédiction, nous avons donc choisis de supprimer cette gamme.

Sur nos spectres prétraités, utilisant les méthodes LB/SNV, SG1/SNV et SG2/SNV, on retrouve la formation d'un pic spectrale au niveau de plage de nombre d'onde 5500 cm^{-1} et 5617 cm^{-1} qui correspond aux vibrations de déformation et d'élongation O-H de la molécule d'eau (H_2O).

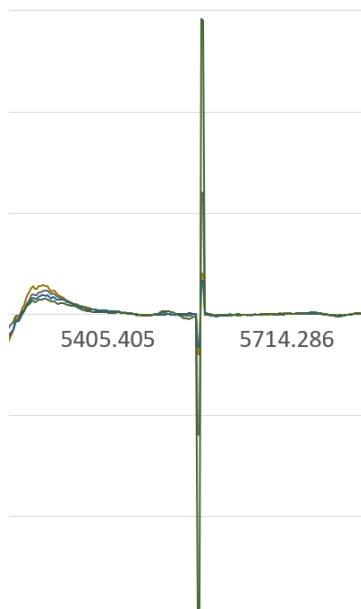


Figure 27. Pic d'absorption des vibrations de la molécule d'eau avec méthode LB/SNV

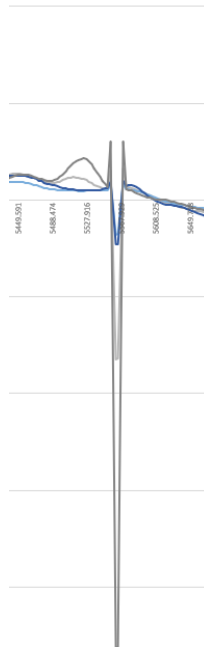


Figure 28. Pic d'absorption des vibrations de la molécule d'eau avec méthode SG1/SNV

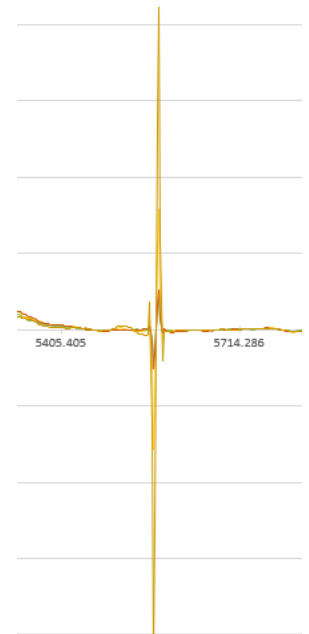


Figure 29. Pic d'absorption des vibrations de la molécule d'eau avec méthode SG2/SNV

On voit que l'intensité d'absorbance de ce pic semble dépendre de l'échantillon considéré quelque soit l'essence de bois. On en déduit qu'il s'agit de l'humidité étrangère à la composition du bois. Ce paramètre variant en fonction de l'échantillon (endroit où le spectre a été pris), indépendamment de l'essence considérée (c'est-à-dire qu'il peut être à la fois faible ou élevé pour un spectre de même essence) risque de réduire la précision du modèle de réseau de neurones.

C-2 Sélection de variables

Les signaux spectraux sont de nature continue et présentent un très grand nombre de variables (1191 variables correspondant à l'ensemble des longueurs d'onde du spectre, après imputation des bandes spectrales de l'eau). Certains modèles comme les réseaux de neurones ne sont pas appropriés (plus de poids à entraîner). Il est donc courant d'appliquer des méthodes de sélection de variables. Ces méthodes ont plusieurs avantages :

- Permet de conserver uniquement les longueurs d'onde ou les plages de longueur d'onde informatives et non bruitées.

- Cela simplifie le modèle et peut le rendre plus performant (augmenter la justesse) du fait de l'élimination de zones ne jouant aucun rôle dans la discrimination des deux espèces.

En effet sans sélection de variables le modèle de réseau de neurones aura tendance à classer les espèces aléatoirement. Ceci se vérifie en observant la courbe d'entraînement de la justesse (ou accuracy en anglais) en fonction du nombre d'apprentissage sur toutes les données (nombre d'époche). On retrouve alors de nombreuses fluctuations dans la courbe d'entraînement prouvant un classement aléatoire des deux espèces. On note une augmentation de la justesse au cours de l'entraînement, montrant que le modèle parvient à modifier ses poids pour discriminer les deux espèces au fur et à mesure de l'aléatoire.

