Utilisation de l'apprentissage automatique en remplacement des simulateurs de débitage de billots de bois

Mémoire

Vincent Martineau

Maîtrise en informatique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Utilisation de l'apprentissage automatique en remplacement des simulateurs de débitage de billots de bois

Mémoire

Vincent Martineau

Sous la direction de:

Jonathan Gaudreault, directeur de recherche
Michael Morin, codirecteur de recherche

Résumé

Déterminer les produits de sciage pouvant être obtenus suite à la transformation d'un billot dans une scierie permet une meilleure assignation des blocs de coupes et d'améliorer la planification des opérations. Bien que les simulateurs de débitage permettent d'anticiper la production associée à un billot, ils ne permettent pas de traiter un grand nombre de billots rapidement.

Nous savons que l'apprentissage automatique peut être utilisé pour la prédiction du panier de produits associé à un billot (c.-à-d. les produits de sciage contenu dans le billot). Toutefois, prédire le panier de produits correspondant à un lot de billots n'a pas encore été étudié. Dans ce contexte, nous avons développé une approche permettant d'obtenir une prédiction par lot. Nos résultats montrent un gain jusqu'à 2% supérieur à la moyenne historique d'une scierie.

Dans un second temps, les travaux précédents se concentrent sur une représentation simplifiée du billot et peu d'études ont exploré d'autres représentations du billot dans le but de prédire le panier de produits associé à chaque billot. Pour cette raison, nous avons utilisé des réseaux de neurones traitant des nuages de points obtenus par la numérisation 3D des billots. Cette approche permet une amélioration de 8% par rapport aux méthodes précédentes. Nous avons également créé une projection de la surface du billot qui performe de manière similaire aux nuages de points, tout en réduisant l'information nécessaire aux réseaux de neurones.

Par la suite, nous avons combiné les diverses approches développées précédemment dans le but d'exploiter les forces de chacune des approches. Ces travaux ont permis d'observer quelques exemples où combiner les approches donnait un gain.

Finalement, nous avons élaboré une série de fonctions de pertes. Celles-ci ont permis aux réseaux de neurones d'obtenir des gains de 15% par rapport aux approches initialement proposées, devenant ainsi l'état de l'art.

Abstract

Determining the sawmill output that can be obtained as a result of the transformation of a log allows for better allocation of cutting blocks and improves operational planning. Although sawing simulators make it possible to anticipate the production associated with a log, they do not allow processing many logs quickly.

We know that machine learning can be used to predict the basket of products associated with a log (i.e., the lumber contained in the log). However, predicting the basket of products corresponding to a batch of logs has not yet been studied. In this context, we have developed an approach to obtain a prediction for a batch of logs. Our results show a gain up to 2% higher than the historical average of a sawmill.

In a second step, previous work focuses on a simplified representation of the log and few studies have explored other representations of the log in order to predict the basket of products. For this reason, we used neural networks that process point clouds obtained by 3D scanning of logs. This approach shows an 8% improvement over previous methods. We also created a projection of the log surface that performs similarly to point clouds, while reducing the information needed by the neural networks.

Subsequently, we combined the various approaches developed previously with the aim of exploiting the strengths of each of the approaches. This work has made it possible to observe some examples where combining approaches improves the prediction.

Finally, we developed a series of loss functions. These have allowed neural networks to achieve gains of 15% compared to the approaches initially proposed, thus becoming the state of the art.

Table des matières

Liste des tableaux

Liste des figures

À ma famille.

You can observe a lot by just watching.

Yogi Berra

Remerciements

Je tiens à remercier monsieur Jonathan Gaudreault et monsieur Michael Morin, mon directeur et mon codirecteur, de m'avoir guidé durant mes recherches. Je les remercie aussi pour la confiance qu'ils m'ont accordée tout au long du projet. Je remercie aussi tous les gens du consortium FORAC pour leur support et pour avoir rendu ce projet possible. Je les remercie aussi pour les nombreux événements qu'ils ont organisés qui m'ont permis d'en apprendre plus sur le domaine forestier.

Je souligne également le soutien de notre partenaire FPInnovations pour les outils, les données et les échangent sur le fonctionnement de la transformation du bois.

Finalement, j'aimerais aussi remercier Sophie Samson pour la révision du présent document et toute ma famille pour les encouragements.

Avant-propos

Ce travail porte sur l'utilisation de l'apprentissage machine en remplacement des simulateurs de débitage.

Le chapitre 2 présente une contribution scientifique, arbitrée par les pairs, intitulée « Quality of sawmilling output predictions according to the size of the lot - The size matters ! » (Martineau, Gaudreault et al., 2021) ayant été présentée à la conférence CIGI-Qualita21 : Conférence Internationale Génie Industriel. Les coauteurs de cet article sont Jonathan Gaudreault, Professeur titulaire au Département d'informatique et de génie logiciel, FSG Ulaval, Michael Morin, Professeur adjoint au Département d'opérations et systèmes de décision, FSA ULaval et Steve Vallerand, FPInnovations. La publication de la contribution scientifique a eu lieu le 31 mai 2021.

Le chapitre 3 est basé sur une seconde contribution scientifique intitulée « Neural Network Architectures and Feature Extraction for Lumber Production Prediction » (Martineau, Morin et al., 2021) qui a été présentée à la conférence Canadian AI 2021 à Vancouver. La différence avec le chapitre présenté dans ce mémoire est que certaines données ont été retirées du jeu de données puisqu'elles n'étaient pas valides. Pour cet article les coauteurs sont : Jonathan Gaudreault, Professeur titulaire au Département d'informatique et de génie logiciel, FSG Ulaval, Michael Morin, Professeur adjoint au Département d'opérations et systèmes de décision, FSA ULaval, Philippe Thomas, Maître de Conférences chez Université de Lorraine et Hind Bril El-Haouzi, professeure à l'Université de Lorraine. La publication de la contribution scientifique a eu lieu le 14 juin 2021.

Mes coauteurs ont contribué à ces articles par leur encadrement. Ils ont aussi contribué avec leur expertise lors des expérimentations et de la rédaction des articles. Ma contribution a été de préparer les données, de produire le code nécessaire aux expérimentations et d'exécuter les expérimentations. De plus, j'étais responsable d'effectuer la revue de littérature et la rédaction des deux articles.

Introduction

La tâche de prévoir les produits de bois résultant de la transformation des billots effectuée dans une scierie donnée est complexe. Comme le montre la figure A, la transformation d'un billot entraîne généralement la production de plusieurs produits de sciage ayant une valeur différente. Une partie de la complexité du problème repose sur le fait qu'un billot peut produire une combinaison de produits de sciage (ci-après nommé panier de produits) différents en fonction de la configuration de la scierie ou de la machinerie présente dans la scierie. Dans ce contexte, bien que l'industrie vise à maximiser la valeur d'un billot, c'est-à-dire la valeur totale des produits de sciage pour ce billot, nous savons qu'il est possible que le billot n'ait pas la même valeur pour deux scieries différentes ou deux configurations différentes de la même scierie.

Dans la pratique, une scierie de taille moyenne peut produire plus de quatre-vingts produits de sciage différents, nos expérimentations démontrent qu'il peut y avoir plus d'un millier de paniers de produits réalisables pour cette scierie.

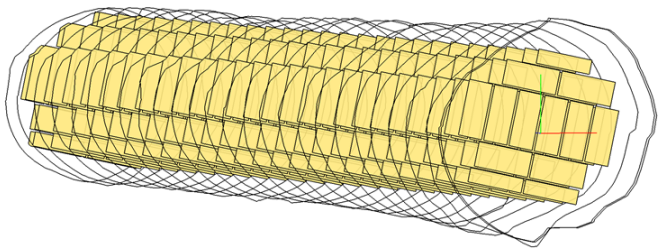

FIGURE A – Représentation d'un billot et des produits de sciage résultant de sa transformation.

Il a été démontré que de bonnes prévisions de la production peuvent générer des gains considérables pour une entreprise de produits forestiers grâce à de meilleures opportunités de planification (Morin, Gaudreault et al., 2020). On peut, par exemple, améliorer l'affectation des blocs de coupe aux scieries ou avoir une meilleure idée de la valeur d'un lot lorsque vient le temps de l'acquérir. De plus, pouvoir prédire le contenu de la production pour un ensemble de billots avant la transformation permet d'optimiser les paramètres de la scierie en conséquence. Du coup, il devient possible d'obtenir une plus grande valeur pour chacun des billots transformés à cette scierie.

Afin de prédire le contenu d'un panier de produits, les entreprises de l'industrie forestière ont accès à des simulateurs de sciage tels que RAYSAW (Thomas, 2013), Optitek (FPInnovations, 2014) et SAWSIM (HALCO, 2016).

Ces outils simulent la transformation d'un billot à partir d'une représentation virtuelle du billot et d'une description de la scierie. Cependant, les simulations de débitage demandent trop de temps pour être utilisées sur de très gros volumes de bois. Selon la configuration du simulateur, la simulation peut prendre de quelques secondes à quelques heures par billot. Un temps de simulation aussi long est inapplicable dans certains contextes décisionnels impliquant un processus d'optimisation. Par exemple, il est impraticable dans un temps raisonnable de simuler un lot de billots pour plusieurs scieries dans le but de planifier l'allocation du bois (Morin, Gaudreault et al., 2020) ou pour la décision à court terme comme la planification d'opérations (Rönnqvist, 2003).

Afin de pallier aux longs temps de simulation, des études antérieures ont montré qu'il est possible d'utiliser l'apprentissage automatique pour résoudre le problème de la prédiction du panier de produits pour un billot. Bien qu'il a été démontré qu'il était possible d'utiliser les modèles prédictifs pour améliorer la prise de décisions dans le cadre de la planification de l'allocation du bois (Morin, Gaudreault et al., 2020). Il a été déterminé que la qualité de la prédiction pouvait être améliorée. Dans ce mémoire, notre objectif est de trouver une approche permettant d'améliorer la qualité de la prédiction lors de la prédiction d'un panier de produits associé à un billot.

Actuellement, la plupart des modèles d'apprentissage automatique développés pour ce problème utilisent des fonctionnalités basées sur le savoir-faire des spécialistes de l'industrie forestière (c.-à-d. représenter les billots de manière paramétrique en utilisant le volume, le diamètre fort, le diamètre faible, la longueur, la courbure et la conicité). Bien que fonctionnels, ces modèles n'exploitent pas toutes les informations disponibles trouvées dans la représentation virtuelle 3D (nuages de points) des billots.

À notre connaissance, il existe, dans la littérature, une seule approche au problème de la prédiction du panier de produits pour un billot qui utilise une représentation des billots qui est un nuage de points 3D (Selma et al., 2018). Cette approche est basée sur l'algorithme de *Iterative Closest Point* (Chen et Medioni, 1992 ; Besl et McKay, 1992) et se concentre sur le calcul de la similarité entre les nuages de points 3D pour le traitement des k voisins les plus proches. Dans ce mémoire, nous montrons comment utiliser les réseaux de neurones ainsi qu'une variété de représentations du billot pour la prévision du panier de produits associés à un billot.

Ce mémoire est organisé de la manière suivante. Premièrement, nous passons en revue la littérature et les concepts associés dans le chapitre 1. Ce chapitre nous permet de comprendre les recherches précédentes dans le domaine et d'établir les techniques que nous allons utiliser dans ce mémoire.

Dans le chapitre 2, nous présentons les résultats d'approches d'apprentissage supervisé, inspirées des travaux de Morin, Gaudreault et al. (2020), et reproduisons les résultats de ces travaux antérieurs sur le nouveau jeu de données utilisé dans ce mémoire. De plus, nous avons remarqué que plusieurs problématiques présentes en foresterie concernent des lots de billots et non des billots uniques. Dans ce contexte, nous avons développé une approche permettant de mesurer la performance sur un lot de billots plutôt que sur les billots individuellement. Ces résultats ont mené à la présentation d'un article évalué par les pairs et présenté lors de la conférence CIGI-Qualita21 : Conférence Internationale Génie Industriel.

Le chapitre 3 présente une série d'approches utilisant les réseaux de neurones pour faire la prédiction du panier de produits associé à un billot. Dans ce chapitre, nous explorons plusieurs architectures de réseaux de neurones ainsi que plusieurs représentations du billot pouvant être utilisées par les réseaux de neurones. Dans le chapitre 4, nous présentons la comparaison des résultats présentés dans le chapitre 3 sur les réseaux de neurones à ceux des approches présentées au chapitre 2. Cette comparaison est effectuée pour les billots individuels ainsi que pour des lots de billots et ce pour toutes les approches que nous avons développées dans les deux premiers chapitres.

Les prochains chapitres du mémoire proposent et évaluent les différentes approches pour améliorer les résultats obtenus. Le chapitre 5 concerne une série d'approches où nous combinons divers algorithmes d'apprentissage automatique et représentations du billot dans le but d'obtenir une meilleure prédiction alors que, dans le chapitre 6, nous proposons puis évaluons une série d'améliorations des fonctions de pertes des réseaux de neurones. Nous avons développé une fonction de perte utilisant la valeur des billots ainsi que deux approches se basant sur la métrique F1. Ces dernières ont été basées sur les connaissances (et intuitions) acquises dans les explorations précédentes.

Enfin, nous concluons ce mémoire en présentant une série de propositions d'approches pour des recherches futures.

Chapitre 1

Concepts préliminaires

Ce chapitre expose les concepts préliminaires qui aideront à mettre le projet en contexte et à expliquer les technologies utilisées. Le chapitre se divise en trois sections. Dans un premier temps (section 1.1), nous décrivons les simulateurs de débitages en expliquant brièvement leur fonctionnement. Par la suite, nous résumerons les études précédentes qui utilisent des approches prédictives en remplacement des simulateurs de débitages (section 1.2). Finalement, la section 1.3 survole les approches d'apprentissage automatique qui sont pertinentes pour nos travaux et les approches possibles pour la préparation des données 1.4.

1.1 Simulateur de débitage

Les simulateurs de débitages sont des outils utilisés par l'industrie forestière permettant de simuler le débitage de billots virtuels dans le but d'obtenir le panier de produits qui maximise le rendement pour ce billot en fonction d'une configuration donnée d'une scierie. Obtenir le panier de produits pour un billot avant le débitage nous permet, en autre, d'assigner à quelle scierie les billots doivent être envoyés pour maximiser la valeur des billots (Todoroki et al., 1990).

Les simulateurs de débitage utilisent une représentation virtuelle des billots pour effectuer la simulation de la transformation des billots. Afin d'effectuer une simulation complète, le simulateur a besoin de la configuration des machines présentes dans une scierie. Le simulateur doit aussi avoir une liste de produits possibles pour la scierie ainsi que la valeur de chacun des produits. La valeur peut être la valeur marchande des produits, mais la valeur peut aussi être choisie en fonction de sa désirabilité pour la scierie, ex., pour forcer la production de certains produits.

Dans ce qui suit, nous survolons les diverses représentations possibles pour les billots numérisés (section 1.1.1) puis la configuration des simulateurs de débitage (section 1.1.2) pour finalement nous attaquer à la simulation du débitage des billots (section 1.1.3).

FIGURE 1.1 – Numérisation de surface d'un billot.

1.1.1 Représentation des billots

Pour un même billot, il est possible d'avoir plusieurs représentations. La représentation contenant le plus d'information provient de numériseurs permettant la tomodensitométrie. Cette technologie consiste à balayer un objet et mesurer l'absorption des rayons X par la matière. Par la suite, un ordinateur est utilisé pour reconstruire les objets en 3D qui peuvent être analysés par un ordinateur (Hsieh, 2003). Bien que cette technologie soit principalement utilisée dans le domaine médical, il est aussi possible de l'appliquer pour la numérisation des billots. L'avantage de cette technique est qu'il est possible de voir les détails sur la composition interne des billots. Cette connaissance aidera à produire de meilleures simulations de débitage et permettra de pouvoir prédire plus adéquatement le grade des produits au centre du billot (Ursella et al., 2018).

Bien que la tomodensitométrie offre des avantages, elle nécessite un équipement complexe, lent et dispendieux. Une alternative est de numériser seulement la surface du billot. Même s'il n'est alors plus possible de connaître les défauts à l'intérieur d'un billot, cette représentation est fréquemment utilisée dans les simulateurs de débitage (Todoroki et al., 1990 ; FPInnovations, 2014). Les appareils permettant de numériser la surface d'un billot utilisent généralement des lasers. On peut retrouver ces appareils autant sur la machinerie qui abat les arbres que sur les machines présentes dans la scierie.

Lors de la numérisation, une machine balaie la surface de l'arbre avec un laser et en extrait un nuage de points. Dans ce contexte, l'unité de base qui représente le billot est un ensemble de points dans un univers en trois dimensions (Levoy et Whitted, 1985). Cependant, le nombre de points est grand pour un seul billot et avant de le fournir au simulateur de débitage, il est commun de réduire le nombre de points en retirant les points apportant le moins d'information (FPInnovations, 2014). La figure 1.1 montre un billot avec une numérisation de surface.

Finalement, la dernière méthode pour représenter un billot consiste à utiliser des données paramétriques bien connues des experts de la foresterie pour représenter le billot. Évidemment,

cette approche est moins précise que les numérisations puisqu'on ne peut pas capturer toutes les subtilités d'un billot. Cependant, ces données sont largement présentes dans l'industrie et plusieurs décisions se basent sur ces données. Ces données peuvent être extraites des représentations précédentes, mais certaines de ces mesures peuvent aussi être prises sur le terrain en utilisant le savoir-faire d'employé de la foresterie. Évidemment, la qualité de la représentation dépend de la quantité de métriques utilisées et de leur qualité, mais les mesures suivantes semblent communes dans l'industrie : le diamètre fin bout, le diamètre large bout, le volume, la longueur, la courbure et la conicité.

1.1.2 Représentation des usines

Les simulateurs de débitage ont besoin de connaître la configuration d'une usine pour faire la simulation. Les simulateurs permettent de modéliser chacune des machines présentes dans l'usine. Les machines peuvent être configurées en fonction du niveau de précision désiré ou pour correspondre plus précisément à une usine réelle. Une fois que les machines sont modélisées individuellement, elles sont connectées entre elles de manière à reproduire les lignes de transformation présentes dans la scierie (Shannon, 1998). Le but de la modélisation est de reproduire aussi fidèlement que possible la transformation d'une pièce de bois pour une scierie donnée.

Afin de faciliter la visualisation des différentes lignes de transformation, les simulateurs de débitage peuvent avoir recours à la programmation visuelle (Green et Petre, 1996). C'est-à-dire qu'il est possible d'utiliser des éléments graphiques qui représentent chacune des machines et des flèches peuvent illustrer la connexion entre les différentes machines et dans quel ordre les produits sont transformés. Dans ce contexte, l'utilisation d'interface personne-machine permet à un spécialiste de visualiser les étapes de la transformation d'un billot et permet de comprendre les étapes de la transformation d'un billot.

1.1.3 Simulation

Les simulateurs sont les outils de choix pour prévoir le panier de produits associé à un billot. Ces outils ont pour but de reproduire le fonctionnement d'une vraie scierie. Cette approche modélise un comportement réel et permet de tester différents scénarios (Shannon, 1975). Il devient possible de tester plusieurs configurations d'une scierie pour valider des hypothèses sans avoir à réellement débiter des billots. Avec cette approche, il devient possible de faire des expérimentations pour comprendre des phénomènes observés ou pour évaluer des scénarios qui seraient possibles dans le futur (Law et Kelton, 2007).

Une fois la scierie modélisée, le simulateur détermine le panier de produits maximisant le rendement qui serait obtenu par la vraie scierie (Todoroki et al., 1990). Pour ce faire, le simulateur de débitage utilise un fichier qui décrit la liste de produits possibles qui contient aussi une valeur marchande ou une valeur qui permet de représenter l'importance d'un produit pour

un opérateur de scierie (aussi appelé valeur d'optimisation). Selon le contexte, le simulateur doit choisir entre la valeur marchande ou la valeur d'optimisation. Avec cette information, le simulateur effectue la simulation du débitage et ajuste les paramètres des différentes machines afin de produire le panier ayant la plus grande valeur.

Le principal désavantage de cette technique est la complexité de la tâche de modélisation. Ce processus demande la participation d'experts et peut prendre beaucoup de temps à réaliser. Un autre problème des simulateurs est qu'il est possible d'observer certaines perturbations lors du débitage en condition réelle que nous ne reproduisons pas avec le simulateur de débitage. Par exemple, un billot peut glisser pendant la rotation ou il peut se produire un imprévu avec la machinerie. Il devient donc important de pouvoir bien reproduire ces éléments stochastiques dans la simulation (Law et Kelton, 2007).

Finalement, un autre désavantage de la simulation est que selon la complexité de la configuration du simulateur et le scénario (ex., usine et nombre de billots), il est possible que la simulation demande beaucoup de ressources et de temps. S'il n'est pas possible de l'accélérer, il sera irréaliste de l'utiliser en pratique pour la prise de décision. La simulation peut prendre quelques minutes par billot. Si nous désirons simuler des milliers de billots, dans des dizaines d'usines pouvant avoir plusieurs configurations, il faudrait compter plusieurs heures ou jours pour effectuer toutes les simulations de débitage. Il serait possible de paralléliser les simulations, mais dans ce cas, il faut disposer des ressources nécessaires pour effectuer cette parallélisation. Malheureusement, il est fréquent de ne pas disposer du temps ou des ressources nécessaires pour pouvoir utiliser les simulateurs de débitage dans de tels contextes.

1.2 Approches pour prédire le panier de produits obtenus d'un billot

Il existe d'autres approches que les simulateurs de débitage pour obtenir le panier de produits associé à un billot. Il est possible d'utiliser des algorithmes d'apprentissage (discutés plus en détail à la section 1.3) qui trouvent une fonction permettant de faire une correspondance entre la représentation du billot et son panier de produits pour prédire le panier de produits pour un billot. Pour ce faire, les algorithmes se basent sur une banque d'exemples qui sont des paires (entrée, sortie). Les données obtenues pour faire cette correspondance peuvent être obtenues en utilisant des simulateurs de débitage, mais elles pourraient aussi être obtenues par la transformation de billot dans une scierie. Cependant, pour la seconde option, il est nécessaire d'avoir un traçage de bout en bout lors de la transformation du billot.

Une particularité de cette approche est que le résultat de la prédiction n'est pas assuré d'être un panier de produits valide pour billot comme c'est le cas avec les simulateurs de débitage. Cependant, Morin, Gaudreault et al. (2020) ont démontré que pour des tâches utilisant un

grand nombre de billots comme l'assignation de blocs de coupe, ces modèles d'apprentissage automatique peuvent fournir une amélioration importante, par rapport à d'autres modèles plus simples comme la prédiction moyenne, en demandant une fraction du temps nécessaire au simulateur de débitage pour faire les calculs.

Dans la littérature, il existe plusieurs recherches qui montrent qu'il est possible d'utiliser l'apprentissage automatique supervisé en remplacement des simulateurs de débitage. Dans un premier temps, il a été démontré qu'en se basant sur une représentation paramétrique des billots, il était possible de prédire avec un certain niveau de succès le panier de produits associé à ce billot (Morin, Paradis et al., 2015). D'autres approches ont été explorées dans le but de faire la correspondance entre un nuage de points et son panier de produits. Tout d'abord, il a été montré qu'il est possible d'utiliser l'algorithme *Iterative Closest Point* (ICP) pour prédire le panier de produits en retournant le panier de produits du nuage de points ressemblant le plus au nuage de points du billot évalué. Un point qui ressort de cette étude est que deux billots jugés similaires peuvent avoir des paniers de produits différents (Selma et al., 2018). Par la suite, il a été montré qu'il était possible d'améliorer cette prédiction en utilisant un premier filtrage qui élimine les billots trop différents et en comparant avec plusieurs billots (une approche similaire aux k voisins les plus proches) (Chabanet, Thomas et al., 2021 ; Chabanet, Chazelle et al., 2021 ; Chabanet, El-Haouzi et al., 2021). Cependant, cette approche n'offre pas des résultats supérieurs aux techniques traditionnelles d'apprentissage automatique basées sur les données paramétriques telles que les forêts d'arbres décisionnels.

1.3 Apprentissage automatique et réseaux de neurones

Traditionnellement lors de la résolution de problèmes, un expert crée et programme lui-même un algorithme destiné à résoudre un problème. Cependant, pour certains problèmes complexes, cette tâche peut être très difficile, voire impossible, à exécuter. Dans ce contexte, il est possible d'utiliser une catégorie d'algorithmes capables d'apprendre à faire cette tâche (Mitchell, 1997).

L'*apprentissage automatique* est une branche de l'informatique qui étudie les algorithmes capables de s'améliorer pour résoudre un problème. Une branche populaire de l'apprentissage automatique est l'apprentissage supervisé. Cette approche se base sur des données d'apprentissage, souvent extraites d'une application réelle, pour construire un modèle qui peut ensuite faire des prédictions pour des données tirées d'une même distribution (Hastie et al., 2009).

Pour l'*apprentissage supervisé*, les données d'entraînement sont composées d'une observation (typiquement, une instance du problème) et d'une étiquette (typiquement, la réponse réputée bonne). Lors de l'entraînement, on fournit les paires (observation, étiquette), aussi nommées paires (entrée, sortie), et l'algorithme génère une fonction qui fait un lien entre les observations et les étiquettes (Russell et Norvig, 2016). Il y a une grande variété d'algorithmes capables de faire cette relation.

Dans le contexte de l'apprentissage supervisé, il existe deux principaux types d'étiquettes de sortie. Dans un premier temps, on peut voir le problème comme étant un problème de classification. C'est-à-dire que nous faisons correspondre une observation à une étiquette représentant une catégorie, ex., une étiquette désignant un panier de produits précis. Dans ce cas, nous parlons d'utilisation de variables discrètes et finies. Une autre approche couramment utilisée est la régression où l'étiquette est continue. Dans le contexte de notre problème, pour cette approche, nous associons une observation à un vecteur de comptes servant à décrire l'observation : chaque indice représente un produit de sciage différent.

À la suite de l'entraînement, nous dirons qu'il y a surapprentissage si l'algorithme apprend trop précisément l'ensemble d'entraînement. Ce qui caractérise ce phénomène est que l'algorithme d'apprentissage permet d'obtenir de très bonnes performances sur l'ensemble d'entraînement alors qu'il n'est pas en mesure d'obtenir des performances similaires sur l'ensemble de tests. Un problème inverse est le sous-apprentissage (*underfitting* en anglais). Dans ce contexte, l'algorithme n'est pas en mesure d'apprendre le jeu de données sur l'ensemble d'entraînement. De manière générale, ce comportement indique que l'algorithme n'est pas assez performant pour bien faire l'apprentissage et il est probable que la performance de l'algorithme soit faible sur l'ensemble de tests (Koehrsen, 2018). Dans la pratique, il est important de trouver la configuration pour les algorithmes d'apprentissage automatique qui permettent d'éviter ces problèmes.

Dans ce qui suit, nous détaillons la classification et la régression pour le problème de prédiction du panier de produits. Premièrement, nous décrivons comment les paniers de produits, qui sont des vecteurs de comptes, peuvent être encodés sous forme d'étiquettes dans les cas de classification (section 1.3.1) et de régression (section 1.3.2). Par la suite, il y aura une présentation des *k plus proches voisins* (section 1.3.3), des *arbres de décision* (section 1.3.4) ainsi que des *forêts d'arbres décisionnels* (section 1.3.5). Finalement, nous explorerons une série d'algorithmes d'apprentissage automatique utilisant les réseaux de neurones (section 1.3.6).

1.3.1 Classification

Lorsque les algorithmes sont utilisés en classification, l'algorithme doit prédire une des étiquettes possibles pour ce problème. L'algorithme peut soit prédire directement la classe, soit produire une distribution de probabilités sur l'appartenance à chacune des classes. Dans ce cas, la classe ayant la plus haute probabilité est la classe choisie.

La classification est utilisée dans de nombreux domaines comme la reconnaissance d'objets dans une image. Dans ce contexte, ce qui est fourni à l'algorithme est une image et l'algorithme doit prédire ce que contient l'image qui lui a été présentée. Par exemple, dans le jeu de données MNIST, l'image est un chiffre manuscrit et l'étiquette correspond à un des 10 chiffres possibles (LeCun, 1998). Nous chercherons à reconnaître un panier de produits à partir de la représentation de la bille.

1.3.2 Régression

En régression, l'algorithme fournit une valeur numérique en sortie au lieu de prédire une étiquette. En régression, il est aussi possible de prédire plusieurs valeurs numériques en sortie.

Un exemple d'utilisation de régression est l'évaluation de la grandeur d'une personne en centimètres à partir de sa photo. Un exemple avec plusieurs valeurs numériques en sortie pourrait être de demander à l'algorithme de fournir la masse en plus de la grandeur de la personne. Dans notre contexte, le vecteur de comptes pour chacun des produits de sciage est représenté par la sortie d'une régression multi-sorties.

1.3.3 K-plus proches voisins

L'approche des k plus proches voisins, souvent reconnues comme étant k-NN dû à son nom anglais *k-nearest neighbors*, est une méthode d'apprentissage supervisé.

Pour effectuer ses prédictions, l'approche k-NN utilise une base de données composées de paires (observation, étiquette). Lors de la prédiction d'une nouvelle observation x, l'algorithme parcourt la base de données pour trouver les k échantillons ayant la distance la plus faible par rapport à x (Altman, 1992). Bien qu'il existe plusieurs approches pour calculer la distance entre les observations, la distance euclidienne est fréquemment utilisée.

Dans un contexte de régression, l'algorithme retourne la prédiction moyenne des k échantillons les plus proches en donnant la même importance à chacun des k échantillons. En classification, l'algorithme fait un vote de majorité sur la classe la plus populaire.

Les applications récentes des k-voisins les plus proches ont montré que cet algorithme fonctionne bien sur des données déséquilibrées. Ce qui est un atout pour un jeu de données comme le nôtre qui n'est pas garanti d'être balancé (Mani et Zhang, 2003).

1.3.4 Arbres de décision

Les arbres de décisions utilisent la correspondance entre les observations et les étiquettes pour générer une structure en arborescence qui permet de prédire l'étiquette. Dans cette arborescence, les nœuds sont représentés par un ensemble de décisions simples basées sur les fonctionnalités en entrée. Une prédiction est obtenue lorsqu'on atteint une feuille de l'arbre (Quinlan,

1986). Il est possible de paramétrer l'algorithme de manière à obtenir un arbre plus ou moins profond.

Le comportement de cet algorithme est un peu différent selon le cas où nous utilisons la classification ou la régression (Breiman et al., 1984). Dans le cas de la classification, la valeur de la classe présente à une feuille est la valeur de l'étiquette la plus fréquente qui atteint cette feuille. Dans le cas de la régression, l'arbre peut prédire une ou plusieurs valeurs numériques continues. Borchani et al. (2015) évalue les diverses approches pour l'utilisation d'arbres de décisions dans un contexte de régression multi-sorties.

Un avantage fort de cet algorithme est qu'il est relativement simple de comprendre pourquoi un modèle fait une prédiction puisqu'on peut analyser les conditions qui permettent de se rendre à une feuille.

1.3.5 Forêt d'arbres décisionnels

L'algorithme de forêt d'arbres décisionnels génère plusieurs arbres de décisions qui sont entraînés sur un sous-ensemble distinct de l'ensemble d'entraînement (Breiman, 2001). Par la suite, l'algorithme combinera la prédiction de l'ensemble d'arbres (la forêt) pour fournir la meilleure prédiction pour l'observation fournie. Cette approche peut fournir les résultats pour la classification et la régression (Schonlau et Zou, 2020). En classification, l'algorithme effectue un vote de majorité parmi les classes retournées par les arbres. Pour la régression, on fera une moyenne de la prédiction des arbres. Un des avantages de cet algorithme par rapport aux arbres de décisions est qu'il tend à diminuer le surapprentissage qui se produit lors de l'apprentissage.

1.3.6 Réseaux de neurones

Dans cette section nous détaillons l'utilisation des réseaux de neurones ainsi que leur fonctionnement. Par la suite, nous survolons une série d'architecture de réseaux de neurones. Soit les *perceptrons multicouches*, les réseaux à couches de convolutions, les réseaux de types *ResNet*, *VoxelNet* et *PointNet*.

Apprentissage par réseaux de neurones

Les réseaux de neurones sont une méthode d'apprentissage automatique qui s'inspire des recherches sur le fonctionnement du cerveau humain (Rosenblatt, 1961). Un réseau de neurones est composé de nœuds (aussi appelé neurone artificiel) qui peuvent être organisés suivant plusieurs architectures. Les différentes architectures que nous avons utilisées sont décrites dans les sous-sections suivantes.

Afin de faire une prédiction, les différents neurones reçoivent un signal qu'ils traitent en appliquant les poids contenus dans les neurones. Par la suite, le signal est transféré aux neurones

suivants dans le réseau de neurones. Typiquement, les réseaux de neurones sont organisés en couche et chaque couche contient un nombre variable de neurones en fonction du problème abordé. Entre les couches, les réseaux de neurones utilisent des fonctions d'activation qui sont des fonctions non linéaires qui permettent au réseau de neurones d'approximer des fonctions complexes.

La couche d'entrée du réseau est considérée comme étant externe au réseau et permet d'envoyer l'observation comme étant le signal pour les premiers neurones. La taille de l'entrée dépend essentiellement du problème et de la représentation choisis pour l'observation. Il existe des données tabulaires qui ne sont qu'une série de valeurs numériques, mais il est aussi possible d'utiliser des images et des nuages de points avec les réseaux de neurones. En revanche, la couche qui fournit les résultats est la couche de sortie et est responsable de fournir la prédiction correspondant à l'étiquette. Toutes les couches se situant entre la couche d'entrée et la couche de sortie sont des couches cachées du réseau.

Pendant l'entraînement, le réseau de neurones ajuste les différents poids présents dans ces neurones en minimisant l'erreur observée en sortie du réseau. L'erreur en sortie du réseau est calculée par une fonction de perte. Il existe plusieurs types de fonctions de perte qui sont utilisés en fonction du problème abordé. L'erreur quadratique moyenne (en anglais *Mean Squared Error* ou MSE) est une fonction populaire en régression alors que *cross entropy* (aussi connu sous le nom de *log loss*) est populaire pour les problèmes de classification.

L'erreur quadratique moyenne est définie comme suit :

$$\sum_{i=1}^{C} (\hat{y_i} - y_i)^2 \,. \tag{1.1}$$

Dans cette formule C représente le nombre de variables à prédire, y_i représente la valeur de l'étiquette à l'indice i et $\hat{y_i}$ est la valeur de la prédiction à l'indice i. L'erreur basée sur la *cross entropy* utilise une prédiction ($\hat{y}$) qui est représenté par un vecteur de N éléments ou chaque indice du vecteur représente une classe différente. La classe choisie pour la prédiction finale est la classe ayant la plus grande valeur (probabilité) dans le vecteur. La fonction de perte de la *cross entropy* est la suivante :

$$-\frac{1}{N} \sum_{i=1}^{N} (y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i})) \,. \tag{1.2}$$

Dans cette formule, N est le nombre de classes que l'algorithme tente de prédire, y_i (0 si ce n'est pas la classe à prédire, sinon 1) indique si la classe à l'indice i est la classe que nous désirons prédire et $\hat{y_i}$ représente la probabilité de la classe telle que prédite par l'algorithme.

Les réseaux de neurones sont entraînés pour un nombre fixe d'*epochs* ou jusqu'à ce que l'erreur observée en sortie ne diminue pas pour un certain nombre d'*epochs*. Une *epoch* est une itération où le réseau de neurones a appris sur toutes les données du jeu de données utilisé pour

l'entraînement. À la fin de l'entraînement, le but est d'avoir l'erreur observée la plus petite possible en sortie du réseau. Cependant, cette valeur atteint très rarement une valeur nulle. Si l'erreur est grande, cela veut dire que le réseau n'est probablement pas capable d'apprendre la fonction qui transforme l'entrée en sortie et pourrait ne pas bien généraliser.

Pour mettre à jour les poids dans le réseau de neurones, il est possible d'utiliser une technique qui s'appelle la *backpropagation*. Avec cette technique, l'erreur relative est remontée dans le réseau de neurones en se divisant sur chacune des connexions entre les neurones. L'idée derrière la *backpropagation* est de calculer le gradient de la fonction de perte. Par la suite, le gradient est remonté dans le réseau en appliquant le théorème de dérivation des fonctions composées pour chacune des couches présentes dans le réseau (Rumelhart et al., 1985). Ce qui implique que tous les éléments du réseau doivent être dérivables de bout en bout. Avec cette technique, il existe deux problèmes importants. Premièrement, il existe le problème de la disparition du gradient qui se produit lorsque l'erreur n'est pas en mesure de remonter dans tout le réseau parce qu'à certains points le gradient devient très faible. Le second problème est l'explosion du gradient qui se produit quand le gradient devient très grand dans le réseau. Ce problème a tendance à beaucoup modifier les poids dans le réseau, ce qui le rend instable lors de l'apprentissage (Pascanu et al., 2013).

La construction et l'apprentissage des réseaux de neurones sont affectés par un certain nombre d'*hyperparamètres*. Des exemples d'*hyperparamètres* sont le taux d'apprentissage (*learning rate*) et le nombre de neurones dans le réseau. Ces valeurs sont fixées avant l'entraînement du réseau, mais peuvent évoluer au cours de l'entraînement.