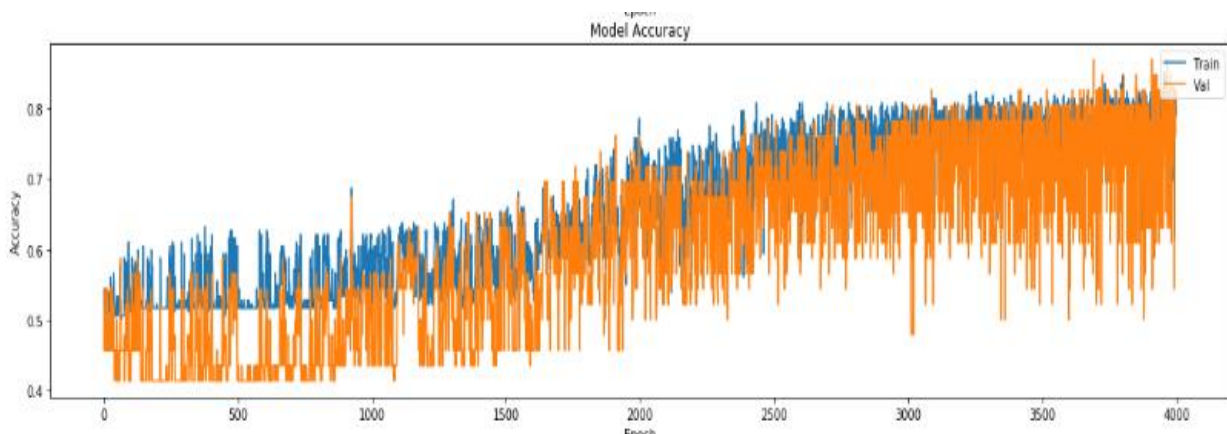


Figure 30. Courbe d'entraînement sans sélection de variables : classement des espèces aléatoirement

a- Utilisation du SelectKBest

La technique SelectKBest permet de sélectionner les k variables de nos données (variables x) dont le score de test de dépendance avec les variables cibles (variable y) est le plus élevé.




Nous utilisons le test de dépendance f_{classif} qui utilise le test d'ANOVA.

Admettons que nous voulons savoir si la boisson bue par une personne affecte son temps de réaction.

On réalise donc une expérience avec trois groupes de personnes. Le premier groupe a de l'eau pour boire, le deuxième un jus d'orange et le troisième un café. On cherche à savoir s'il y a une différence au niveau du temps de réaction entre les groupes.

L'Hypothèse nulle H_0 annonce que pour tous les groupes le temps de réaction est identique. Donc la moyenne de chaque groupe est identique.

Expérience 1 :




		
10	11	12
12	14	13
18	19	17
24	23	25
36	38	37
Groupe 1	Groupe	Groupe 3

Nous pouvons voir qu'il y a une forte variation à l'intérieur de chaque groupe. Certaines personnes sont plus rapides que d'autres.

Mais tous les groupes ont l'air d'être plutôt similaires.

On peut donc conclure que la plus grande différence est due aux personnes, et la boisson n'a pas réellement d'impact sur le temps de réaction.

Expérience 2 :

		
29	17	10
29	18	11
30	19	12
31	19	12
31	20	13
Groupe 1	Groupe 2	Groupe 3

Nous pouvons voir qu'il y a une forte variation entre chaque groupe.

Et à l'intérieur des groupes la variation est faible.

On peut donc conclure que la plus grande différence est due à la boisson.

Nous cherchons à savoir d'où vient la variance totale de la population :

- De la variance entre les groupes
- De la variance dans les groupes






On calcul alors un ratio :
$$F = \frac{\text{variation entre les groupes}}{\text{variation dans les groupes}}$$

Le résultat d'une ANOVA nous donne $F(2,12) = 4,27$

avec une probabilité $p = 0.04$, avec de tels résultats nous rejetons l'hypothèse H_0 . Avec (2,12 les degrés de libertés de la variance entre les groupes et dans les groupes).

Le test d'ANOVA ne fonctionne pas seulement quand on a 3 ou plus groupes, mais aussi quand nous avons de multiples variables.

Expérience 3 :

			
	30, 31, 31, 32, 32	28, 30, 27, 29, 32	25, 26, 25, 28, 29
	31, 31, 33, 35, 30	29, 30, 28, 29, 31	28, 30, 27, 26, 27
	Groupe 1	Groupe 2	Groupe 3

Par exemple, si je teste trois groupes pour leur temps de réaction le matin et un autre lot de 3 groupes le soir.

Une analyse de la variance peut nous dire :

-S'il y a une différence significative entre les trois boissons.

-Si l'heure de la journée, apporte une différence significative.

-Et finalement, s'il y a des interactions, par exemple le café pourrait avoir plus d'effet le soir que le matin.

En rapportant ce test à notre problème nous pouvons déterminer en regroupant la caractéristique numérique par le vecteur cible, les moyennes pour chaque groupe et voir si elles sont significativement différentes.

Les k longueurs d'onde obtenant le meilleur score de dépendance avec les variables cibles (variable y).

	4000	4001	...	8000 (nm)
Sapin (0)	30, 31 ... 32, 32	28, 30 ... 29, 32	...	25, 26 ... 28, 29
	30, 31 ... 32, 32	30, 31 ... 32, 32		30, 31 ... 32, 32
Epicéa (1)	31, 31 ... 35, 30	29, 30 ... 29, 31	...	28, 30 ... 26, 27
	30, 31 ... 32, 32	30, 31 ... 32, 32		30, 31 ... 32, 32

5- Optimisation et Résultats

A- Les hypers-paramètres

Après l'écriture des lignes de codes fournissant la structure des modèles, de KNN et de réseau de neurones, la prochaine étape consiste à les améliorer afin d'obtenir les meilleures performances. On procède alors aux réglages des hyperparamètres de ces modèles.