Perceptron multicouche

Un perceptron multicouche est généralement reconnu comme la version la plus simple d'un réseau de neurones. Dans ce contexte, le réseau de neurones utilise une série de couches pleinement connectées. Une couche pleinement connectée est une couche où chacun des neurones est connecté à tous les neurones de la couche suivante. Chacune des couches est séparée par une fonction d'activation non linéaire (LeCun et al., 1998).

Il n'y a pas d'obligations à utiliser une fonction d'activation particulière. Initialement, la fonction sigmoïde était populaire, mais dans les recherches récentes la fonction $ReLU$ a gagné en popularité puisqu'ils sont plus rapides à calculer et offrent des performances comparables à la fonction sigmoïde (Glorot et al., 2011).

Réseaux à convolutions

Les réseaux de neurones à convolutions sont une classe de réseaux de neurones couramment utilisés dans le traitement des images (Ciresan et al., 2011). Cette approche est inspirée par le fonctionnement du cortex visuel. Dans le cortex visuel, les neurones corticaux ne répondent

qu'aux signaux présents dans une certaine section du champ virtuel (Hubel et Wiesel, 1968). Pour reprendre cette idée, les couches de convolutions utilisent un ensemble de filtres qu'ils glissent sur les données fournies en entrée de la couche (LeCun et al., 1998).

L'idée de cette approche est de faire glisser les filtres sur l'entrée de chaque couche. On pourrait voir les filtres comme étant un curseur se déplaçant sur une fermeture éclair. Le curseur n'a accès qu'aux données étant couvertes par le filtre. Le but des filtres est donc de trouver des particularités locales qui vont fournir un signal fort pour le reste du réseau. Comme les mêmes filtres sont utilisés pour toutes les localités, nous pouvons dire que les poids sont partagés puisque les mêmes poids sont utilisés plusieurs fois dans une même couche. Ce partage de poids peut être vu comme une régularisation qui permet d'éviter le surapprentissage dans le réseau (LeCun et al., 2015).

Cependant, bien que cette forme de réseaux de neurones soit populaire dans le traitement des images, ils sont également efficaces dans d'autres contextes (Ji et al., 2012 ; Karpathy et al., 2014). C'est pour ces raisons que nous avons choisi de les utiliser dans un contexte en trois dimensions tel que présenté dans le chapitre 3.

Réseaux résiduels

Les réseaux de neurones résiduels sont une variation des réseaux de neurones qui s'inspirent de cellules pyramidales dans le cortex cérébral. Pour répliquer ce comportement, les réseaux utilisent des *skips connections* qui permettent de court-circuiter des parties du réseau de neurones. Cette approche est apparue dans le modèle *Resnet* qui est un réseau à convolution utilisé pour la classification et qui utilise les *skips connections* pour contrôler le signal dans le réseau (He et al., 2016). Cependant, l'idée des *skips connections* peut-être utilisée dans plusieurs formes de réseau de neurones (Srivastava et al., p. d.).

La motivation d'ajouter des sauts tente de régler deux problèmes dans les réseaux de neurones. Dans un premier temps, on diminue le problème de la disparition du gradient (*vanishing gradient*) puisqu'il est possible de contourner des parties du réseau qui pourraient réduire le gradient. La seconde idée est de régler le problème de saturation de la précision lorsque les réseaux sont trop profonds. Le problème de saturation se règle naturellement puisque le réseau de neurones possède maintenant un chemin pour faire circuler le signal de manière à éviter les couches du réseau qui causent le problème de saturation.

Il existe plusieurs variations de ces réseaux qui se sont montrés efficaces. On peut penser au réseau *HighwayNets* qui apprend le poids des sauts dans le réseau à l'aide d'une matrice de poids (Srivastava et al., p. d.). Les réseaux ayant plusieurs sauts partant d'une même couche s'appellent *DenseNets* et ont prouvé leur efficacité dans le traitement des images (Huang et al., 2017). Cependant, ces deux dernières approches n'ont pas été utilisées dans le contexte de ce travail.

VoxelNet

Il a été montré qu'il était possible d'utiliser des voxels pour reconnaître les objets dans une scène. *VoxelNet* est une des premières approches à obtenir du succès pour reconnaître et placer des objets dans un nuage de points (Zhou et Tuzel, 2018).

Pour cette approche, il faut diviser un volume dans une matrice en trois dimensions qui permet de décrire un volume en divisant l'espace avec une série de cellules de mêmes tailles. Chaque cellule dans la matrice est ce qui s'appelle un voxel et est composé d'une série de valeurs qui permet de décrire le contenu d'une cellule (Foley et al., 1990). Dans le cas des expérimentations initiales de *VoxelNet*, les recherches utilisent des scènes qui sont obtenues par des systèmes *LIDARs* qui sont en mesure de numériser une scène. Par la suite, il y a une étape de conversion qui convertit la scène numérisée (nuage de points) en voxels. Dans ce contexte, chaque voxel contient des valeurs numériques qui mentionnent si une classe d'objets est présente ou non. Si l'objet n'est pas présent, une valeur nulle est utilisée alors que 1 est utilisé si l'objet est présent.

L'architecture de *VoxelNet* utilise des couches appelées *Voxel Feature Layer*. Ces couches ont un fonctionnement qui pourrait être similaire aux convolutions à la différence que nous déplaçons un noyau dans un environnement en 3D au lieu de le déplacer dans un espace 2D.

Bien que ces approches aient un certain succès, les recherches récentes ont montré qu'il était possible d'obtenir de meilleurs résultats en utilisant d'autres architectures (Gujjar, 2018).

PointNet

PointNet est une architecture de réseau de neurones présentée par l'Université Standford en 2016 dans le but de reconnaître des objets ou de segmenter des scènes représentées par des nuages de points non reliés entre eux (Qi et al., 2017).

Des approches précédentes, telles que *VoxelNet*, convertissent ces nuages de points en voxels. Cependant, ne pas faire cette transformation vient avec un certain nombre d'avantages. Parmi les avantages, nous avons que travailler avec un nuage de points requiert moins de données, on évite la quantification en ne convertissant pas des valeurs continues vers un voxel qui doit posséder une résolution finie.

L'architecture de *PointNet* vient aussi avec d'autres avantages. La méthode proposée par les auteurs est invariante de l'ordre des points. C'est-à-dire que l'ordre dans lequel les points sont fournis au réseau n'a pas d'importance. Un autre avantage est que le réseau est invariant à la transformation. C'est-à-dire que le modèle ne tient pas compte de la translation ou de la rotation de l'objet numérisé. Dans notre contexte, cet avantage nous sauve le temps de préparation des données qui serait de trouver un alignement commun pour les billots et qui rend nos expériences consistantes.

1.4 Préparation des données pour les problèmes d'apprentissage automatique

Une des premières étapes de la résolution d'un problème utilisant l'apprentissage supervisé est la préparation des données. Ces étapes incluent le nettoyage de données, la génération des étiquettes, l'augmentation et l'exploitation des données.

Le nettoyage des données est une étape importante pour que l'apprentissage automatique soit efficace (Brownlee, 2020). Dans cette étape, on tente de retirer les données qui sont invalides. Bien qu'il serait souhaitable que les données soient toujours valides, il est possible que les données dans le jeu de données ne soient pas valides. Des exemples de nettoyages de données pourraient être de retirer les doublons et retirer les données incomplètes. Une étape cruciale lors du nettoyage de données est de s'assurer que les données ne « coulent » pas de l'ensemble de tests vers l'ensemble d'entraînement. Un tel comportement aurait pour effet de donner l'impression que l'apprentissage est plus performant qu'il ne l'est vraiment.

L'augmentation est une technique utilisée pour augmenter la quantité de données disponibles pour l'apprentissage. Pour ce faire, nous modifions légèrement les données déjà présentes dans le but de créer de nouvelles données qui ont la même étiquette (Wang, Perez et al., 2017). Dans le cas de la classification d'images, il est possible de modifier les couleurs, d'appliquer une translation ou d'appliquer une rotation. Cette technique a prouvé être efficace pour les réseaux de neurones utilisant des images.

Dans le cas de la classification, un débalancement des classes peut nuire à l'apprentissage (Abd Elrahman et Abraham, 2013). Une technique qui a prouvé être efficace est le suréchantillonnage (Barandela et al., 2003). Cette technique pige au hasard des exemples des classes minoritaires et augmente leur nombre d'occurrences dans le jeu de donnée. Un risque de cette technique est de trop augmenter une classe minoritaire ayant une observation similaire de telle sorte qu'elle ait une distribution similaire à une classe plus populaire et que l'algorithme ne soit plus en mesure de faire la distinction entre les deux classes.

Il existe d'autres approches pour générer des données synthétiques. Une approche qui peut être utilisée serait de générer complètement de nouveaux exemples. Typiquement, on sait que cette approche fonctionne bien avec les réseaux antagonistes génératifs (Bowles et al., 2018). Dans cette approche, l'idée est d'avoir deux réseaux. Un premier qui permet de dire si les données présentées constituent un exemple valide. On appelle ce réseau le *discriminateur*. Le second est utilisé pour générer des données qui pourront être utilisées par le modèle. Ce modèle s'appelle le *générateur*. Le but est d'entraîner les deux réseaux de manière à ce que le générateur essaie de déjouer le discriminateur et que le discriminateur devienne le plus performant possible à détecter les faux exemples. De cette manière, on peut arriver à des exemples convaincants de nouvelles données (Makhzani et al., 2015).

Il y a d'autres méthodes permettant de générer de nouveaux exemples en se basant sur les connaissances d'experts dans le domaine. Avec ces données, il est possible de créer de nouveaux exemples qui sont valides et qui aideront l'apprentissage (Barse et al., 2003).

Chapitre 2

Quality of sawmilling output predictions according to the size of the lot – The size matters !

2.1 Résumé

Lors de l'évaluation de modèles d'apprentissage automatique supervisé, on considère généralement le rendement de prédiction moyen obtenu sur les tests individuels comme mesure de choix. Toutefois, lorsque le modèle est destiné à prédire quels produits du bois seront obtenus lors du sciage de certains billots, c'est généralement la performance pour un lot complet qui importe. Dans cet article, nous montrons l'impact de cette nuance sur le plan de l'évaluation du modèle. En fait, la qualité d'une prédiction (globale) s'améliore considérablement lorsque l'on augmente la taille des lots, ce qui offre un solide soutien à l'utilisation de ces modèles en pratique.

2.2 Abstract

When comparing supervised learning models, one generally considers the average prediction performance obtained over individual test samples. However, when using machine learning to predict which lumber products will be obtained when sawing logs, it is usually the performance over the entire lot that matters. In this paper, we show the impact of this by evaluating a model performance for various batch sizes. The quality of a (global) prediction improves tremendously when batch size increases, which offers a strong support for the use of such models in practice.

2.3 Introduction

For many reasons, the forest product industry needs forecasting what its inventory would allow producing. There exist needs for performance forecasting, configuration decisions and wood allocation between several plants. It has been shown that accurately forecasting sawmills output can improve profitability due to better decision-making (Morin, Gaudreault, et al., 2020). Tools for such evaluations include sawmill simulators, such as the Optitek (FPInnovations, 2014). These tools are designed to simulate the transformation process of a log at a given plant (Lin et al., 2010). However, simulation is computationally expensive with large volume of woods, for companies owning multiple plants or when testing multiple sawmill configurations. As a result, supervised learning has been recently proposed to improve or replace slow simulators by a fast relation from the space of the logs characteristics to the space of the sawmill outputs, i.e. lumbers (Morin, Paradis, et al., 2015).

Morin, Gaudreault, et al. (2020) showed that, although supervised learning models produced for that particular problem are only approximate, they remain more than useful for decision-making, e.g. to make decision regarding wood allocation between sawmills. In this paper, we explain why seemingly low performance on individual logs does not hinder a models' usefulness for decision-making. As demonstrated by our experiments, the errors in the prediction for some given logs are partially compensated by the errors on other logs as the batch size increases. In addition, sophisticated models, i.e. models trained using a machine learning procedure, proved useful and more accurate than a simple average for all batch sizes and especially for smaller batches. This explains the good performance of such models for decision-making, especially on the wood allocation problem (Morin, Gaudreault, et al., 2020). It makes these models especially promising in the specific use cases where a simulator tends to be too slow, i.e. on large amount of data.

The paper is organized as follows. We first review the related literature and concepts in Section 2.4. In Section 2.5, we present the experimental process and data supporting our contributions, interpret the precision, the recall and the F1-score performance measure in the context of our machine learning problem, and present the results. We conclude in Section 2.6.

2.4 Related Concepts and Literature

In the following subsections, we introduce general notions regarding sawing simulation as well as supervised learning for sawing simulation.

2.4.1 Sawing Simulators

Sawing simulators make it possible to anticipate which products a log will generate when processed at a given sawmill. These tools generally take three inputs, namely a sawmill model

(described in a suitable formal language), a feasible product list, and a virtual description of the log.

The sawmill model details the equipment, the links between the machines as well as their capacities. The feasible product list describes the lumbers that can be produced by the factory along with their value. Log sawing by-products, such as sawdust and shavings, are also associated with values. A product value can either be based on its true market value, or it can reflect its importance w.r.t. the current needs of the company. Finally, a virtual log is a representation of a log that needs to be cut, often a three-dimensional scan. It is generally modeled as a point cloud approximating the surface of the log (Thomas, 2013).

Simpler representations, such as parametric descriptions, might be used in some studies. Parametric descriptions are vectors of characteristics such as the length of the log, its strong diameter, its weak diameter, its curvature, its tapering, and its volume. Other characteristics, such as the log species, could be introduced in the parametric description.

Given a sawmill model, a feasible product list and a virtual log, the sawing simulator performs a simulation of all the feasible cuts to choose the one leading to an optimal yield for this log, i.e. the simulator finds a set of lumbers maximizing the total value of the transformed log (Todoroki et al., 1990). An example of a sawing simulation result can be seen in Figure 2.1.

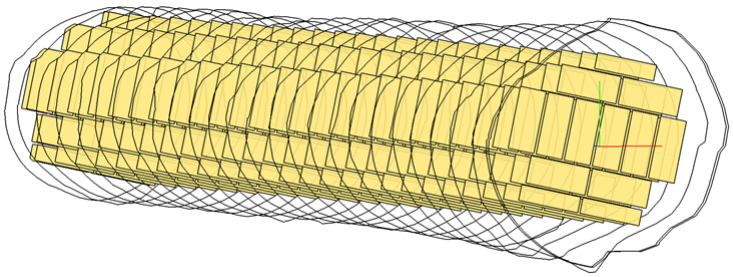

FIGURE 2.1 – Representation of a log and of the lumber products resulting from its break down.

2.4.2 Supervised Learning for Sawing Simulation

Although there exist some machine learning model using the three-dimensional log scans as input (e.g. (Selma et al., 2018)), we focus in this paper on models using the parametric description of a log. Both types of models provide as output a vector of counts for each feasible lumber products. This vector of counts describes the basket of products one should obtain if processing the log at the modeled sawmill.

Supervised learning algorithms aims to build a mapping from a feature vector space to a label space using a training set of examples consisting of pairs of inputs and outputs. In our context, the features are the logs parametric descriptions, and the labels are the vectors of counts representing the baskets of products.

In supervised learning, there are two main types of output labels, namely discrete (and finite), and continuous. The former leads to what we call a classification problem whereas the latter leads to a regression problem. In what follows, we describe how basket of products, which are vectors of counts, can be encoded as labels in both the classification and the regression cases.

Classification for Sawing Simulation

A class is a discrete label. For classification purposes in the sawing simulation context, it is possible to consider each individual basket of products encountered in the training set, i.e. combination of lumber products, as a specific class.

The drawback of this approach is that each specific combination needs to be encountered in the data. Therefore, it is possible that some feasible combinations of products, i.e. some feasible baskets, cannot be predicted by the model. One example would be a class that is present in the test dataset and not in the training dataset. Conceptually, one of the main benefits of the approach is that all the predictions are guaranteed to be feasible for at least one existing log (as they have been seen at least once in the data).

Regression for Sawing Simulation

When performing a regression in our context, a regression model individually predicts the number of products of each feasible lumber from the feasible product list (i.e. each position in the vector count).

Contrary to classification it is now possible to predict any basket of products even if they were not part of the train set. For each of the feasible product, the model provides a real number. The drawback of the approach is that there are no such things as decimal lumber counts in practice. This might be a problem when using the model for applications that need predictions for individual logs. However, in many applications, there is few need to obtain precise counts and we aim at evaluating a global prediction, i.e. the prediction on a batch of logs. As a result, we are using the real values provided by the model directly to compute the performance on a batch of logs.

This approach comes with some challenges. One way to address the problem would be to create one model per product output. The problem with this approach is that it assumes independent outputs. In the case of our application, individual entries in the count vector are related to one another. That is, each predicted product might take the space needed for another. For that reason, we used a single model trained to predict the entire vector of counts.

The specific procedure depends on the chosen learning algorithm.

Learning Algorithms for Sawing Simulation

In the literature, multiple learning algorithms have been evaluated in the context of sawing simulation. In this paper, we focus on three machine learning algorithms that were proved efficient for that problem by Morin et al. (2015), namely k-nearest-neighbors (KNN) (Fix, 1985), decision tree (Breiman et al., 1984), and random forest (Breiman, 2001). We experiment with both the regression and the classification version of these algorithms.

k-Nearest-Neighbors The KNN algorithm is based on the idea that similar logs should have similar product baskets. During training, the algorithm will use a data structure that makes it easy to compare the distance between two logs. In this study, we use the Euclidian distance between the parametric description of two logs. The algorithm chose the k nearest logs.

In classification mode the algorithm performs a majority vote to determine the class whereas in regression mode it averages the basket of the nearest k logs. Although we tested multiple values of k, a value of one allowed us to achieve good results in training while maintaining a good generalization for examples that were not seen in the training phase. Recent applications of KNN has shown that it performs well on unbalanced data, e.g. (Cai et al., 2020). Since our dataset is highly unbalanced, it made the algorithm a promising choice in the context of our application.

Decision Tree The decision tree learning algorithm uses the training examples to generate a tree-based model where the nodes are simple decision rules based on the input features. For prediction, an unseen example is passed through the tree and the model output is determined when it reaches a leaf.

In classification mode, each leaf represents the class to be predicted which is determined by the examples the training set that reached that leaf during training (by a majority vote). In the case of regression, we use a variation of the algorithm to predict multiple output values. Although, it would be possible to create a set of independent trees, product counts in the basket are dependent. We therefore used a single tree that can predict the entire basket at once such as described in (Borchani et al., 2015). Recent applications of the decision tree algorithm assessed it as a simple, efficient, and easy to interpret model (Lan et al., 2020).

Random Forest The random forest algorithm builds multiple decision trees. It does so by sampling the training set N times. Each sample is used to create a single tree (Schonlau and Zou, 2020). For prediction, the data are passed through each tree and the predictions are combined. In the case of classification, a majority vote is performed whereas in regression the trees outputs are averaged.