Le modèle K-Nearest-Neighbours possède les hyperparamètres suivants :

- K : nombre de voisins à prendre en compte lors de la discrimination du spectre.
- Les poids (weights) : la fonction de poids utilisée dans la prédiction.
 - « Uniforme » : poids uniformes. Tous les points de chaque quartier sont pondérés de manière égale.
 - « Distance » : poids des points déterminés par l'inverse de leur distance. Dans ce cas, les voisins plus proches d'un point de requête auront une plus grande influence que les voisins qui sont plus éloignés.
- Metric : La métrique de distance à utiliser pour calculer les distances entre le point à discriminer et les points classés
 - « Euclidienne », « Hemming », « Manhattan », « Minkowski ».
- Et bien d'autres paramètres, mais ces trois sont les plus importants.

Le Réseau de neurone possède les hyperparamètres suivants :

- Nombre d'épochs : Nombre de fois où l'ensemble de données d'entraînement est affiché au réseau pendant l'entraînement.
- Taille du Lot : Le nombre de motifs affichés sur le réseau avant la mise à jour des poids.
- L'algorithme d'optimisation : Permet d'entraîner le réseau de neurones en optimisant la fonction de coût qui est définie comme la moyenne de la perte, entre la valeur prédite et la valeur réelle. À l'aide d'algorithmes d'optimisation, nous minimisons la valeur de la fonction de coût en mettant à jour les valeurs des paramètres entraînaibles W et b.
 - "SGD", "RMSprop", "Adagrad", "Adadelta", "Adam", "Adamax", "Nadam".
- Les fonctions d'activations : Contrôle la non-linéarité des neurones individuels.
 - "softmax", "softplus", "softsign", "relu", "tanh", "sigmoid", "hard sigmoid", "linear"
- Taux d'apprentissage : Contrôle la quantité de mise à jour du poids à la fin de chaque lot.
- L'élan : Contrôle la mesure dans laquelle la mise à jour précédente doit influencer la mise à jour actuelle du poids. Nous allons essayer des valeurs d'élan de 0,2 à 0,8.
- Taux des abandons : Permet de réguler et de limiter le surajustement et d'améliorer les capacités du modèle à se généraliser. Nous allons essayer les pourcentages d'abandons entre 0,0 et 0,9.
- Initialisation des poids : Nous utilisons de petites valeurs aléatoires sur chaque couche.
 - "uniforme", "lecun_uniform", "normal", "zero", "glorot_normal", "glorot_uniform", "he_normal", "he_uniform"
- Nombre de neurones de la couche cachée : Contrôle la capacité de représentation du réseau.
 - Nous utilisons la gamme suivante : (1, 5, 10, 15, 20, 25, 30)

B- Le validation set

Si on règle notre modèle en optimisant ses performances sur le test set, alors on ne pourra plus utiliser les données du test set pour faire l'évaluation finale de notre modèle. Pourquoi ?

Pour évaluer un modèle, il faut le soumettre à des données qu'il n'a jamais vues. Or si on règle notre modèle sur les données du test set, et qu'on l'évalue sur ces mêmes données, alors le modèle risque de prédire la classe du spectre sans aucune difficulté, puisqu'il est réglé dessus.

Donc il est impératif de découper une troisième section dans notre Dataset : la Validation set.

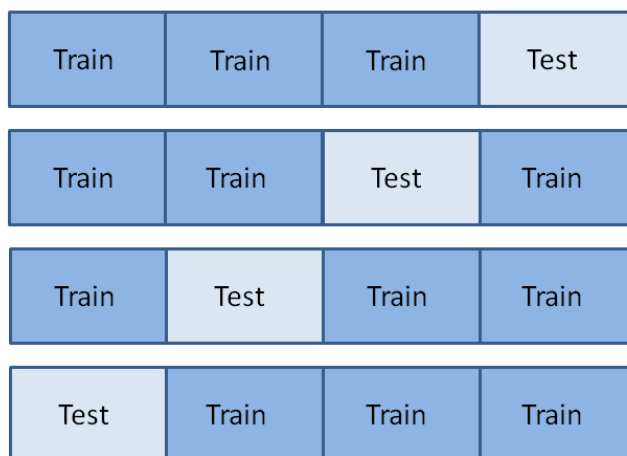
Cette section nous permet de trouver les meilleurs hyperparamètres, en gardant de côté les données du test-set qui va nous servir à évaluer le modèle paramétré.

C- Randomized-Search-CV

Randomized-Search-CV nous permet de trouver le modèle avec les meilleurs paramètres en comparant les différentes performances d'un nombre k de modèles utilisant des paramètres choisis aléatoirement, dans notre gamme d'hyperparamètre. Elle fonctionne en utilisant la technique de cross-validation.

La cross validation consiste à entraîner puis valider notre modèle sur plusieurs découpes possible du trainset.

Par exemple découpons notre train-set en 4 parties, on entraîne notre modèle sur les 3 premières parties puis on le valide sur la 4ème partie. Ensuite on sélectionne un validation-set sur un autre quart du train-set puis on entraîne le modèle avec le reste du train-set. On répète l'opération pour toutes les configurations possibles. A la fin on fait la moyenne des 4 scores obtenus puis on le compare au résultat d'un autre modèle.