2.5 Experiments

Our experiments are intended to show the impact of the size of the lot on the quality of the prediction for the prediction models using the KNN, the decision tree and the random forest learning algorithms. In this section, we present the industrial data used for this study 2.5.1, the details of the model-building phase 2.5.2, and the evaluation metrics we use to compare the quality of the models 2.5.3. Finally, we analyze and compare the performance of our models on three aspects. First, by exploring the average prediction quality on individual logs 2.5.4. Second, by comparing the performance on batches of increasing size 2.5.5. Third, by highlighting the most common source of error in predictions 2.5.6.

2.5.1 Data

Data was provided by FPInnovations. Our dataset contains a total of 2,235 logs. All the logs from this dataset are known to produce a non-empty basket of products. To train the supervised learning models, we used the following log characteristics: length, strong diameter, weak diameter, curvature, taper, and volume.

We used the Optitek sawing simulator FPInnovations (2014) to virtually transform each of these 2,235 logs and obtain the actual basket of products. A total of 85 different sawing products are present in those vectors leading with 85 entries.

We obtained 1,188 different baskets of products which leads to 1,188 classes when considering the classification version of the problem. Considering the high number of classes relative to the number of examples in our dataset, it is expected that several classes will have a low representation. For example, there are roughly 30% of classes that are only present once in the dataset while one of the classes is present 58 times. This particularity of the dataset implies that certain classes in the training set will not be present in the test set (and vice-versa).

This class distribution is particularly complex for classification models because it is expected that these models do not try predicting classes that have not been seen in training. However, models should predict similar classes that are present at training. In this situation even if the classes are different, it is possible that these two classes have several common elements.

It is important to mention that the logs were not chosen for the purpose of having a balanced input or output. This means that some lengths or diameters are more common than others

and some only appear once. The same thing is true for the basket of products, hence the classes.

Figure 2.2 shows a histogram of the length of the logs in the complete dataset. The figure allows us to see that some lengths are much more frequent and that some lengths are very rare and have only 1 example (5.43 meters). Furthermore, we see that around 800 logs have a length of 3.76 meters. This distribution is expected to complicate the task of fitting a model on the data.

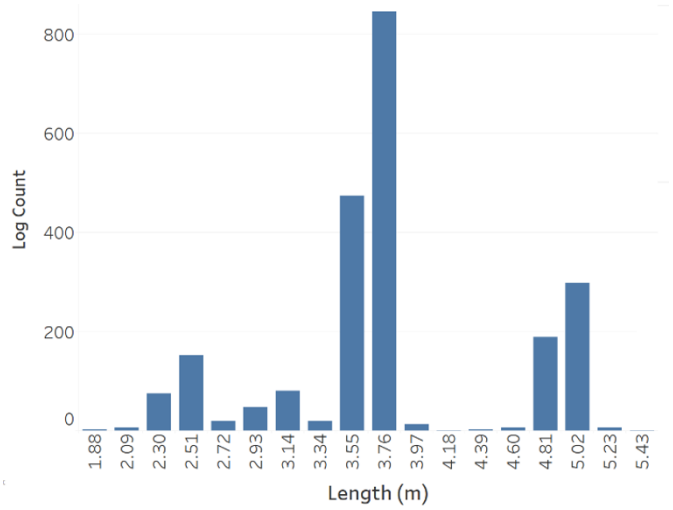

FIGURE 2.2 – Distribution of logs by length (m) in the dataset.

2.5.2 Model Training

To carry out the experiments, the logs in the dataset is randomly separated so that 85% of the logs are used for training, i.e. to fit a model, and 15% are used to form a test dataset to evaluate the performance of the model on unseen data. We created 10 such partitions which will allow the experiments to be repeated 10 times for each of the algorithms.

For all splits, we first trained models using the KNN, the decision tree, and the random forest algorithms on the training set. Furthermore, to establish a base case reference, we also consider a dummy model which systematically predicts the average basket of products as seen in the train set. It should be noted that the dummy model prediction procedure is akin to simple forecasting heuristics such as using the historical average of the production.

2.5.3 Performance Measure

To evaluate the performance of the learned models, we need to assess how the basket of products they predict (vector y) compares to the actual basket of products (vector b) generated by the simulator. Vectors y and b may either correspond to the quantities obtain from the sawing of a given log or of a whole batch of logs.

In what follows, we use the well-known precision, recall and F1-score metrics to evaluate the models [(Rijsbergen, 1979), (Powers, 2020)]. We recall that precision and recall are calculated using the number of true positives (TP), the number of false positives (FP), and the number of true negatives (TN) (Rijsbergen, 1979). In our context, we define TP, FP, and TN as follows:

— TP represents the number of lumbers that are present in both vectors.

— FP is the number of lumbers predicted but not actually produced.

— FN is the number of lumbers that were manufactured but not predicted.

Given a predicted basket y and an actual basket b, the precision is defined as

$$precision(y, b) = \frac{TP(y, b)}{TP(y, b) + FP(y, b)}\,, \tag{2.1}$$

and the recall is defined as

$$recall(y, b) = \frac{TP(y, b)}{TP(y, b) + FN(y, b)}\,. \tag{2.2}$$

Finally, the F1 score is computed as the harmonic mean between precision and recall which leads to the following equation:

$$F1(y, b) = \frac{2TP(y, b)}{2TP(y, b) + FP(y, b) + FN(y, b)}\,. \tag{2.3}$$

It should be noted that by evaluating a set of models built using a learning algorithm and comparing their average results to the average results of models built using another learning algorithms we are able to determine which algorithm generates the best models either for individual logs or for a batch of logs.

2.5.4 Performance on Individual Logs

This section is used to demonstrate the quality of our prediction for individual logs. In the case of individual log predictions, we present the performance metrics for individual predictions averaged on a batch of unseen logs and multiple replications where a replication consists of the following three steps (as described in Section 2.5.2). First, we partition the dataset into a training set and into a test set. Second, we use each learning algorithm to build a model based on the data of the training set. Third we evaluate each of the generated models on the test set by using it to make a prediction of the basket of each individual log it contains. The quality of the prediction for individual logs in terms of a specific metric, e.g. the F1-Score, is then averaged.

Figure 2.3 reports the average F1-Score for the different models we are using in this research. We are comparing the decision tree, KNN and random forest models to the dummy predictor.

FIGURE 2.3 – Average F1-score for individual predictions on the test set (95% confidence intervals; 10 replications; individual replication scores are shown in blue).

All models perform much better than the dummy prediction. This result can be explained by the large variety of feasible basket of products in our dataset and their uneven distribution. This characteristic of the dataset proves to be problematic for the dummy model.

Among all the tested learning algorithms, KNN shows the worst performance. As it was shown by Selma et al. (2018), similar logs could have very different basket of products which hinders nearest neighbors approaches efficiency. Although, the difference between KNN and the dummy predictor is 28%.

Tree-based machine learning algorithms (decision tree and random forest) performed best. This is in line with the result of Morin, Paradis, et al. (2015). What is interesting is that the regression and classification models have similar performance. This result highlights the possibility to achieve good prediction even if the training dataset does not contain at least one example of each feasible basket of products.

Using regression also provides an advantage over classification. For our experiments, the number of classes is defined by the number of different basket of products present in our dataset. One problem with classification is that small training datasets are very likely not to contain all feasible classes. As a result, a classification model output is likely to be wrong on at least some of the products in the baskets. In regression, this is less of an issue since the model can output an unseen basket.

Finally, the model with the best performance is the random forest in classification. In this case the algorithm is performing 39% better than the dummy predictor.

2.5.5 Performance for Batches

In this section, we explore the effect of the size of the batches on the quality of the forecasts. We also validate our initial hypothesis that the errors made by a model when predicting the

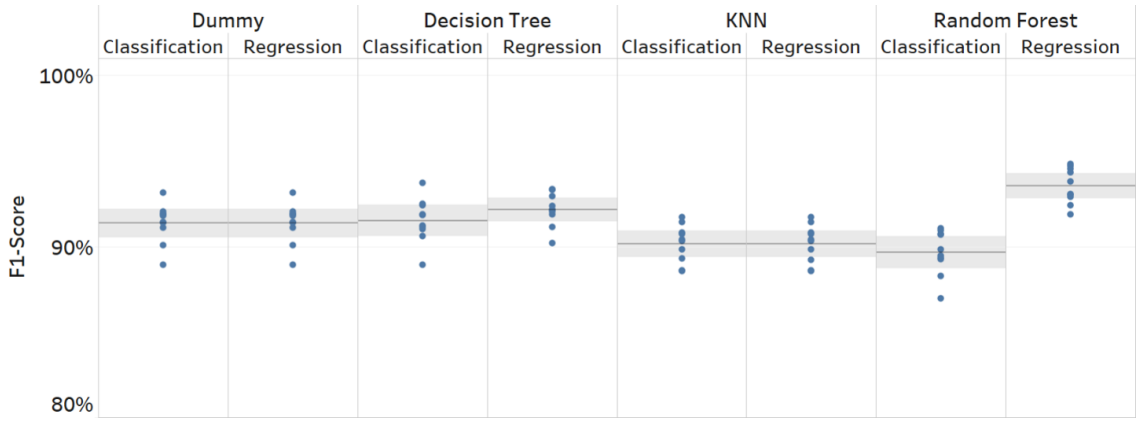

FIGURE 2.4 – Average F1-score for global predictions on the test set along (95% confidence intervals; 10 replications; individual replication scores are shown in blue).

output of a sawmill for an unseen log is compensated for by errors made on other logs. Finally, we confirm that sophisticated models are useful for all tested batch sizes and especially on smaller batches where a simple average does not achieve a sufficient performance.

Figure 2.4 reports the F1-score for the prediction of the global basket of products associated with the whole test dataset. We can see a significative improvement for all models. All models improve by at least 40% in terms of F1-score. The dummy predictor has the largest improvement (approx. 76%). This result is not surprising because with a large enough sample the average should improve to be closer to a perfect prediction. This establishes a base case for the other algorithms.

It should be noted that the random forest algorithm in classification is no longer the best performer and the accuracy of its prediction is only 89%. Not only is this model no longer the best performer, but it is now our worst performer. KNN has a F1-score of about 90%, which is lower than that of the dummy predictor. The best performer is random forest in regression with a F1-score of 93%.

Figure 2.5 illustrates more precisely how the performance is affected by the size of the batch. We kept the same test dataset but report global results for randomly chosen subsets of different sizes.

We notice that the performance curve is logarithmic. The batch size does not need to be huge to see dramatic performance improvement. This renders the models useful for a wide range of applications involving small batches of logs such as log piles, e.g. for wood allocation purposes. Companies can therefore achieve interesting results in decision-making on such problems using such models although they appeared to be only approximate for individual predictions.

In classification all models perform better than dummy for smaller batch size; the smaller the batch is, the greater the difference. For example, decision tree will perform 42% (in terms of

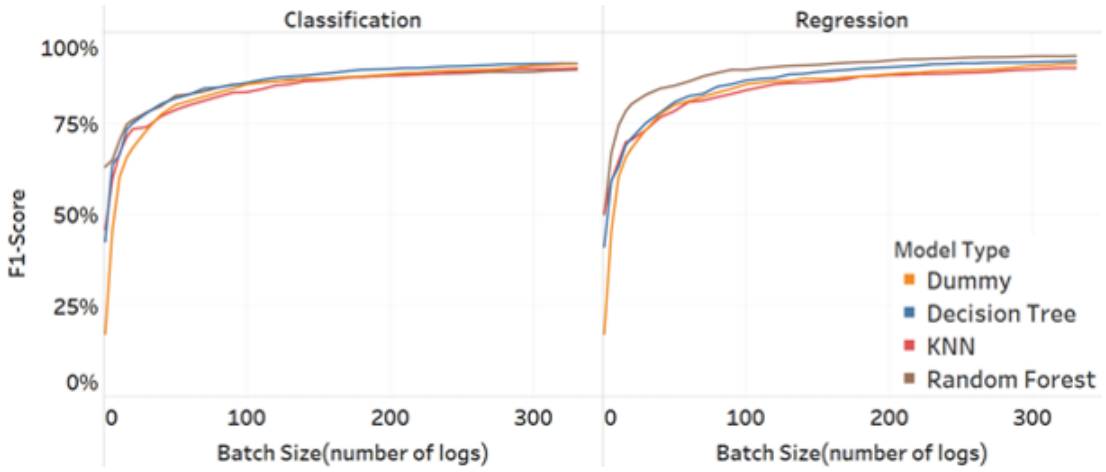

FIGURE 2.5 – F1-score (in %) for KNN, Decision Tree, Random Forest, and Dummy as a function of the size of the batch (number of logs).

F1-score) better than dummy for batch size of 1. At batch size of 50, we can see that dummy is beating the KNN algorithm and will do better than random forest at batch size of 110. With our experiment dummy never reach the level of prediction of the decision tree, but with a larger test dataset it is possible that dummy would yield better result than the decision tree classifier.

For regression, the picture is different. Again, for small batches all our algorithms perform better than the dummy predictor and again the dummy predictor performs better than KNN (starting at a batch size of 50). However, this time decision tree and random forest are doing better than dummy for batch size of 330. Not only do these models perform better, but random forest has a prediction that is 3% over the dummy in terms of F1-score.

This clearly supports the fact that the regression version of random forest dominates the other approaches and would be a safe choice whatever is the size of the lot.

Figure 2.6 details the F1-score, precision, and recall metrics according to the size of the batch for the best classification model (decision tree) and the best regression model (random forest). Recall and precision are relatively close to each other regardless of the size of the lot. For both models, recall and precision tends to both increase with batch size. However, we can point that the difference is much smaller for decision tree in classification mode. The distance between precision or recall and the F1 score is of about 0.2%. For random forest in regression mode, the spread is larger, and the difference is slightly over 1%. We also notice that the difference tends to be larger for smaller batch sizes than for larger batch sizes.

Figure 2.5 shows the 95% confidence intervals around the F1-score average for the various batch size. The spread of the 95% confidence interval diminishes significantly as batch size increases. This is especially true for small batches, e.g. when increasing the size of the batch

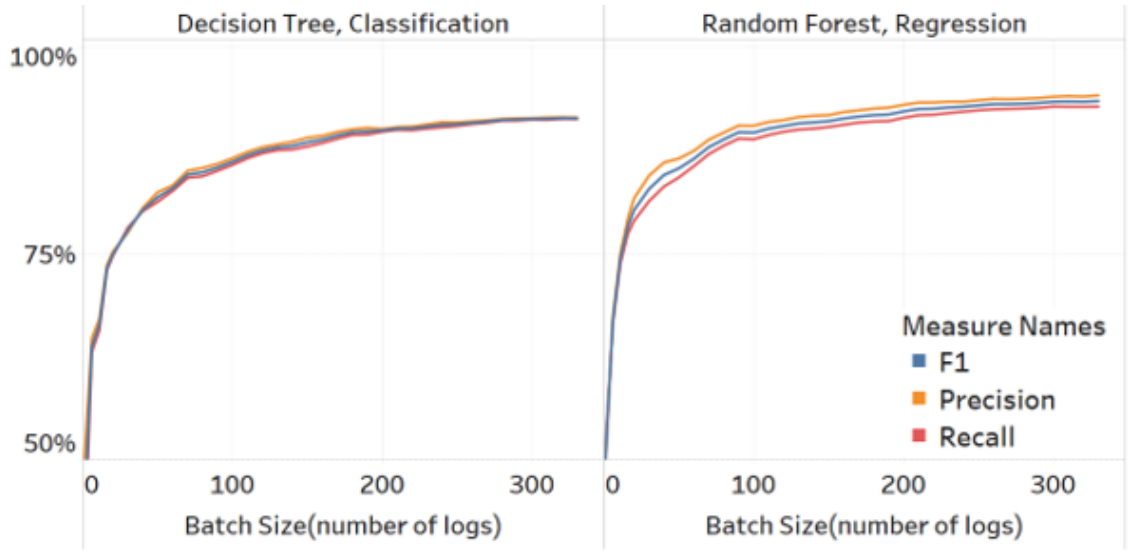

FIGURE 2.6 – F1-score, precision, and recall (in %) as a function of the size of the batch (number of logs) for decision tree in classification and random forest in regression

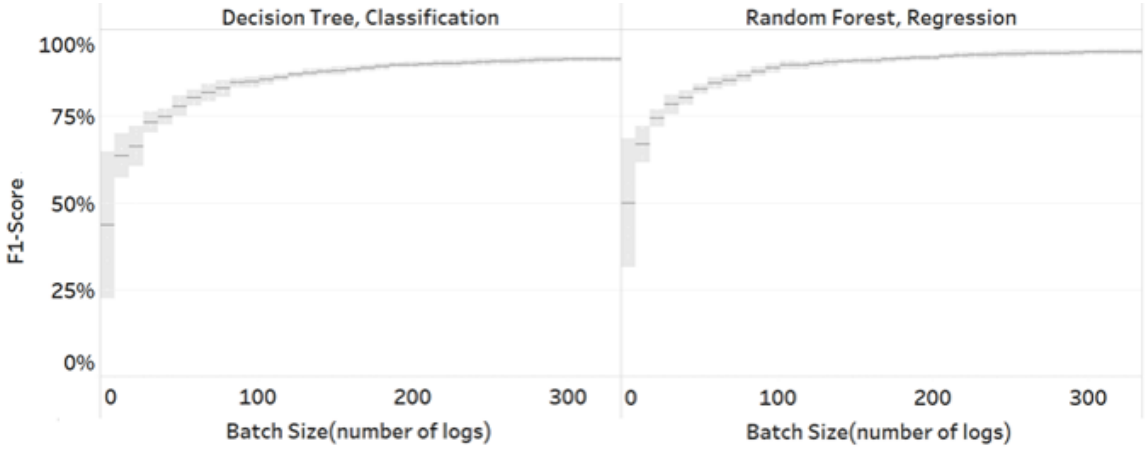

FIGURE 2.7 – 95% confidence intervals for F1-score as a function of the batch size (number of logs)

from 1 to 10 logs. Of course, for small batches, it is possible to make a prediction that is wrong that has a large impact on the overall prediction for the batch. The fact that the interval is narrowing as the size of the batch increases confirms our hypothesis that errors tend to be canceled.