Modèle 1	Modèle 2
0,96	0,78
0,91	0,95
0,84	0,72
0,72	0,87
Moyenne = 0,855	Moyenne = 0,83

Figure 31. Exemple de Cross-validation de 25 % sur le Trainset

D- Recherche du réseau de neurone optimal

a- Structure et hyperparamètres optimaux

La structure du réseau correspond à :

- Une couche d'entrée de 10 neurones, correspondant aux 10 longueurs d'ondes sélectionnés par le Select-K- Best.
- Une couche intermédiaire composée de 30 neurones.
- Une couche finale composé d'un seul neurone et utilisant la fonction d'activation sigmoïde. Ce dernier annoncera nos prédictions comme expliqué dans la partie 3-B-a.

Les hyperparamètres sélectionnés par Randomized-Search-CV sont les suivants :

- weight_constraint = 25
- optimizer= Nadam
- learn_rate = 0.1
- init_mode = 'he_normal'
- dropout_rate = 0,3
- batch_size = 1000
- activation = 'tanh'

b- Optimisation du nombre d'époques

Le nombre d'époque, c'est-à-dire le nombre de fois que l'échantillon passe intégralement dans le réseau, influe sur la durée d'entraînement. Logiquement, si ce nombre augmente, la précision augmente. Cependant, il se peut que l'erreur se mette aussi à augmenter en même temps que la précision. C'est le phénomène de surapprentissage. Il faut donc trouver le juste milieu.

Si l'on entraîne notre modèle à reconnaître sur l'image, un chien. Mais nous ne possédons dans notre database, que des photos d'une race de chien spécifique (prenons un cocker par exemple). Maintenant à force d'entraîner notre modèle, sur ces images, le modèle se spécialise dans la reconnaissance de cocker et non dans celle d'un chien.



Figure 32. Courbe d'entraînement et de Validation pour illustrer l'overfitting

c- Les Résultats

Le programme est testé parmi 2 classes : Le Sapin (classe 1), et l'Epicéa (classe 0). Chaque classe comporte 145 spectres de Sapin, et 141 spectres d'Epicéa. Pour la phase d'entraînement, on utilise 115 spectres d'Epicea, et 113 spectres de Sapin. Pour la phase de test on utilise 32 spectres de Sapin et 26 spectres d'Epicea.

Voici la matrice de confusion correspond Trainset :

	EPICEA	SAPIN
EPICEA	100	15
SAPIN	21	92

Les résultats sont présentés dans cette matrice de confusion. La colonne de gauche représente les classes réelles et la ligne supérieure les classes prédites. Dans la phase d'entraînement, on observe qu'il y a 192 spectres qui sont prédits correctement sur 228 ce qui donne une Justesse de 0,84. Une Précision de 0,83%, un Recall de 0,87% et un score f1 de 0,84%, concernant la classe Epicea. Et une précision de 0,86 %, un Recall de 0,81% et un score f1 de 0,84 % concernant la classe Sapin.

Voici la matrice de confusion correspondant au Testset :

	EPICEA	SAPIN
EPICEA	25	1
SAPIN	1	31

Dans la phase de test, on observe qu'il y a 56 spectres qui sont prédits correctement sur 58 ce qui donne une Justesse de 0,97%. Une Précision de 0,96%, un Recall de 0,96% et un score f1 de 0,96%, concernant la classe Epicea. Et une précision de 0,97 %, un Recall de 0,97% et un score f1 de 0,97 % concernant la classe Sapin.

E- Recherche du K-Nearest-Neighbors optimal

a- Les hyperparamètres optimaux

Les hyperparamètres sélectionnés par Randomized-Search-CV sont les suivants :

Weights = 'uniform' P = 2

K (N_neighbors) = 4

Metric = 'minkowski'

Leaf_size = 32

Algorithm = 'auto'

b- Les Résultats

Le programme est testé parmi 2 classes : Le Sapin (classe 1), et l'Epicéa (classe 0). On prend le même Testset et Trainset que le Dataset précédant.

Voici la matrice de confusion correspondant au Trainset :

	EPICEA	SAPIN
EPICEA	111	4
SAPIN	22	91

Les résultats sont présentés dans cette matrice de confusion. La colonne de gauche représente les classes réelles et la ligne supérieure les classes prédites. Dans la phase d'entraînement, on observe qu'il y a 202 spectres qui sont prédits correctement sur 228 ce qui donne une Justesse de 89%. Une Précision de 83%, un Recall de 97% et un score f1 de 90%, concernant la classe Epicea. Et une précision de 0,96 %, un Recall de 81% et un score f1 de 88 % concernant la classe Sapin.

Voici la matrice de confusion correspondant au Testset :

	EPICEA	SAPIN
EPICEA	26	0
SAPIN	2	30

Dans la phase de test, on observe qu'il y a 56 spectres qui sont prédits correctement sur 58 ce qui donne une Justesse de 0,97%. Une Précision de 93%, un Recall de 100% et un score f1 de 96%, concernant la classe Epicea. Et une précision de 100%, un Recall de 94% et un score f1 de 97 % concernant la classe Sapin.

Comparaison des deux modèles :

En comparant ces deux modèles on observe que durant la phase d'entraînement, le réseau de neurone à plus de facilité à reconnaître les spectres d'Epicea que ceux du Sapin à l'inverse du KNN. Cependant dans la phase de test, on obtient pratiquement les mêmes résultats pour les deux modèles.