2.5.6 Sources of error

Each given type of lumber product has a length, a width, a thickness, and a grade which, taken together, fully define the product. We call these characteristics the lumber attributes. By artificially ignoring errors on a specific lumber attribute, we artificially increase the score of

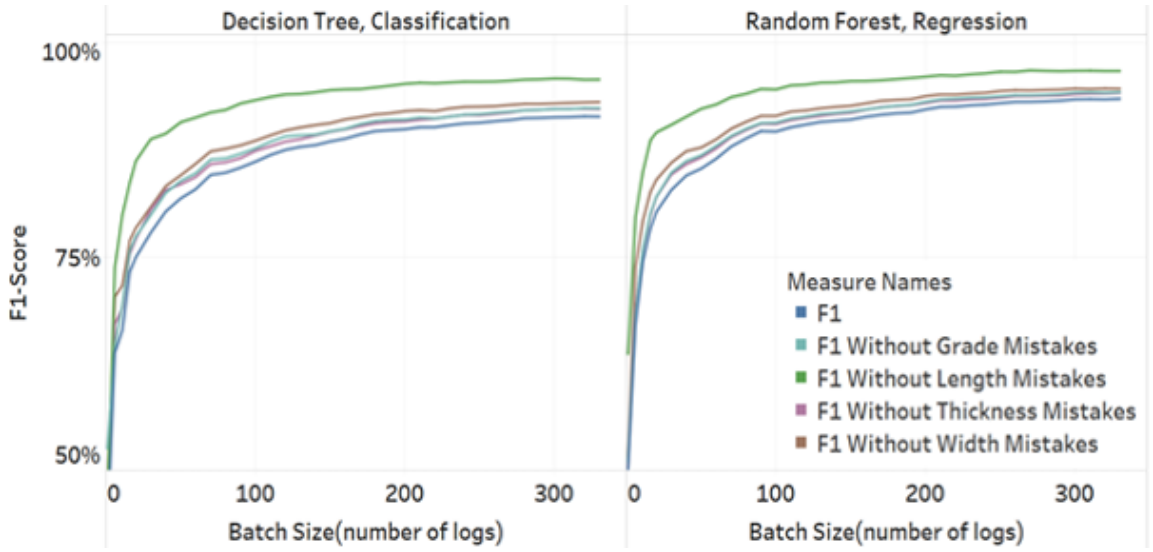

FIGURE 2.8 – F1-score with and without single attribute mistakes for all types of error as a function of the size of the batch (number of logs).

the original model. Let us suppose we choose ignoring the length errors. Then, the magnitude of the score increment reflects the number of times the original model predicted the wrong length for a lumber without making a mistake on the other attributes.

The idea behind this analysis is to visualize "where" our models produce errors, but also to understand how to improve them in the long run. If, for instance, a model that predicts products that have the right width, length, and thickness is frequently misled on the grade, then it may be possible that the parametric data does not provide enough information to properly predict the grade. In this case it may be necessary to add information in the parametric description of the logs to improve the model.

Figure 2.8 reports the F1-score when we do not consider a specific type of mistakes. It shows that the length is the main source of error in our predictions. This is especially true in the case we are using a decision tree classifier as the curve without length mistakes is clearly above the others.

The curve representing the errors along the length is the curve that detaches most from the other curves. This suggests a higher error rate on length compared to other lumber attributes. Of course, other attributes also appear as sources of errors when inspecting the other curves, but the magnitude of the gain when ignoring them is less.

2.6 Conclusion

We showed how, in the context of supervised learning for sawing simulation approximation, the quality of a model can not only be quantified by its performance on individual logs, but

also by its performance on batches. These results confirm that prediction errors on one log can be compensated for by errors on other logs, but also shows why more sophisticated models are preferable than a simpler model based on the historical average, especially for small batches.

Furthermore, we presented metrics that can be easily interpreted in an industrial context in the forest-product industry. Since those metrics make it easier to understand what is happening with the models in terms of over and under prediction of lumber counts, it becomes possible to explore the sources of errors in the models and interpret them in the context of the application. Those analyses pave the way to further model improvements.

2.7 Acknowledgement

The authors would like to thank the FORAC Research Consortium and its partners. Our gratitude goes as well to the Natural Sciences and Engineering Research Council of Canada (NSERC) who provided funding for this research.

References

Borchani, H., G. Varando, C. Bielza, and P. Larranaga (2015). « A survey on multi-output regression ». In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.5, pp. 216–233.

Breiman, L. (2001). « Random forests ». In: *Machine learning* 45.1, pp. 5–32.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and regression trees*. CRC Press.

Cai, L., Y. Yu, S. Zhang, Y. Song, Z. Xiong, and T. Zhou (2020). « A Sample-Rebalanced Outlier-Rejected k-Nearest Neighbor Regression Model for Short-Term Traffic Flow Forecasting ». In: *IEEE access* 8, pp. 22686–22696.

Fix, E. (1985). *Discriminatory analysis: nonparametric discrimination, consistency properties*. Vol. 1. USAF school of Aviation Medicine.

FPInnovations (2014). *Optitek 10. User's Manual*.

Lan, T., H. Hu, C. Jiang, G. Yang, and Z. Zhao (2020). « A comparative study of decision tree, random forest, and convolutional neural network for spread-F identification ». In: *Advances in Space Research* 65.8, pp. 2052–2061.

Lin, W., J. Wang, and R. E. Thomas (2010). « A three-dimensional optimal sawing system for small sawmills in central Appalachia ». In: *17th Central Hardwood Forest Conference*. Vol. 78, p. 67.

Morin, M., J. Gaudreault, E. Brotherton, F. Paradis, A. Rolland, J. Wery, and F. Laviolette (2020). « Machine learning-based models of sawmills for better wood allocation planning ». In: *International Journal of Production Economics* 222, p. 107508.

Morin, M., F. Paradis, A. Rolland, J. Wery, F. Laviolette, and F. Laviolette (2015). « Machine learning-based metamodels for sawing simulation ». In: *2015 Winter Simulation Conference (WSC)*. IEEE, pp. 2160–2171.

Powers, D. M. (2020). « Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation ». In: *arXiv preprint arXiv:2010.16061*.

Rijsbergen, C. van (1979). *Information Retrieval, 2nd edButterworths*.

Schonlau, M. and R. Y. Zou (2020). « The random forest algorithm for statistical learning ». In: *The Stata Journal* 20.1, pp. 3–29.

Selma, C., H. B. El Haouzi, P. Thomas, J. Gaudreault, and M. Morin (2018). « An iterative closest point method for measuring the level of similarity of 3D log scans in wood industry ». In: *Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, pp. 433–444.

Thomas, R. E. (2013). « RAYSAW: A log sawing simulator for 3D laser-scanned hardwood logs ». In: *Proceedings, 18th Central Hardwood Forest Conference*. Vol. 117, pp. 325–334.

Todoroki, C. et al. (1990). « AUTOSAW system for sawing simulation ». In: *New Zealand Journal of Forestry Science* 20.3, pp. 332–348.

Chapitre 3

Prédiction des produits de sciage par l'utilisation des réseaux de neurones

Ce chapitre présente les résultats de différentes approches utilisant les réseaux de neurones pour prédire le panier de produits associé à un billot. Les réseaux de neurones ont montré être efficaces sur une grande variété de problèmes (He et al., 2016 ; Qi et al., 2017). Dans ce contexte, nous décrivons diverses représentations du billot dans le but de trouver une représentation permettant un meilleur apprentissage des réseaux de neurones (section 3.1). En effet, jusqu'à maintenant, la majorité des recherches sur la prévision de la sortie d'une usine pour la transformation d'un billot utilisent les données paramétriques associées au billot et non les numérisations de billots ou un encodage de ces numérisations. Par la suite, nous expliquons la procédure d'entraînement utilisée lors de l'apprentissage des réseaux de neurones (section 3.2). Finalement, nous présentons les résultats pour les prédictions individuelles (un billot à la fois) en utilisant les réseaux de neurones (section 3.3).

3.1 Représentations alternatives

Nous décrivons différentes représentations de la couche d'entrée, à savoir les nuages de points bruts (section 3.1.1) et les nuages de points normalisés (section 3.1.2), les projections 2D de la surface billots (section 3.1.3) et les caractéristiques extraites du savoir-faire de l'industrie des produits forestiers que nous appelons données paramétriques (section 3.1.4).

3.1.1 Nuages de points bruts

Les nuages de points bruts sont obtenus de différentes manières. Par exemple, les scieries ont généralement des numériseurs au début de leur ligne qui numérisent les billots entrant de manière à maximiser le rendement de la transformation en produits de sciage. Certaines abatteuses sont équipées d'un numériseur sur la tête d'abatage qui numérise le tronc de l'arbre en même temps qu'il abat l'arbre. Cette technique fournit une numérisation d'une tige qui peut

ensuite être recoupée en plusieurs billots. Dans de nombreux cas, les billots proviennent de bases de données de billots utilisées pour la configuration ou la simulation de la scierie ou pour la prise de décision basée sur l'optimisation.

La figure 3.1 illustre un nuage de points brut pour un billot spécifique. Un tel nuage de points possède quelques particularités. Le nuage de points est composé de tranches (ou sections). Chaque tranche est le résultat d'une lecture à distance fixe de l'origine du billot. Lors de la numérisation, les tranches contiennent un grand nombre de points. Un algorithme retire les points les moins significatifs dans le but de réduire le nombre de points et faciliter l'analyse du billot. Cependant, il n'y a aucune garantie sur le nombre de points utilisés pour représenter une tranche. Dans notre jeu de données de billots réels numérisés dans une scierie par notre partenaire. Dans ce jeu de données, le nombre moyen de points est de 18 474 points, mais certains billots ont moins de 5 000 points et d'autres plus de 25 000. Ces différences sont généralement expliquées par le type de machine utilisée pour la numérisation et par les caractéristiques du billot. Par exemple, un billot plus long possède généralement plus de points qu'un billot plus court. Une autre particularité de cette représentation est que la résolution du nuage de point n'est pas constante. Effectivement, des tranches peuvent être manquantes et les points d'une tranche ne sont pas uniformément répartis sur la surface du billot.

FIGURE 3.1 – Représentation d'un billot en utilisant un nuage de points fournis par les numériseurs de billots.

3.1.2 Nuages de points normalisés

Notre seconde représentation du billot implique de normaliser les *nuages de points bruts*. Nous normalisons les nuages de points bruts en assurant un nombre constant de points dans chaque tranche et une distance constante séparant les tranches. Notre algorithme de normalisation procède en deux étapes, d'abord il construit des tranches normalisées, puis il construit un billot complet à partir des tranches normalisées. La *normalisation du nuage de points* est obtenue suite aux étapes suivantes.

1. Les points d'une tranche sont connectés pour former une boucle. Ensuite, de nouveaux points sont répartis uniformément autour de la tranche.

2. Le centroïde de la tranche est calculé en faisant la moyenne de la position des nouveaux points.
3. Le billot est placé de manière à ce que l'axe Z représente la longueur du billot alors que la courbure est mesurée par rapport à l'axe des Y.
4. Le point de référence est choisi de manière à être intersecté avec l'axe Y et la périphérie de la tranche telle que définie par sa *spline* de base (ou *B-spline*).
5. En supposant que nous souhaitons avoir H points par tranche, nous positionnons tous les points à l'intersection du segment de ligne du centroïde à la périphérie de la tranche *B-spline* dans la direction $\frac{2\pi i}{H}$ où i est le point i^{th} dans le sens des aiguilles d'une montre sur la périphérie de la tranche.

Nous avons choisi la B-spline puisqu'elle convient à l'ajustement de surface (Dierckx, 1995). Une fois le nombre de points par tranche normalisé, l'algorithme procède à la construction du nuage de points normalisé en créant de nouvelles tranches et en alignant les points de chaque tranche. À cette fin, nous créons W nouvelles tranches contenant chacune H points. Dans nos expériences, $H = 128$ points par tranche a fourni de bons résultats. De plus, nous avons déterminé que la longueur maximale d'un billot doit être inférieure à 560 cm. Nous avons choisi $W = 128$ tranches ce qui revient à fixer la distance entre chaque tranche à 4,73 cm. Pour positionner les nouveaux points sur la longueur, nous créons une seconde B-Spline à partir des points H_i pour chacune des tranches (c.-à-d. relier les points de toutes les tranches pour un angle donné autour de l'axe de référence). Comme nous désirons un nombre constant de points par billot, des tranches factices sont ajoutées pour allonger les billots plus courts, c'est-à-dire que les billots avec moins de W tranches sont remplis avec des tranches contenant des points ayant une valeur de zéro. Un exemple de projection est présenté dans la figure 3.2.

3.1.3 Projections 2D

Les projections 2D du billot nous permettent de représenter un billot comme étant une matrice de pixels en deux dimensions. L'idée est d'encoder les nuages de points normalisés représentant les billots en 3D de sorte que les informations qu'ils contiennent soient plus facilement accessibles aux réseaux de neurones destinés au traitement des images 2D. Intuitivement, on peut voir la forme 3D enveloppante d'un billot comme étant un cylindre. Il est possible de visualiser une ligne au centre du cylindre reliant les centroïdes aux deux extrémités du billot. L'information qui nous intéresse est la distance entre le centre du cylindre et la surface du billot sous différents angles. Pour effectuer la projection, nous utilisons le nuage de points normalisé décrit dans la section 3.1.2 pour produire une matrice de pixels H par W où chaque pixel représente la distance du centre du cylindre pour un angle donné. Sauf mention contraire, tous les billots sont alignés sur la même extrémité de la matrice de pixels et déroulés/dépliés selon un axe standard. Par conséquent, pour les billots plus courts que la longueur maximale du billot, nous insérons une série de pixels noirs à une des extrémités de la projection.

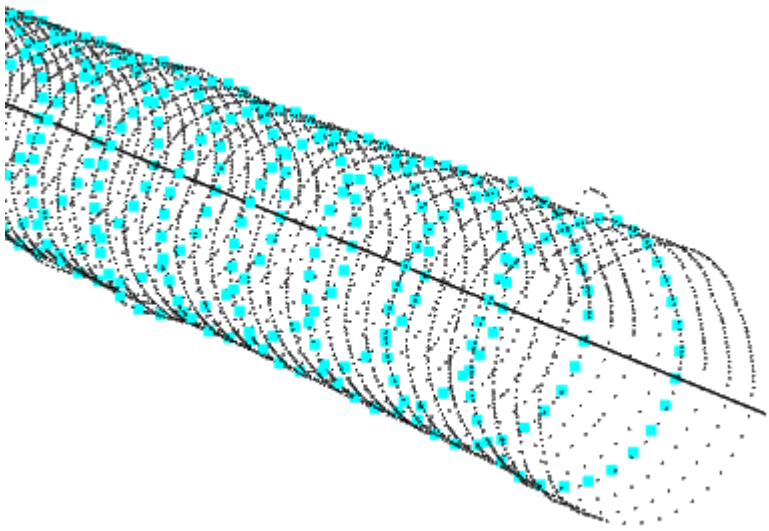

FIGURE 3.2 – Représentation d'un nuage de points normalisé (en bleu) représentant un billot par rapport au nuage de points brut du billot.

La figure 3.3 montre une représentation de la projection du billot présenté dans la figure 3.1. Les pixels sont agrandis pour mieux montrer leurs nuances de gris. Pour un pixel donné, une nuance de gris plus claire indique que le point est plus éloigné du centre du cylindre et une nuance de gris plus foncée indique qu'il est plus proche du centre. Une couleur noire indique une distance nulle. Pour avoir un niveau de blanc constant, nous avons utilisé un rayon de référence maximal de 40 cm qui est supérieur au diamètre maximal des billots dans notre jeu de données industrielles.

Un inconvénient de l'approche est que les billots avec une courbure très prononcée conduiraient à un axe de référence qui est à l'extérieur de certaines tranches conduisant à une erreur dans la mesure du rayon. Malgré cette restriction, cette représentation permet de transmettre les mêmes informations que des nuages de points 3D normalisés avec une couche d'entrée 66% plus petite. En effet, pour un nuage de points, nous devons fournir trois valeurs numériques pour chaque point (les coordonnées en x, y, z). Avec la projection, nous transformons le nuage de points en matrice 2D qui ne contient qu'une valeur numérique (la distance au centre).

Dans les prochaines sections, nous décrivons les variantes de nos approches de projection 2D.

Options de remplissage pour les données

Pour tenter d'atténuer la perte d'informations le long des points de référence choisis pour dérouler le billot, nous proposons un remplissage circulaire et un remplissage alterné à deux couches.

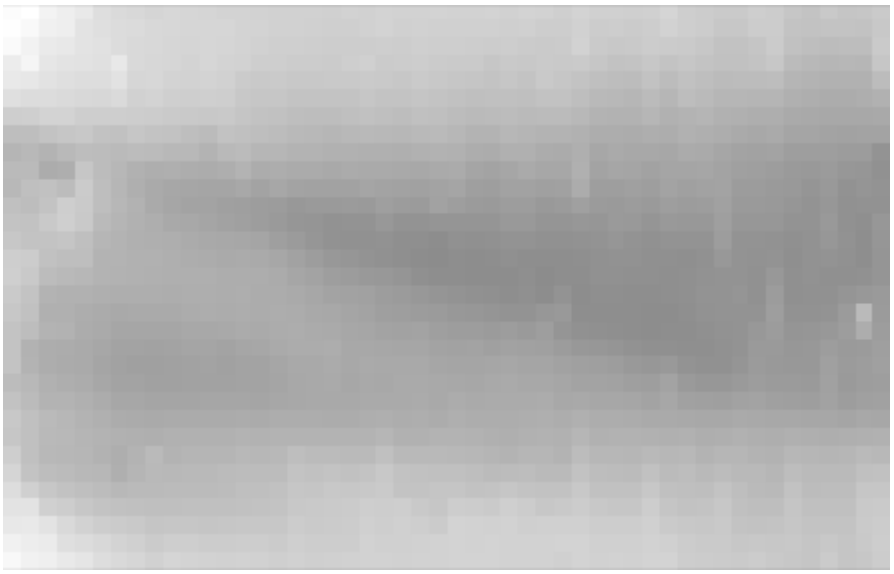

FIGURE 3.3 – La matrice de pixels permettant de visualiser une projection en deux dimensions d'un billot. Chaque pixel représente la distance entre le centre du billot et sa surface. Une ligne représente l'information pour un angle donné. Une colonne représente l'information pour une distance donnée prise sur la longueur à partir de l'origine du billot.

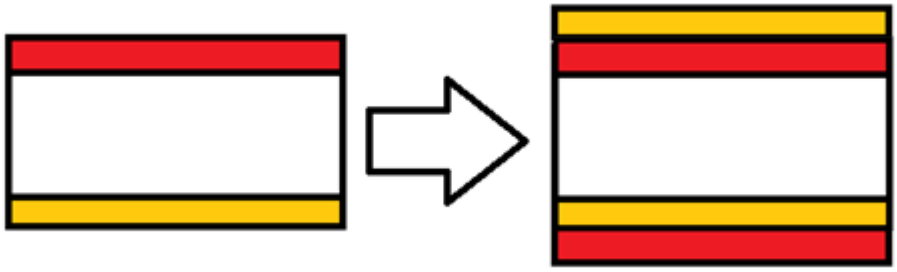

FIGURE 3.4 – Un exemple de remplissage circulaire pour un billot. Le billot est le même que celui présenté dans 3.3.

En *remplissage circulaire*, on duplique une partie de l'extrémité supérieure (resp. inférieure) de la projection et on la concatène à l'extrémité opposée. La figure 3.4 montre comment le transfert de données est effectué pour le haut et le bas de la projection. En jaune (resp. rouge), on voit la portion de la projection 2D que l'on copie du bas (resp. haut) de la matrice de pixels de la projection 2D vers le haut (resp. bas). Dans le cas de nos expériences, nous copions 10% des données.