En augmentant le nombre de spectres dans mon Dataset réduisant ainsi le surentrainement, nous pouvons obtenir des résultats encore meilleurs. Mais d'une manière générale, nous pouvons conclure que ces deux modèles sont très fiables.

IV- CONCLUSION

Ces 2 mois de stage passés au FCBA, ont été une expérience professionnelle très positive. Ce stage m'a réconforté dans mon choix de travailler dans les sciences de l'ingénieur.

Il m'a permis de me confronter à de nouveaux problèmes, tout en innovant pour les résoudre. (Que ce soit dans la recherche des bonnes informations sur internet, ou l'utilisation de mes propres idées).

J'ai apprécié particulièrement ce travail de recherche en autonomie, et surtout communiquer les avancés du programme aux experts ingénieurs, tout en recevant leurs conseils.

J'ai pu découvrir les avantages et les inconvénients du métier de programmeur et plus globalement celui de chercheur. Il faut se poser ses propres objectifs pour avancer le plus rapidement possible dans le temps imparti. (Il m'est arrivé de perdre des journées entières à chercher des informations pour me rendre compte qu'elles ne me servaient à rien.)

Il faut garder à l'esprit que faire de la programmation correspond à mon sens à 70% de recherches internet (Recherches sur le choix du type de modèle, recherches sur les meilleurs prétraitements, recherches sur leurs fonctionnements, et finalement des recherches de solutions pour résoudre des erreurs de codes (parfois très long)) et 30% de codes (ajustement de la structure d'un réseau de neurone et d'un KNN à nos données : les spectres infra-rouges proches, etc...)

Je me sens maintenant plus à l'aise avec la programmation en Python (sur les bibliothèques utilisées) et ces outils que je trouve passionnant.

J'espère mettre à profit les connaissances acquises durant ce stage, dans ma future carrière et renouveler cette expérience avec FCBA ou d'autres établissements.

Je tiens à remercier encore une fois le personnel du FCBA et en particulier Mr. Patrice Garcia qui m'a permis de travailler sur cet axe de recherche passionnant !

BIBLIOGRAPHIE (Documents principaux)

1- FCBA | Votre outil technologique pour les filières forêt, bois, construction et ameublement.
<https://www.fcba.fr>

2- Conception d'un système de reconnaissance d'espèce de bois
[file:///C:/Users/louis/AppData/Local/Packages/microsoft.windowscommunicationsapps_8wekyb3d8bawe/LocalState/Files/S0/134/Attachments/paper2\[2191\].pdf](file:///C:/Users/louis/AppData/Local/Packages/microsoft.windowscommunicationsapps_8wekyb3d8bawe/LocalState/Files/S0/134/Attachments/paper2[2191].pdf)

3- Différence entre Intelligence Artificielle, Machine Learning et Deep Learning :
<https://penseeartificielle.fr/difference-intelligence-artificielle-machine-learning-deep-learning/>

4- Principe de la propagation avant et de la rétropropagation :
http://galaxy.agh.edu.pl/~vlasi/AI/backp_t_en/backprop.html

5- Chimio métrie spectroscopie infrarouge :
http://math.agrocampus-ouest.fr/infoglueDeliverLive/digitalAssets/41740_Poly_Bertrand_2008.pdf

6-Traitement du signal Analyse de signaux avec transformée de fourrier :
<https://www.youtube.com/watch?v=Bldjvw9YZhk&t=592s>

7-Méthodes de Chimiométrie et machine learning (site très utilisé) :
<https://ondalys.fr/ressources-scientifiques/methodes-de-machine-learning/#DeepLearning>

7-Mathématiques appliquées et traitement du signal pour l'évaluation de la dégradation de la biomasse :
<https://www.archives-ouvertes.fr/tel-01482728/document>

8-Diffusion et réflexion :
<http://forumbachelor.free.fr/gma/GMA2/index.php?page=10#:~:text=Lorsqu%27une%20onde%20acoustique%20rencontre,est%20le%20phénomène%20de%20diffusion.>

9-Scikit-learn site :
-Choix du model : https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
-K-NeighborsClassifier : <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
-Select-K-Best: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
-Randomized-Search-CV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

10-Site de résolution d'erreur python Stackoverflow :
<https://stackoverflow.com>

11- Thèses de FCBA sur :
-Traitement du signal
-Réseau de neurones

ANNEXE

Figure 33. Principales bandes d'absorption des l'infrarouge proche dans l'étude de la matière organique.