En *remplissage alternatif à deux couches*, chaque point est représenté exactement deux fois pour conserver toutes les informations autour de la ligne de partage. Pour ce faire, nous créons deux projections : la première est la même que celle présentée dans la section 3.1.3, l'autre est composée d'une rotation de 180 degrés sur l'axe de la longueur. Les deux projections sont

FIGURE 3.5 – Un une projection à deux couches pour un billot. Le billot est le même que celui présenté dans 3.3.

placées dans deux canaux. Il s'agit du même principe que les canaux de couleur pour une image. De cette façon, toutes les informations sont présentes exactement deux fois et chaque point de la surface du journal est représenté dans son contexte entier. La figure 3.5 représente une simplification du processus. Dans cet exemple, nous utilisons deux couches qui correspondent à deux projections. La deuxième projection montre la projection du même billot pivoté de 180 degrés. Cette représentation permet une couche d'entrée 33% plus petite que celle du nuage de points normalisé. Cette fois, la taille des données que nous fournissons au réseau de neurones est plus grande du au fait que nous devons doubler les projections 2D définies dans la section 3.1.3, mais nous n'avons besoin que de deux valeurs numériques pour chaque point plutôt que trois comme c'est le cas avec les nuages de points.

Augmentation de données

Considérant que les réseaux de neurones à convolution sont sensibles à la translation, nous avons implémenté l'augmentation des données pour fournir plus d'exemples lors de l'entraînement. *L'augmentation des données* consiste à générer des exemples d'apprentissage supplémentaires en effectuant des modifications aléatoires sur les exemples réels de l'ensemble d'apprentissage (Wang, Perez et al., 2017). Dans notre contexte spécifique, nous effectuons une translation aléatoire de la matrice de pixels non nuls le long de la largeur de la projection. Plus simplement, nous déplaçons un nombre aléatoire de tranches de remplissage de la droite vers la gauche de la matrice contenant la projection. Lors de l'augmentation des données, une rotation aléatoire des billots est également effectuée le long de leur axe de longueur. Nous effectuons une augmentation des données avant de procéder au remplissage circulaire ou alterné à deux couches. Ces étapes sont effectuées lors de l'entraînement et seront différentes à chaque fois qu'un billot est échantillonné pour l'entraînement.

3.1.4 Données paramétriques

Les caractéristiques basées sur le savoir-faire de l'industrie des produits forestiers, également appelées données paramétriques, sont des caractéristiques conçues par des ingénieurs forestiers pour caractériser les billots. Dans le cadre de nos travaux, nous utilisons le diamètre fort (cm)

[1]Pour calculer le diamètre fort et faible, il faut calculer le diamètre aux deux extrémités du billot. Le diamètre fort est le plus grand des deux diamètres et le diamètre faible sera le plus petit des deux.

Tableau 3.1 – Toutes les combinaisons de représentations des billots (en lignes) et d'architectures de réseau de neurones (en colonnes) ; tous les modèles sont testés à la fois en classification et en régression.

	MLP	**ResNet**	**PointNet**
Nuage de Points	–	–	✓
Nuage de Points Normalisé	✓	–	–
Projections 2D			
Régulier	✓	✓	–
Remplissage Circulaire	✓	✓	–
Remplissage Alterné	✓	✓	–
Projections 2D avec Augmentation			
Régulier	✓	✓	–
Remplissage Circulaire	✓	✓	–
Remplissage Alterné	✓	✓	–
Paramétrique	✓	–	–

et faible (cm)[1], la longueur (m), le volume (dm^3), une mesure de la courbure du billot (cm/m), et la conicité (cm/m). Ces caractéristiques peuvent être calculées à partir des nuages de points 3D bruts à l'aide d'algorithmes spécifiques sans nécessiter de simulation de sciage.

3.2 Expérimentations

Le but de nos expérimentations est de déterminer la combinaison d'architecture et de représentation du billot qui donne les meilleurs résultats. Les modèles de réseaux de neurones que nous utilisons sont les perceptrons multicouches, les réseaux de type *Resnet* (He et al., 2016) ainsi que les réseaux de type *PointNet* (Qi et al., 2017). Les représentations utilisées sont les données paramétriques, les projections 2D avec les différents types de remplissage et les nuages de points qui peuvent être normalisés ou non. La table 3.1 montre les différentes combinaisons possibles qui combinent les architectures de réseaux de neurones et des représentations. Nous avons testé un total de 15 combinaisons d'architecture et de représentation du billot. Considérant que nous explorons les résultats dans un contexte de régression et de classification, nous avons un total de 30 variantes.

Dans ce qui suit, nous fournissons plus de détails sur le processus que nous avons utilisé pour former les modèles et comment nous évaluons les performances des modèles.

3.2.1 Entraînement des réseaux

Pour nous assurer que nos observations se généralisent bien, nous avons partitionné de manière aléatoire notre ensemble de données industrielles afin que 85% des billots soient utilisés pour entraîner les modèles et 15% pour les évaluer. L'augmentation des données est effectuée de manière à ce que les échantillons ne puissent pas « couler » de l'ensemble de données d'appren-

tissage vers l'ensemble de données de test. De plus, sauf mention contraire, les scores rapportés correspondent à la moyenne de 10 exécutions sur de telles partitions. Les partitions utilisées sont les mêmes que celles utilisées dans le chapitre 2.

Pour entraîner nos modèles, nous utilisons l'optimiseur *Adam* et nous autorisons 500 épisodes (epochs) d'entraînement. Pendant l'entraînement, nous réduisons le taux d'apprentissage à chaque fois que nous atteignons 10 *epochs* sans que le réseau ne s'améliore. Finalement, nous utilisons l'arrêt précoce pour réduire le surapprentissage. L'entraînement s'arrête lorsque le taux d'apprentissage est réduit deux fois sans obtenir de gain lors de l'apprentissage. Le taux d'apprentissage initial est fixé à 0,0001.

3.3 Résultats et discussion

Les expériences ont été menées en deux phases. Dans la première phase, nous comparons nos multiples variantes de projection 2D pour voir si une variante surpasse les autres (section 3.3.1). Puis, dans une deuxième phase, nous comparons les combinaisons d'architecture et de représentation (section 3.3.2). Dans la seconde phase, nous n'utilisons que les meilleures projections 2D à l'issue de la première phase.

3.3.1 Phase 1 : Comparaison de la performance pour les Projections 2D

Le tableau 3.2 compare les performances, en termes de métrique F1 moyenne pour les différentes variations des projections 2D en entrée du réseau. Pour chaque représentation, nous évaluons aussi les effets de l'augmentation de données. Pour évaluer la performance, nous utilisons les perceptrons multicouches et l'architecture proposée par *ResNet*. Évidemment, dans ce contexte, nous n'utilisons pas l'architecture *PointNet* puisque cette dernière n'est pas en mesure de traiter les projections 2D et demande un nuage de points. Pour toutes les approches, nous évaluons la prédiction du modèle en mode régression et classification. Les projections 2D régulières, c'est-à-dire sans remplissage, sont comparées au remplissage circulaire et au remplissage alterné à deux couches. En plus de comparer les représentations, nous comparons aussi les approches à la prédiction moyenne correspondant à notre jeu de données.

Notre premier constat est que l'augmentation des données aide les algorithmes d'apprentissage à obtenir une mesure F1 moyenne plus élevée dans le cas de la classification. Nous remarquons une amélioration d'au moins 8% lorsque l'augmentation des données est utilisée pour la classification et les perceptrons multicouches. Cependant, pour la régression, il ne semble pas y avoir d'avantage aussi important à utiliser l'augmentation des données. Dans ce contexte, nous remarquons un gain avoisinant les 2%. Cette observation est vraie pour tous les types de projection 2D que nous avons développés. Il est aussi possible de constater que les perceptrons multicouches sont plus performants en classification qu'en régression. Dans ce cas, le gain est de plus de 12%. Pour les réseaux de neurones de type *ResNet* le gain d'utiliser l'augmentation

Tableau 3.2 – Valeurs de la métrique F1 moyenne des réseaux de neurones à l'aide de nos différentes projections 2D (avec des intervalles de confiance à 95%) ; pour chaque paire d'architectures et de types de sortie, la meilleure valeur moyenne de la métrique F1 moyenne est surlignée en gris.

	Sans augmentation de données		**Avec augmentation de données**	
	Classification Mesure F1	Régression Mesure F1	Classification Mesure F1	Régression Mesure F1
Prédiction moyenne	0,148 ± 0,002	0,148 ± 0,002	0,148 ± 0,002	0,148 ± 0,002
MLP				
Régulier	0,481 ± 0,010	0,416 ± 0,007	0,544 ± 0,012	0,416 ± 0,0125
Remplissage circulaire	0,480 ± 0,009	0,421 ± 0,007	0,541 ± 0,007	0,422 ± 0,010
Remplissage alterné	0,492 ± 0,009	0,418 ± 0,012	0,543 ± 0,006	0,417 ± 0,008
ResNet				
Régulier	0,436 ± 0,008	0,466 ± 0,008	0,483 ± 0,012	0,473 ± 0,007
Remplissage circulaire	0,451 ± 0,009	0,463 ± 0,0081	0,479 ± 0,015	0,479 ± 0,007
Remplissage alterné	0,450 ± 0,013	0,466 ± 0,013	0,485 ± 0,013	0,469 ± 0,004

de données est d'environ 1%.

Nous remarquons aussi que pour une architecture donnée (avec ou sans augmentation de données) et un mode d'apprentissage (régression ou classification), nous ne pouvons pas conclure qu'il existe une différence significative en termes de métrique F1 moyenne entre les représentations que nous avons développées. Néanmoins, nous remarquons que les performances moyennes sont plus élevées en utilisant les perceptrons multicouches en classification alors que l'architecture *ResNet* est plus performante en régression.

En ce qui concerne les performances de *ResNet* par rapport à celles du perceptron multicouche. Il y a une différence pouvant aller jusqu'à 0,06 point dans le cas des projections 2D et l'augmentation de données. Ces résultats sont tout de même surprenants. En effet, nous pouvons considérer les projections 2D comme étant une image en noir et blanc et nous savons que les réseaux *ResNet* sont généralement plus performants que les perceptrons multicouches dans le traitement des images (He et al., 2016). Cependant, dans notre cas, il est possible qu'il soit difficile d'extraire des motifs de nos projections 2D et que les convolutions utilisées dans les réseaux *ResNet* ne soient pas pertinentes.

3.3.2 Phase 2 : Comparaison des performances en fonction des architectures et des représentations de données

Le tableau 3.3 montre une comparaison entre les différents modèles, représentations que nous avons utilisées pour nos expériences. La valeur moyenne de la métrique F1 moyenne, la précision et le rappel y sont présentés. Pour les projections 2D, nous avons retenu les approches les plus performantes présentées dans la section 3.3.1. Il s'agit du remplissage circulaire avec augmentation de données pour le perceptron multicouche et du remplissage régulier avec aug-

Tableau 3.3 – Valeurs moyennes de la métrique F1 moyenne (avec des intervalles de confiance à 95%), précision et rappel moyens pour toutes les architectures de réseau de neurones et les différentes représentations de données. Pour chaque paire d'architectures et de types de sortie, la meilleure valeur moyenne de la métrique F1 moyenne est surlignée en gris.

	Classification			**Régression**		
	Mesure F1	Prec.	Rap.	Mesure F1	Prec.	Rap.
Prédiction moyenne	0,148 ± 0,002	0,158	0,152	0,148 ± 0,002	0,158	0,152
MLP						
Données paramétriques	0,512 ± 0,013	0,514	0,532	0,423 ± 0,006	0,464	0,401
Remplissage circulaire	0,541 ± 0,007	0,535	0,574	0,422 ± 0,010	0,462	0,402
Points normalisés	0,517 ± 0,009	0,514	0,546	0,421 ± 0,008	0,461	0,400
PointNet						
Nuage de points	0,546 ± 0,017	0,541	0,577	0,457 ± 0,007	0,475	0,454
ResNet						
Projection 2D	0,483 ± 0,012	0,486	0,501	0,473 ± 0,007	0,514	0,450

mentation de données pour *ResNet*.

Toutes les représentations que nous avons utilisées fonctionnent nettement mieux que le modèle de référence prédisant la moyenne de l'ensemble d'apprentissages. Sur nos données industrielles, nous voyons de forts avantages à utiliser des réseaux de neurones plutôt qu'à utiliser une production moyenne pour une usine donnée. Nous observons que l'approche la plus performante est obtenue lorsque nous utilisons *PointNet* avec les nuages de points en classification. Alors qu'en régression, il s'agit des projections 2D avec l'approche *ResNet*. Considérant que ces résultats sont supérieurs aux résultats obtenus avec les données paramétriques, il est crédible de croire qu'une représentation plus complexe du billot peut potentiellement aider à la prédiction du panier de produits.

En général, les modèles de classification fonctionnent nettement mieux que les modèles de régression. Ce comportement est constant pour tous les modèles. Comme on peut le voir dans le tableau 3.3, les valeurs moyennes de précision et de rappel sont différentes selon que l'on utilise le réseau en mode classification ou régression. Fait intéressant, tous les modèles de classification obtiennent un rappel plus élevé alors que pour tous les modèles de régression, nous observons le contraire. Nous concluons que nos modèles de régression ont tendance à sous-estimer la production alors que nos modèles de classification ont tendance à la surestimer. Ce comportement pourrait s'expliquer par le fait qu'une classe doit être convertie en un vecteur de comptes pour calculer les métriques de précision et de rappel. Il est donc possible que certains produits de sciage soient ajoutés lors de la conversion si le billot est similaire à un autre billot avec plus de produits. Pour la régression, il n'y a aucune contrainte pour prédire un nombre exact pour chaque compte. Dans ce cas, nous pensons qu'il est possible que les

réseaux tentent de prédire de petites valeurs pour la majorité des produits de sciage possible alors qu'ils devraient prédire un compte de 0 puisqu'il n'y a pas de produits.

3.4 Conclusion

Dans cette recherche, nous avons développé et évalué 30 combinaisons d'architectures de réseaux de neurones, de représentations des billots, de modes de sortie et de processus d'entraînement pour le problème de prédiction de la production de bois. Nos recherches montrent bien qu'il est possible de prédire le panier de produits associé à un billot grâce aux réseaux de neurones. Nos résultats montrent également qu'il est possible d'exploiter les différentes représentations que nous avons développées. Toutefois, les réseaux de neurones ayant accès aux représentations complexes performent généralement mieux à défaut de demander plus de calculs.

Nos expérimentations ont montré que les réseaux de types *PointNet* sont les réseaux qui obtiennent la meilleure qualité de prédiction en classification avec une valeur de 0,546 pour la mesure F1. Lorsque nous sommes en régression, les réseaux de types *ResNet* sont les plus performants avec une valeur de 0,473 pour la métrique F1.

De plus, nous avons observé que les résultats sont meilleurs en classification qu'en régression. Il serait possible de croire que l'approche en classification ne soit pas fonctionnelle puisqu'il pourrait y avoir une infinité de classes. Cependant, une scierie a une capacité limitée de transformation des billots. Dans la réalité, la scierie est limitée par la taille des billots qu'elle peut accepter et par la capacité des machines faisant les transformations. Il est faux de dire qu'il y a une infinité de classes possibles. Il serait donc intéressant de faire une analyse sur le nombre réel de classes possible par rapport à une configuration de scierie et d'évaluer la performance de la classification par rapport au nombre de classes réalisables.

Chapitre 4

Comparaison des réseaux de neurones aux autres approches d'apprentissage automatique

Au chapitre 3, nous avons présenté les résultats de nouvelles approches par réseaux de neurones pour la prédiction d'un panier de produits associé à un billot. Nos approches explorent différentes représentations du billot ainsi que diverses architectures du réseau. Au chapitre 2, nous avons présenté la qualité de prévision par lot d'approches « classiques » d'apprentissage automatique dont certaines avaient été originalement évaluées par Morin, Paradis et al. (2015) pour la prévision par billot. Dans le présent chapitre, nous complétons l'analyse des approches par réseau de neurones par une comparaison aux approches « classiques » d'apprentissage automatique à la fois en prévision par billot et par lot.

Dans un premier temps, nous comparons les prédictions par billot (section 4.1). La seconde étape sera de comparer les prédictions pour les lots pour les différents modèles (section 4.2). Avec ces résultats, nous pouvons observer que, dans un contexte de prédiction par billot, les réseaux de neurones utilisant les nuages de points performent mieux en classification que les autres approches d'apprentissage automatique alors que les forêts d'arbres décisionnels sont toujours l'approche la plus performante en régression. En ce qui concerne la prédiction par lot, nos résultats montrent que les réseaux de neurones ne sont pas plus performants que les autres approches. Certaines solutions à cette lacune des réseaux de neurones seront explorées dans les chapitres subséquents.

4.1 Comparaison des prédictions par billots

Les résultats d'expérimentations de Morin, Gaudreault et al. (2020) nous montrent qu'il est possible, avec la prédiction par billot, d'augmenter la performance pour le problème de l'as-

Tableau 4.1 – Valeurs de la métrique F1 moyenne pour les différentes approches d'apprentissage automatique (réseaux de neurones et des approches dites « classiques ») et représentation possible pour le billot (avec des intervalles de confiance à 95%) ; pour chaque paire d'architectures et de types de sortie, la meilleure valeur moyenne de la métrique F1 est surlignée en gris.

		Classification	**Régression**
Entrée	**Algorithme**	**F1**	**F1**
Paramétrique	Moyenne	0,1481 ± 0,0022	0,1481 ± 0,0022
	Arbre Décision	0,5042 ± 0,0098	0,5257 ± 0,0079
	k-NN($k = 1$)	0,4483 ± 0,0084	0,4482 ± 0,0084
	k-NN($k = 10$)	0,4379 ± 0,0122	0,4109 ± 0,0052
	k-NN($k = 20$)	0,4391 ± 0,0117	0,3813 ± 0,0052
	k-NN($k = 30$)	0,4333 ± 0,0096	0,3606 ± 0,0059
	k-NN($k = 40$)	0,4228 ± 0,0131	0,3471 ± 0,0057
	Forêt d'arbres	0,5262 ± 0,0097	0,4993 ± 0,0046
	MLP	0,5118 ± 0,0131	0,4234 ± 0,0064
Proj.2D	MLP	0,4813 ± 0,0097	0,4156 ± 0,0073
	Resnet	0,4362 ± 0,0079	0,4657 ± 0,0075
Proj.2D(aug)	MLP	0,5435 ± 0,0117	0,4156 ± 0,0125
	Resnet	0,4827 ± 0,0119	0,4726 ± 0,0065
Proj.2Dpad	MLP	0,4795 ± 0,0088	0,4214 ± 0,0066
	Resnet	0,4510 ± 0,0089	0,4630 ± 0,0081
Proj.2Dpad(aug)	MLP	0,5407 ± 0,0068	0,4217 ± 0,0095
	Resnet	0,4785 ± 0,0151	0,4790 ± 0,0070
Proj.2D2couches	MLP	0,4922 ± 0,0088	0,4180 ± 0,0121
	Resnet	0,4503 ± 0,0125	0,4660 ± 0,0125
Proj.2D2couches(aug)	MLP	0,5428 ± 0,0057	0,4174 ± 0,0080
	Resnet	0,4848 ± 0,0129	0,4691 ± 0,0040
Nuage de points	PointNet	0,5464 ± 0,0168	0,4565 ± 0,0069
Points Normalisés	MLP	0,5171 ± 0,0093	0,4208 ± 0,0079
	Resnet(3DConv.)	0,4968 ± 0,0167	0,4693 ± 0,0077

signation des blocs de coupe par rapport à une méthode plus simple utilisant la moyenne historique (Morin, Gaudreault et al., 2020). Ce point illustre l'importance d'une prédiction par billot de qualité. La table 4.1 présente une comparaison de la qualité de la prédiction pour les billots individuels tant pour les méthodes de réseaux de neurones (résultats présentés au chapitre 3) que les approches « classiques » d'apprentissage automatique (résultats présentés au chapitre 2). Rappelons que la table contient la valeur correspondant à la moyenne de 10 réplications pour la métrique F1 moyenne de l'ensemble de tests.

De manière générale, les algorithmes ont plus de facilité en classification qu'en régression. Cette observation est valide tant pour les réseaux de neurones que pour les autres approches présentées dans le chapitre 2. En classification, nous observons une métrique F1 moyenne maximale de 0,5464 pour l'approche utilisant les nuages de points et les réseaux *PointNet*. En revanche, pour la régression, nous observons une valeur F1 moyenne maximale de 0,5257 pour l'approche utilisant les arbres de décisions.

Nous remarquons aussi qu'aucune des approches ne se démarque particulièrement. Il aurait été facile de croire qu'en utilisant des représentations du billot contenant plus d'information et des réseaux de neurones, nous aurions automatiquement de meilleurs résultats. Cependant, nous remarquons que bien que l'approche la plus performante soit un réseau de neurones, l'utilisation de réseaux de neurones ne garantit pas de meilleures prédictions. C'est beaucoup plus évident si nous nous attardons à la régression où ce sont les arbres de décisions qui sont les plus performants. Ces observations sont compatibles avec d'autres études qui ont montré que les réseaux de neurones ne sont pas automatiquement plus performants que les approches comme les arbres de décision et les forêts d'arbres décisionnels (Nitze et al., 2012 ; Liu et al., 2013).

4.2 Comparaison des prédictions par lots

La contribution importante présentée dans le chapitre 2 est la prédiction par lot. Ici, nous complétons les résultats présentés dans la figure 2.4 en ajoutant l'évaluation des réseaux de neurones et des différentes représentations possibles. De plus, nous explorons en détail la prédiction par lot avec des paniers de produits qui sont prédits par classification et par régression. Il est possible de voir ces résultats dans la table 4.3. Il est important de rappeler que la taille de l'ensemble de test est de 335 billots et que la métrique F1 globale est calculée sur tout l'ensemble de tests. Les résultats présentés sont aussi la moyenne de 10 réplications sur des partitions de données différentes.

Dans les résultats présentés pour la prédiction par lot dans le chapitre 2, la meilleure approche était d'utiliser les forêts d'arbres décisionnels avec les données paramétriques en régression. Dans ce contexte, nous avions une valeur moyenne de 0,9343 pour la métrique F1 globale qui combine les billots de l'ensemble de tests. Cette valeur est supérieure à celle de la prédiction moyenne qui obtient un F1 moyen de 0,9167. Les autres approches proposées dans ce chapitre et celles proposées au chapitre 3 ont une prédiction avoisinant 0,90. Ce qui se trouve à être légèrement inférieur à la prédiction moyenne. Sur un lot de grande taille, il est attendu que la prédiction moyenne d'une scierie performe bien puisque les erreurs sur un billot peuvent être compensées par un autre billot. Ce phénomène est connu sous le nom de la loi des grands nombres qui dit que la moyenne empirique s'approche de la vraie valeur lorsque la taille de l'échantillon est très grande (Dekking et al., 2005). Malgré tout, nous avons trouvé trois ap-

Tableau 4.2 – Valeurs de la métrique F1 globale sur un lot de 335 billots pour les différentes approches d'apprentissage automatique (réseaux de neurones et des approches dites « classiques ») et représentation possible pour le billot (avec des intervalles de confiance à 95%) ; pour chaque paire d'architectures et de types de sortie, la meilleure valeur moyenne de la métrique F1 globale est surlignée en gris.

		Classification	**Régression**
Entrée	**Algorithme**	**F1**	**F1**
Paramétrique	Moyenne	0,9156 ± 0,0074	0,9156 ± 0,0074
	Arbre Décision	0,9125 ± 0,0069	0,9162 ± 0,0070
	k-NN($k = 1$)	0,9044 ± 0,0064	0,9043 ± 0,0064
	k-NN($k = 10$)	0,7834 ± 0,0178	0,9157 ± 0,0046
	k-NN($k = 20$)	0,7489 ± 0,0103	0,9086 ± 0,0058
	k-NN($k = 30$)	0,7339 ± 0,0118	0,9052 ± 0,0062
	k-NN($k = 40$)	0,7190 ± 0,0116	0,9024 ± 0,0065
	Forêt d'arbres	0,7689 ± 0,0159	0,9343 ± 0,0058
	MLP	0,8699 ± 0,0083	0,8978 ± 0,0074
Proj.2D	MLP	0,8977 ± 0,0104	0,8950 ± 0,0127
	Resnet	0,8812 ± 0,0092	0,9052 ± 0,0060
Proj.2D(aug)	MLP	0,8109 ± 0,0243	0,8904 ± 0,0198
	Resnet	0,8958 ± 0,0092	0,9104 ± 0,0037
Proj.2Dpad	MLP	0,9009 ± 0,0102	0,9031 ± 0,0044
	Resnet	0,8831 ± 0,0109	0,8963 ± 0,0116
Proj.2Dpad(aug)	MLP	0,7920 ± 0,0252	0,8983 ± 0,0143
	Resnet	0,8979 ± 0,0119	0,9112 ± 0,0047
Proj.2D2couches	MLP	0,8972 ± 0,0093	0,8966 ± 0,0164
	Resnet	0,8928 ± 0,0086	0,8989 ± 0,0134
Proj.2D2couches(aug)	MLP	0,8019 ± 0,0149	0,8983 ± 0,0135
	Resnet	0,8996 ± 0,0090	0,9076 ± 0,0062
Nuage de points	PointNet	0,8165 ± 0,0204	0,9085 ± 0,0080
Points normalisés	MLP	0,8320 ± 0,0383	0,7708 ± 0,0477
	Resnet(3DConv.)	0,9034 ± 0,0071	0,9063 ± 0,0029

proches en régression capables de surpasser la prédiction moyenne. Parmi ces trois approches, une seule est significativement meilleure que la prédiction moyenne des forêts d'arbres décisionnels.

Dans ce contexte, nous observons que les résultats sont significativement moins bons pour la classification que pour la régression. Par exemple, les pour les forêts d'arbres décisionnels, nous obtenons une valeur F1 globale de 0,9359 en régression alors que cette même valeur F1 globale est de 0,7676 en classification. Ce qui se trouve à être 17,9% plus bas que la prédiction moyenne d'un panier de produits pour la taille de lot que nous utilisions. Dans les faits, en utilisant la classification pour prédire les paniers de produits, seuls les arbres de décisions offrent des résultats similaires à la prédiction moyenne avec une valeur F1 globale de 0,9125. Bien que nous n'ayons pas une explication qui explique clairement ces résultats, nous avons remarqué que les prédictions en régression peuvent fournir des fractions de produits. Ce qui n'est pas possible en classification ou tous les comptes de produits sont des entiers. Il est possible que le gain obtenu en régression provienne de toutes ces fractions de produits qui permettent d'aller chercher quelques points sur la prédiction par lot. De plus, comme nous l'avons précisé dans le chapitre 2, il est possible que nous sous-prédisions moins de produits avec la régression que nous en surprédisons avec la classification.

En classification, nous observons des valeurs de la mesure F1 globale qui sont entre 0,8 et 0,9. Nous remarquons aussi que pour les projections en 2D, l'augmentation de données et les perceptrons multicouches semblent avoir un impact négatif sur la prédiction pour le lot. Par exemple, nous observons une valeur F1 globale de 0,8977 pour le perceptron multicouche sans augmentation alors que la valeur F1 globale est de 0,8109 lorsque nous utilisons des techniques d'augmentation de données lors de l'apprentissage. Nous observons des résultats similaires pour toutes les projections 2D. Cependant, il est important de mentionner que nous n'obtenons pas ce phénomène lorsque les projections sont utilisées avec le modèle *ResNet*. Avec deux des trois projections que nous avons développées, nous obtenons des résultats supérieurs en utilisant l'augmentation de données.

Un fait intéressant est que si on se réfère à la table 4.1, nous pouvons affirmer qu'il n'y a pas une corrélation directe entre la performance sur la prédiction d'un billot et la prédiction par lot. Par exemple, les forêts d'arbres décisionnels en régression sont une des approches les plus performantes pour la prédiction par billot et pour le lot. En revanche, les projections 2D avec augmentation utilisant *ResNet* pour la régression ont généralement une performance sous la moyenne alors qu'ils font relativement bien lors de la prédiction par lot.

4.3 Conclusion

Dans ce chapitre, nous avons pu regrouper tous les résultats que nous avons obtenus au début de nos travaux et les bonifier. Pour les prédictions par billot, nous remarquons qu'il semble y

avoir un petit gain à utiliser les représentations plus complexes des billots et les réseaux de neurones lorsque nous sommes en classification. En effet, les nuages de points et l'architecture *PointNet* obtiennent une valeur de 0,5464 pour la métrique F1. En régression, les arbres de décisions obtiennent une valeur 0,5257 pour la métrique F1. Même si la classification est plus performante, toutes nos approches de régression sont plus performantes que d'utiliser la prédiction moyenne pour la scierie.

Concernant les prédictions par lot, nous avons obtenu des résultats où l'apprentissage automatique fournit des résultats similaires et pouvant être supérieurs à la prédiction moyenne. Bien que nous ayons observé un comportement intéressant où la prédiction globale pour un lot est plus performante que la moyenne, il se pourrait que cette approche ne généralise pas nécessairement pour d'autres jeux de données. Dans la majorité des approches que nous avons pu tester, il semble que la prédiction de la moyenne d'une scierie soit plus efficace pour la taille de notre jeu de test qui est de 335 billots. Selon cette analyse, il semble que la prédiction pour le lot soit généralement meilleure lorsqu'elle est effectuée en mode régression. La meilleure approche en régression est la forêt d'arbres décisionnels avec une valeur de 0,9343 pour la métrique F1 alors que la moyenne de production pour la scierie est de 0,9156.

Chapitre 5

Combinaison des approches développées pour la prédiction des produits de sciage associés à un billot

Dans ce chapitre, nous expliquons comment il est possible de combiner les algorithmes d'apprentissage automatique et les représentations proposées dans les chapitres 2, 3 et 4. Le but de combiner les algorithmes d'apprentissage automatique et les représentations est l'amélioration de la qualité de prédiction en utilisant les forces de chacune des approches. Comme le démontrent les résultats, combiner les approches ou les représentations n'améliore pas systématiquement la qualité des prédictions. Toutefois, il est possible d'obtenir des améliorations intéressantes pour la prédiction de billots avec une combinaison judicieuse d'algorithmes et de représentations. Nos résultats pour les cas où la combinaison est fonctionnelle pavent la voie pour des travaux futurs.

Le chapitre est organisé de la manière suivante. Premièrement, nous passons d'abord en revue les concepts permettant d'expliquer comment il est possible de combiner les approches précédentes (section 5.1). Par la suite, nous présentons les résultats pour les différentes approches qui nous permettent de combiner les algorithmes d'apprentissage automatique (section 5.3). Finalement, nous présentons une conclusion sur ces expérimentations (section 5.4).

5.1 Approches pour la combinaison des modèles d'apprentissage automatique

Il existe plusieurs approches en apprentissage automatique pour combiner des modèles dans le but d'améliorer la prédiction (Dzeroski et Zenko, 2002 ; Wolpert, 1992 ; Breiman, 1996). Dans ce chapitre, nous combinons les approches présentées précédemment. Pour faire l'entraînement, nous utiliserons une approche de moyenne (section 5.1.1), de sélection gourmande

(section 5.1.2), de sélection par *métamodèle* (section 5.1.3) et finalement les modèles bout en bout (section 5.1.4). Pour chacune de ces approches, nous explorons la régression et la classification.

5.1.1 Approche moyenne

Dans notre contexte, l'approche moyenne, aussi connu sous le nom de *bagging*, est une approche qui prend la prédiction de plusieurs modèles et effectue une moyenne des paniers de produits prédits par les différents algorithmes d'apprentissage automatique (Breiman, 1996).

L'approche proposée par Breiman (1996) mentionne qu'en régression, nous utilisons la moyenne de chacune des valeurs fournies en sortie. En classification, l'approche proposée utilise un vote de majorité sur la classe la plus populaire (Breiman, 1996). Dans le cadre de nos travaux, pour la classification, nous convertissons la classe en panier de produits avant de faire la moyenne de celle-ci. Cette idée est soutenue par le fait que nous avons un grand nombre de classes et qu'un vote de majorité semblait peu approprié pour notre situation.

5.1.2 Approche par sélection vorace

Cette approche se base sur l'intuition que certains modèles pourraient être meilleurs pour prédire certains types de paniers de produits ou pour mieux différencier certains attributs d'un billot. Dans ce cas, nous voulons valider s'il est possible d'ajouter un second modèle qui nous dira quelle prédiction nous devrions considérer pour un billot donnée.

Plus formellement, cette approche, inspirée de Wolpert (1992), considère N modèles entraînés pour faire une prédiction sur une même tâche. Chacun des N modèles entraînés produits une prédiction P_i qui définit un panier de produits. Par la suite, un modèle M, prenant en entrée les P prédictions, est utilisé pour tenter de réduire l'erreur de généralisation des divers modèles dans N. Le modèle M a pour but de faire la prédiction de i qui représente un indice dans N et qui permet de choisir la meilleure prédiction P_i. Le modèle M tente de prédire, pour un billot donné, lequel des N modèles a fourni la meilleure prédiction.

5.1.3 Approche sur les métaprédictions

L'approche basée sur les métaprédictions est similaire à l'approche vorace présentée précédemment. La différence majeure est que le modèle M ne prédit pas l'indice correspondant à la meilleure prédiction, mais bien le panier de produits final.

Plus formellement, on peut définir l'approche en disant que nous avons N modèles différents entraînés sur une même tâche. Chaque modèle N_i produit une prédiction P_i. Un second modèle M prend toutes les prédictions P en entrée et fournit en sortie un panier de produits. Comme présenté dans la littérature, le modèle M peut être un arbre de décision ou une régression

linéaire (Dzeroski et Zenko, 2002). Il serait possible d'utiliser d'autres approches pour le modèle M, mais nous avons préféré nous limiter à ce qui se trouvait dans la littérature.

5.1.4 Approche de bout en bout

Les approches de bout en bout ont prouvé être efficaces dans plusieurs domaines (Graves et al., 2014). L'idée de l'approche bout en bout est de faire un seul modèle qui accepte en entrée plusieurs représentations du billot qu'il peut combiner de manière à prédire un seul panier de produits. Lors de la rétropropagation (*backpropagation*), l'erreur est remontée dans l'entièreté du réseau de neurones. Le but étant que le modèle trouve lui-même les informations importantes pour faire sa prédiction en fonction des représentations en entrée (Schmidhuber, 2015).

Pour des fins de simplification, nous reprendrons uniquement les architectures de réseaux de neurones ayant été utilisées dans le chapitre 3. Nous savons qu'il est possible de connecter des couches de divers types. L'important est que la contrainte qui satisfait la taille des entrées d'une couche soit respectée par la couche précédente (Schmidhuber, 2015). Il est important que la sortie d'une couche ait la même taille que l'entrée de la couche suivante.

Nous testons des combinaisons de représentations. Soit les données paramétriques avec les projections 2D, les données paramétriques avec les nuages de points et finalement les projections 2D et les nuages de points. Afin de prendre la représentation en entrée, nous utilisons les perceptrons multicouches (MLP) pour les données paramétriques, l'architecture *ResNet* pour les projections 2D et le réseau *PointNet* pour les nuages de points. Nous considérons que ces modèles sont équivalents au niveau L_0 des approches précédentes. Par la suite, les sorties des divers réseaux sont fournies en entrée à un perceptron multicouche qui lui se charge de faire la prédiction finale du panier de produit. Ce perceptron multicouche constitue le niveau L_1. Dans ces expériences, nous avons choisi de ne pas combiner les trois représentations pour un même réseau de neurones puisque nous n'avions pas les ressources informatiques nécessaires pour mener à bien ces expérimentations. Il aurait été possible de modifier certains *hyperparamètres* des réseaux de neurones pour s'assurer de la bonne exécution, mais nous avons choisi de ne pas faire cette étape.

Afin de combiner les modèles au niveau L_0, nous avons retenu deux approches. La première est de concaténer toutes les sorties des blocs qui traitent les diverses représentations. Cette approche a montré un certain niveau de succès dans les réseaux DenseNet (Huang et al., 2017). Une seconde approche est d'additionner les sorties de chaque bloc. Cette approche est similaire à ce qui est utilisé dans le réseau *ResNet* et a montré de bons résultats pour la classification d'images (He et al., 2016).

5.2 Expérimentations

Le but de nos expérimentations est de montrer l'effet de la combinaison des différents modèles pour prédire le panier de produits pour un billot et de voir si la combinaison de divers modèles peut améliorer la qualité de la prédiction.

Dans la section qui suit, nous allons présenter les modèles que nous avons retenus des travaux précédents (section 5.2.1). Par la suite, nous regardons l'approche utilisée pour entraîner les divers modèles que nous combinons dans nos expérimentations (section 5.2.2). Finalement, nous décrivons la procédure d'entraînement pour une variation de l'approche avec métaprédiction (section 5.2.3).

5.2.1 Approches sélectionnées

Pour ce chapitre, nous choisissons parmi les approches d'apprentissage automatique présentées dans le chapitre 4. Le but étant de prendre des approches que nous savons être performantes. Cependant, nous avons aussi choisi les approches d'apprentissage automatique pour valider deux hypothèses. Nous voulons vérifier s'il est préférable de prendre plusieurs approches avec la même représentation ou de prendre des approches avec des représentations différentes.