Nombre d'onde cm^{-1}	Attribution	Source
4063	vibration d'élongation C-H + vibration de déformation C-H	Cellulose
4235	vibration de déformation O-H ou C-H + vibration d'élongation C-H ou C-H ₂	Cellulose
4268	vibration d'élongation C-H + vibration de déformation C-H et 2 nd harmonique C-H	Cellulose
4280	vibration d'élongation C-H + vibration de déformation C-H ₂	Lignine
4283	vibration d'élongation C-H + vibration de déformation C-H	Cellulose et hémicellulose
4296-4288	vibration d'élongation C-H + vibration de déformation C-H	Hémicellulose
4365	vibration d'élongation C-O + H-O ou vibration de flexion C-H ₂ + vibration d'élongation C-H ₂	Cellulose
4401	vibration d'élongation C-H + vibration de déformation C-H	Hémicellulose
4404	vibration d'élongation C-H ₂ + vibration de déformation C-H ₂	Cellulose et hémicellulose
4411	vibration d'élongation H-O + vibration d'élongation C-O	Lignine
4546	vibration d'élongation C-H + vibration d'élongation C=O	Lignine
4686	vibration d'élongation C-H + vibration d'élongation C=O	Hémicellulose
4739	vibration de déformation O-H + vibration d'élongation H-O	Cellulose
4780-4760 ; 4808	vibration de déformation O-H et C-H + vibration d'élongation O-H	Cellulose
5051 ; 5220- 5150	vibration de déformation O-H + vibration d'élongation H-O de H ₂ O	Eau
5495 ; 5464	vibration de déformation O-H + 2 nd vibration d'élongation harmonique C-O	Cellulose
5577 ; 5593	Premier vibration d'élongation harmonique C-H	Cellulose
5583 ; 5795	Premier vibration d'élongation C-H	Lignine
5618	Premier vibration d'élongation harmonique C-H ₂	Cellulose
5816	Premier vibration d'élongation harmonique C-H	Cellulose /hémicellulose/li gnine
5800 ; 5848 ; 5865	Premier vibration d'élongation harmonique C-H	Hémicellulose
5900 ; 5939	Premier vibration d'élongation C-H	Lignine
5950	Premier vibration d'élongation C-H	Hémicellulose
5980 ; 5974 ; 5963 ; 5978	Premier vibration d'élongation C-H	Lignine
6003	Premier vibration d'élongation C-H	Hémicellulose

PROGRAMME 1 : PRE-TRAITEMENTS ET RESEAU DE NEURONE

```
##### PROGRAMME RESEAU DE NEURONE #####

##### IMPORT #####
import matplotlib.pyplot as plt

import numpy as np
import pandas as pd
import pandas
import os
import statistics
import math
import datetime
import scipy
from scipy.signal import savgol_filter
import tensorflow as tf
from keras.wrappers.scikit_learn import KerasClassifier
from keras.callbacks import TensorBoard
from keras.models import Sequential
from keras.layers import Activation, Dense
from keras.constraints import max_norm
from keras.layers import Dropout
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import learning_curve
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import RandomizedSearchCV

##### OPTION PREPROCESSING #####

PREPROCESSING = "Oui"

#####CHARGE DU DATAFRAME CORRESPONDANT A LA DATABASE DE LA TRANSMITANC ET CREATION DU TRAINSET ET DU TESTSET

def dataframe():
    #print(os.listdir())
    no = pd.read_excel("Nombre_d'onde.xlsx")
    data = pd.read_excel("DATABASE_Sapin_Epicea_Ref1.xlsx")
    data = data.set_index("File Name")
    #data      #Retirer le # pour voir le dataset
    print('Data set:', data.shape)
    trainset, testset = train_test_split(data, test_size=0.2, random_state=1)

    print("Train set:" ,trainset.shape)
    print("Test set:", testset.shape)

    #trainset['Essence'].value_counts()
    #testset ['Essence'].value_counts()

    #trainset      #Retirer le # pour voir le trainset ou le testset
    #testset
    return data, no, trainset, testset
```

```
##### FONCTIONS DE PRE-PROCESSING DES DONNEES #####
```

```
#ENCODAGE DES VARIABLES
```

```
def encodage (data):
    code1 = {'Bout':0,
             'Chant':1,
             'Face':2}
    data['FACES'] = data['FACES'].map(code1)
    code2 = {'Epicea':0,
             'Sapin':1}
    data['Essence'] = data ['Essence'].map(code2)

    return data
#encodage (data)
```

```
#APPLICATION DE LA FONCTION -LOG POUR PASSER DE LA TRANSMITTANCE A L'ABSORBANCE:
```

```
def trans_abs(data):
    abs = pd.DataFrame((np.log10(data.iloc[0: , 2:].values))*-1)
    data[data.columns[2:]] = abs[abs.columns[0:2151]].values

    return data
#trans_abs(data)
```

```
#SG1
```

```
def SG1(data, no):
    for m in range (0 , len(data)):
        x = no.values
        y = data.iloc[m , 2: ]
        yhat = scipy.signal.savgol_filter(y, 5, 3)

        yp = np.zeros(2150)

        for i in range (0, 2150):
            yp[i] = (yhat[i+1] - yhat[i]) / (x[i+1] - x[i])

        data.iloc [m, 2:2152] = yp
    return data
# SG1 (data)
```

```
#NORMALISATION DES DONNEES DE LA DATABASE:
```

```
def normalisation (data):
    for i in range (0 , len(data)):
        normalisation = (data.iloc[i , 2: ] - (statistics.mean(data.iloc[i , 2: ]))) / (statistics.stdev(data.iloc[i , 2: ]))
        data.iloc[i, 2: ] = normalisation.values
    return data
#normalisation (data)
```

```
#IMPUTER 2
```

```
def imputer (data):
    data = data.drop(data.columns[2152], axis=1)
    data = data.drop(data.columns[1420:1480], axis=1)
    data = data.drop(data.columns[2:902], axis=1)

    return data
#imputer (data)
```

```

#INVERSE L'ORDRE DES NOMBRES D'ondes
def invers_axes(data):
    data = data.reindex(columns=data.columns[::-1])
    return (data)
#invers_axes(data)

##### FONCTION QUI FAIT APPELLE AU DIFFERENTS PREPROCESSING #####

def preprocessing (data, no):
    data = encodage (data)
    print ("encodage terminé")
    data = trans_abs(data)
    print("transmittance to absorbance terminé")
    data = SG1(data, no)
    print ("SG1 terminé")
    data = normalisation (data)
    print ("normalisation terminée")
    data = imputer (data)
    print ("imputation terminée")
    data = invers_axes(data)
    print ("inversion des axes terminée")

    X = data.drop('Essence', axis=1)
    y = data['Essence']

    print(y.value_counts())

    return X, y

##### SAUVEGARDE DES DONNEES PRE-TRAITEES #####

def sauvegarde (X_train, y_train, X_test, y_test):
    X_train.to_pickle('X_trains.pkl')
    y_train.to_pickle('y_trains.pkl')
    X_test.to_pickle('X_test.pkl')
    y_test.to_pickle('y_test.pkl')

##### LECTURE DES DONNEES PRE-TRAITEES #####

def load ():
    X_train = pd.read_pickle('X_trains.pkl')
    y_train = pd.read_pickle('y_trains.pkl')
    X_test = pd.read_pickle('X_test.pkl')
    y_test = pd.read_pickle('y_test.pkl')

    print ("X_train:", X_train.shape)
    print ("y_train:", y_train.shape)
    print("X_test:", X_test.shape)
    print("y_test:", y_test.shape)

    return (X_train, y_train, X_test, y_test)

```

```

##### PROCEDURE D'EVALUATION DU RESEAU DE NEURONE #####

log_dir = "logs" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

def evaluation_dl (model, X_train, y_train, X_test, y_test):
    model.fit(X=X_train, y=y_train , kerasclassifier__callbacks=[tensorboard_callback])

    ypred_train = model.predict(X_train)
    ypred_train = [1 if y>= 0.5 else 0 for y in ypred_train]
    ypred_test = model.predict(X_test)
    ypred_test = [1 if y>= 0.5 else 0 for y in ypred_test]

#I-Evaluation du model sur le train data set

#Matrice de confusion
print('Confusion Matrix trainset: \n', confusion_matrix(y_train, ypred_train))

#Tableau de Score
print (classification_report(y_train, ypred_train))
print ('Accuracy trainset: ', accuracy_score(y_train, ypred_train))

#II- Evaluation du model sur le test data set
#Matrice de confusion
print('Confusion Matrix testset: \n', confusion_matrix(y_test, ypred_test))

#Tableau de Score
print (classification_report(y_test, ypred_test))
print ('Accuracy testset: ', accuracy_score(y_test, ypred_test))

##### SELECTION DES VARIABLES #####

selection = SelectKBest(f_classif, k=10)

##### LE MODEL DE DEEP LEARNING #####

def create_model ( activation = 'tanh', dropout_rate = 0.3, init_mode = 'he_normal', weight_constraint =25, optimizer= 'Nadam',
learn_rate = 0.1 , momentum=0, input_dim = 10, neurons=30):

    #Creat the model
    md = Sequential ()
    md.add(Dense (neurons,
                    input_dim=input_dim, kernel_initializer=init_mode, activation=activation, kernel_constraint=max_norm(weight_constraint)))
    md.add(Dropout(dropout_rate))
    md.add(Dense(1, kernel_initializer=init_mode, activation= 'sigmoid'))

    #Compile model

    md.compile(loss= 'binary_crossentropy',
                optimizer=optimizer,
                metrics=['accuracy'])

    return md

#model
model = make_pipeline(selection, KerasClassifier(build_fn= create_model, batch_size=1000, epochs=10))

```

```

##### REGLAGE DES HYPERS PARAMS #####

def best_hyper_params (X_train, y_train, X_test, y_test):

##### Choisissez Les plages d'hyperparamètres à régler
activation = ['sigmoid', 'relu', 'tanh', 'hard_sigmoid', 'linear']
momentum = [0.0, 0.1, 0.4, 0.6, 0.8, 0.9]
learn_rate = [0.001, 0.01, 0.1, 0.2, 0.3]
dropout_rate = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
weight_constraint= [1, 5, 10, 15, 20, 25, 30]
neurons = [1, 5, 10, 15, 20, 25, 30]
init =['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']
optimizer = ['SGD' , 'RMSprop', 'Adagrad', 'Adadelata', 'Adam' , 'Adamax' , 'Nadam']

##### Choisissez Les plages d'hyperparamètres à régler
epochs = [3000]
batch_size = [1000 , 2000]
param_grid = dict(kerasclassifier__epochs=epochs) # Ajoutez d'autres paramètres à régler (ex: kerasclassifier__act
kerasclassifier__momentum ...)

#####
grid = RandomizedSearchCV(model, param_grid, scoring = 'accuracy', cv=4, n_iter=50)
grid_result = grid.fit(X_train, y_train)

#####
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
#####
y_pred = grid.predict(X_test)

return grid.best_estimator_

##### MAIN FUNCTION #####

def main ():
    if PREPROCESSING == "Oui":
        data, no, trainset, testset = dataframe ()
        X_train, y_train = preprocessing(trainset, no)
        X_test, y_test = preprocessing(testset, no)
        #sauvegarde (X_train, y_train, X_test, y_test)
        print (evaluation_dl (best_hyper_params (X_train, y_train, X_test, y_test),X_train, y_train, X_test, y_test ))

    else:
        X_train, y_train, X_test, y_test = load ()
        print (evaluation_dl (best_hyper_params (X_train, y_train, X_test, y_test),X_train, y_train, X_test, y_test ))

if __name__ == "__main__" :
    main()

```

Output :

```

Confusion Matrix trainset:
[[99 16]
 [19 94]]

```

	precision	recall	f1-score	support
0	0.84	0.86	0.85	115
1	0.85	0.83	0.84	113
accuracy			0.85	228
macro avg	0.85	0.85	0.85	228
weighted avg	0.85	0.85	0.85	228

```

Accuracy trainset: 0.8464912280701754
Confusion Matrix testset:
[[25  1]
 [ 1 31]]

```

	precision	recall	f1-score	support
0	0.96	0.96	0.96	26
1	0.97	0.97	0.97	32
accuracy			0.97	58
macro avg	0.97	0.97	0.97	58
weighted avg	0.97	0.97	0.97	58

```

Accuracy testset: 0.9655172413793104
..

```


PROGRAMME 2 : K-NEAREST-NEIGHBORS

```
##### PROGRAMME K-NEAREST NEIGHBORS #####
##

##### IMPORT #####
import matplotlib.pyplot as plt
import sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import f1_score, confusion_matrix, classification_report
from sklearn.model_selection import learning_curve
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.pipeline import make_pipeline
import numpy as np
import pandas as pd

##### LECTURE DES DONNEES PRE-TRAITEES #####

def load():
    X_train = pd.read_pickle('X_trains.pkl')
    y_train = pd.read_pickle('y_trains.pkl')
    X_test = pd.read_pickle('X_test.pkl')
    y_test = pd.read_pickle('y_test.pkl')

    print("X_train:", X_train.shape)
    print("y_train:", y_train.shape)
    print("X_test:", X_test.shape)
    print("y_test:", y_test.shape)

    return (X_train, y_train, X_test, y_test)

##### PROCEDURE D'EVALUATION DU MODEL DE MACHINE LEARNING #####

def evaluation(model, X_train, y_train, X_test, y_test):

    model.fit(X=X_train, y=y_train)
    ypred_test = model.predict(X_test)
    ypred_train = model.predict(X_train)

    print(confusion_matrix(y_test, ypred_test))
    print(classification_report(y_test, ypred_test))

    print(confusion_matrix(y_train, ypred_train))
    print(classification_report(y_train, ypred_train))

    N, train_score, val_score = learning_curve(model, X_train, y_train, cv=4, scoring = 'f1',
                                              train_sizes=np.linspace(0.1, 1, 10))

    plt.figure(figsize=(12, 8))
    plt.plot(N, train_score.mean(axis = 1), label='train score')
    plt.plot(N, val_score.mean(axis=1), label= 'validation score')
    plt.legend()

##### SELECTION DE VARIABLES #####

selection = make_pipeline(SelectKBest(f_classif, k=10))

##### LE MODEL DE DEEP LEARNING #####

KNN = make_pipeline(selection, KNeighborsClassifier())
```

```
##### REGLAGE DES HYPERS PARAMS #####
```

```
def best_hyper_params (X_train, y_train, X_test, y_test):
```

```
##### Choisissez Les plages d'hyperparamètres à régler
```

```
hyper_params = {'kneighborsclassifier__leaf_size': range(20, 35),
                 'kneighborsclassifier__n_neighbors': range(3, 5),
                 'kneighborsclassifier__p' : [1, 2],
                 'kneighborsclassifier__weights' : ['uniform', 'distance'],
                 'kneighborsclassifier__algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute'],
                 'kneighborsclassifier__metric' : [ 'manhattan', 'minkowski'],
                 'pipeline__selectkbest__k': range (8, 11)}
```

```
#####
```

```
grid = RandomizedSearchCV(KNN, hyper_params, scoring = 'accuracy', cv=4, n_iter=500)
grid_result = grid.fit(X_train, y_train)
```

```
#####
```

```
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
#####
```

```
y_pred = grid.predict(X_test)
```

```
return grid.best_estimator_
```

```
##### MAIN FUNCTION #####
```

```
def main ():
```

```
    X_train, y_train, X_test, y_test = load ()
```

```
    print (evaluation (best_hyper_params (X_train, y_train, X_test, y_test),X_train, y_train, X_test, y_test ))
```

```
if __name__ == "__main__" :
```

```
    main()
```

Output :

```
[[26  0]
 [ 2 30]]
```

	precision	recall	f1-score	support
0	0.93	1.00	0.96	26
1	1.00	0.94	0.97	32
accuracy			0.97	58
macro avg	0.96	0.97	0.97	58
weighted avg	0.97	0.97	0.97	58

```
[[111  4]
 [ 22 91]]
```

	precision	recall	f1-score	support
0	0.83	0.97	0.90	115
1	0.96	0.81	0.88	113
accuracy			0.89	228
macro avg	0.90	0.89	0.89	228
weighted avg	0.90	0.89	0.89	228