Dans un premier temps, comme nous voulons valider si combiner des modèles utilisant la même représentation était une bonne approche, nous avons choisi de prendre tous les modèles utilisant les données paramétriques en entrée. C'est-à-dire, les arbres de décisions, les forêts d'arbres décisionnels et les k voisins les plus proches (avec $k = 1$).

En ce qui concerne les différentes représentations, les approches d'apprentissage automatique utilisées en classifications sont *PointNet* avec les nuages de points sans augmentations, les projections 2D avec des augmentations et un réseau *ResNet* et finalement les arbres de décision avec les données paramétriques. Les modèles utilisés en régression sont les forêts d'arbres décisionnels avec les données paramétriques, les projections 2D avec augmentation de données avec un réseau *ResNet* et finalement les nuages de points avec un réseau *PointNet*. À titre de référence, la table 5.2.1 résume les résultats des approches et représentations que nous avons choisi de conserver pour les prédictions individuelles. Ces résultats sont extraits de la table 4.1.

5.2.2 Entraînement des modèles

Dans nos expérimentations, nous avons plusieurs niveaux de prédiction avant d'obtenir notre prédiction finale. Nous appliquons un modèle qui agit sur la sortie d'un modèle précédent et il peut y avoir plusieurs modèles qui se succèdent. Dans ce qui suit, nous dénotons par L_0 le niveau qui prend en entrée les représentations des billots. Un niveau L_i, avec $i > 0$, utilise les prédictions au niveau L_{i-1}. Dans le cadre de ces travaux, nous limitons nos expérimentations à 2 niveaux. Nous avons donc les niveaux L_0 et L_1 (Dzeroski et Zenko, 2002).

Tableau 5.1 – Résultats en prédiction par billots des approches et représentations ayant été retenues pour la combinaison d'approches (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

		Classification	**Régression**
Entrée	**Algorithme**	**F1**	**F1**
Parametric	ArbreDecision	0,5042 ± 0,0098	0,5257 ± 0,0079
	k-NN($k = 1$)	0,4483 ± 0,0084	0,4482 ± 0,0084
	Forêt d'arbres décisionnels	0,5262 ± 0,0097	0,4993 ± 0,0046
Proj.2D(aug)	Resnet	0,4827 ± 0,0119	0,4726 ± 0,0065
Points	PointNet	0,5464 ± 0,0168	0,4565 ± 0,0069

Lors de nos expériences, les billots sont séparés de manière aléatoire de sorte que 85% des billots sont utilisés pour l'entraînement que nous appelons le jeu de données T et 15% pour valider la performance des divers modèles que nous appelons jeu de données V. Au niveau L_0 l'ensemble d'entraînement (T_0) est composé de 1900 billots alors que l'ensemble de tests (V_0) contient 335 billots.

Pour les niveaux suivants, soit i plus grand que 0, la procédure est plus complexe pour l'entraînement. Dans ce cas, les données fournies en entrée aux modèles sont les prédictions du niveau L_{i-1}. Pour entraîner les modèles au niveau L_i, nous divisons les données d'entraînement du jeu de données T_{i-1} du niveau L_{i-1} en deux partitions dont 85% pour un jeu de données T_i et 15% restants ira pour un jeu de données V_i. Le jeu de données T_i sera utilisé pour entraîner un modèle L^*_{i-1}. Le modèle L^*_{i-1} est le même modèle que L_{i-1}, mais nous l'entraînons avec un sous-ensemble du jeu de données d'entraînement question de créer un jeu d'entraînement pour le modèle L_i (V_i). Il n'est pas possible de prendre le modèle déjà entraîné pour L_{i-1} puisqu'il est possible que le modèle surapprenne le jeu de données ce qui rendrait invalide l'apprentissage pour le niveau L_i (Wolpert, 1992). Avec l'approche proposée, le modèle du niveau L_i est entraîné avec les prédictions des données du jeu de données L_0 lorsque passé dans le modèle L^*_{i-1} (Dzeroski et Zenko, 2002).

Finalement, afin de nous assurer de la reproductibilité, nous avons créé 10 partitions de ce type qui permettront de répéter les expériences 10 fois pour chacun des algorithmes. Afin de pouvoir se comparer aux expérimentations précédentes, pour ce niveau, la séparation du jeu de données au niveau L_0 est la même que pour les autres chapitres.

5.2.3 Entraînement de la variation de l'approche de métaprédiction

Lors de nos expérimentations, nous avons voulu valider une hypothèse qu'il peut y avoir un avantage d'alterner le type de sorties pour les différents niveaux. Par exemple, nous avons montré qu'il y avait des gains d'utiliser la régression au niveau L_0 et la classification au niveau

L_1. Nous avons aussi testé d'utiliser la classification au niveau L_0 et la régression au niveau L_1. À notre connaissance, ces approches n'ont pas été étudiées dans la littérature.

La principale motivation pour cette variation que nous avons développée pour combiner les approches d'apprentissage automatiques est que nous avons très peu de données lorsque nous entraînons le modèle au niveau L_1. Effectivement, nous avons peu d'exemples pour bien apprendre la correspondance entre la représentation d'un panier de produits prédits en classification (soit une distribution de probabilités) et de le convertir dans le format utilisé en régression (soit un vecteur de comptes). L'inverse est aussi vrai lorsque nous passons de la régression vers la classification. Nous avons supposé qu'avec plus d'exemples pour l'apprentissage du niveau L_1, il était possible que les résultats s'améliorent. Par conséquent, nous ne séparons pas la partition d'entraînement T_0 en T_1 et V_1. Afin de créer le jeu de données nécessaires pour apprendre le niveau L_1, nous demandons simplement au niveau L_0 de fournir une prédiction pour chacun des éléments présents dans le jeu de données T_0. Ces nouvelles prédictions seront connues comme étant un nouveau jeu de données T_0' et c'est avec ce jeu de données que nous entraînons le niveau L_1. Du coup, les modèles au niveau L_1 sont maintenant entraînés avec 1900 données plutôt que 285 comme c'est le cas pour l'approche décrite précédemment.

5.3 Expérimentations et résultats

Nous présentons maintenant les résultats pour les diverses approches de combinaison discutées précédemment, soit l'approche utilisant la prédiction moyenne, c.-à-d. *bagging* (section 5.3.1), suivit de la sélection vorace (section 5.3.2), prédiction à partir de métaprédictions (section 5.3.3) et une variation de cette approche que nous avons développée (section 5.3.4). Finalement, nous expliquons les résultats pour les approches bout en bout utilisant les réseaux de neurones (section 5.3.5).

Dans ce qui suit, pour la présentation des résultats, nous utilisons la convention suivante. Dans les graphiques, toutes les combinaisons ont un préfixe qui permet de spécifier la nature exacte de l'entrée. Le préfixe est du type $([C|R][T|V])$. Les valeurs C et R permettent de spécifier si la sortie du niveau L_0 est en classification (C) ou en régression (R). Les valeurs T et V permettent de dire si nous entraînons le modèle L_1 est entraîné sur tout l'ensemble d'entraînement (T) ou l'ensemble de validation (V). Les exemples n'ayant pas de préfixes sont les expériences originales que nous rappelons de manière à faciliter la comparaison.

5.3.1 Performance des prédictions moyennes

Cette section est utilisée pour démontrer la qualité des prédictions pour les billots lorsque nous faisons la moyenne des prédictions de plusieurs modèles. Tel que nous avons décrit dans la section 5.2.1, nous explorons des variations de la même représentation et différentes représentations dans le but de voir s'il existe un contexte qui semble plus favorable à cet exercice.

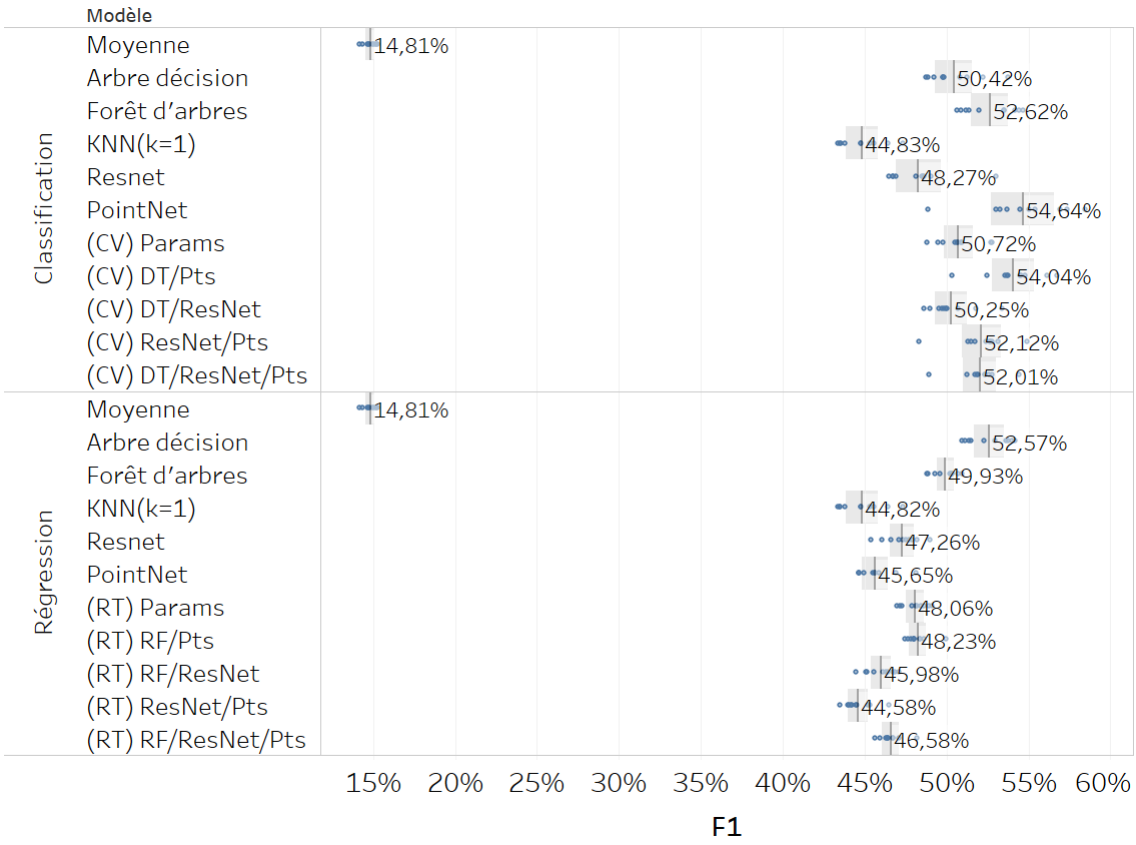

FIGURE 5.1 – Qualité de la prédiction F1 moyenne (avec des intervalles de confiance à 95%) lorsque nous faisons la moyenne des paniers de produits prédit par différents algorithmes ; les résultats présentent la moyenne de 10 réplications.

La figure 5.1 indique la valeur F1 moyenne pour les différentes combinaisons de modèles que nous utilisons dans ces travaux. Dans ce graphique, les modèles utilisés sont l'arbre de décision (DT), k-voisins les plus proches (k-NN) et forêt d'arbres décisionnels (RF). Les algorithmes utilisant les nuages de points sont marqués par Pts. Par exemple, un test identifié DT/Pts signifie que nous avons fait la moyenne des prédictions pour les arbres de décisions et les réseaux de neurones de type *PointNet* utilisant les nuages de points.

Parmi ces résultats, nous remarquons que dans une grande majorité des cas, la qualité de la prédiction pour la combinaison de plusieurs modèles est généralement moins bonne que le meilleur des modèles pris individuellement. En revanche, la moyenne des prédictions est généralement meilleure que l'approche la moins performante de la combinaison. Par exemple si nous comparons la moyenne des *PointNet* et *ResNet* en classification, nous avons une valeur de 0,5212 pour la métrique F1 moyenne alors que *PointNet* seul a une valeur de 0,5464 et que *ResNet* seul à une valeur de 0,4827. Un comportement similaire s'applique aux autres combinaisons.

Cependant, il semble y avoir une exception. En régression, si nous prenons la combinaison de *PointNet* et *ResNet*, nous pouvons observer une légère diminution de la qualité de la prédiction F1 moyenne par rapport aux modèles individuels. Cette combinaison possède une valeur de 0,4458 pour la métrique F1 moyenne alors que la métrique F1 moyenne est de pour 0,4565 *PointNet* et de 0,4726 *ResNet*. En plus de cette observation, nous pouvons remarquer que les prédictions moyennes en régression semblent s'approcher de la prédiction la moins bonne de la combinaison. En classification, nous remarquons que la prédiction tend vers la meilleure prédiction des modèles combinés sans toutefois la dépasser.

5.3.2 Performance de l'approche par sélection vorace

Cette section démontre la qualité des prédictions pour les billots individuels lorsque nous effectuons une sélection vorace, c'est-à-dire que nous choisissons la meilleure prédiction du panier de produits au niveau L_0. Tel que nous avons décrit dans la section 5.1.2, nous explorons s'il est possible de créer un second modèle qui est capable de nous dire parmi une série d'approches laquelle fait la meilleure prédiction.

La figure 5.2 indique la valeur moyenne pour la métrique F1 moyenne des différentes combinaisons des modèles sélectionnés. Par exemple, une combinaison identifiée DT/Pts, le test que nous effectuons est de fournir la prédiction des arbres de décision (DT) et les réseaux de type *PointNet* utilisant les nuages de points (Pts) à un autre arbre de décision qui nous dit si nous devons prendre la prédiction des arbres de décision ou de prendre la prédiction des forêts d'arbres décisionnels.

En explorant les résultats, nous remarquons qu'il y a quelques combinaisons utilisant la sélection vorace qui performent mieux que le meilleur modèle de la combinaison. Par exemple, en classification nous avons la combinaison entre les arbres de décisions et *PointNet* qui obtient une valeur de 0,5554 pour la métrique F1 moyenne. Ce qui est supérieur aux modèles pris individuellement. Soit, 0,5042 pour les arbres de décisions et de 0,5464 pour *PointNet*. Les autres approches en classification font similaires à la meilleure des approches. En régression nous pouvons observer des résultats similaires. Encore une fois, pour les autres modèles, les résultats sont similaires à la meilleure des approches fournies en entrée. Dans ce cas-ci, combiner les modèles semble améliorer la métrique F1 moyenne.

Cependant, il y a un phénomène intéressant qui ressort des résultats. Par exemple, en classification, nous pouvons remarquer que nous avons trois modèles avec une valeur près de 0,504 pour la métrique F1 moyenne. Ces modèles sont la combinaison des modèles utilisant les données paramétriques (Params), la combinaison DT/ResNet et la combinaison DT/ResNet/Pts. Il se trouve que la valeur 0,504 est exactement la valeur de la mesure F1 moyenne pour les arbres de décisions. Nous observons que même l'approche que nous avons prise pour entraîner le modèle au niveau L_1 (décrites à la section 5.1.2), il y a un modèle dominant qui fait en

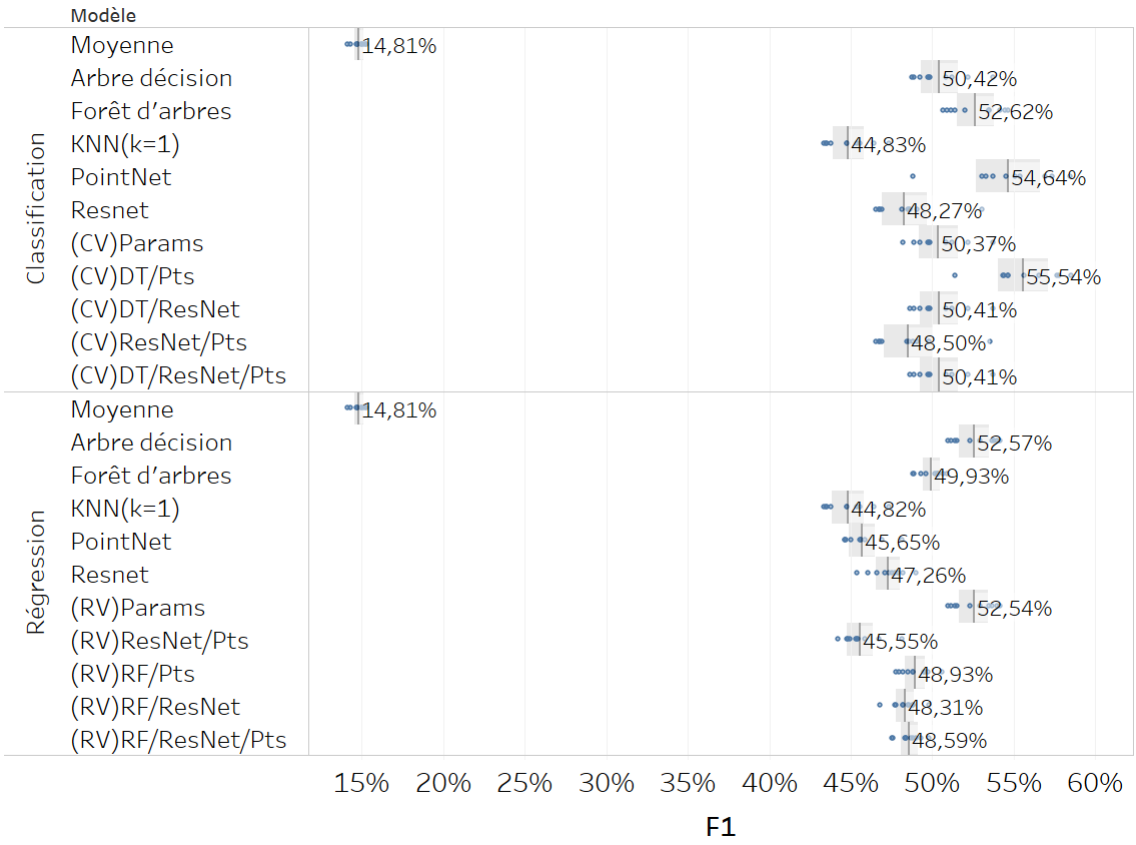

FIGURE 5.2 – Prédiction F1 moyenne (avec des intervalles de confiance à 95%) lorsque nous effectuons la sélection vorace de la meilleure prédiction pour différents algorithmes ; les résultats présentent la moyenne de 10 réplications.

sorte que le modèle de niveau L_1 prédit toujours le même algorithme comme étant le plus performant.

En analysant les données, nous constatons que la bonne étiquette est pratiquement toujours celle correspondant aux arbres de décision. Dans ce contexte, nous concluons que l'approche proposée crée un jeu de données fortement déséquilibré et que c'est possiblement ce qui explique nos résultats de toujours prédire la même approche (Guo et al., 2008). Nous avons aussi remarqué que ce comportement pouvait se répéter avec d'autres algorithmes que les arbres de décisions.

Ces résultats nous ont aussi permis de trouver une faille dans notre génération de données. Dans le cas où nous aurions une égalité dans la prédiction au moment de créer les étiquettes nécessaires à l'entraînement du niveau L_1. Nous choisissions systématiquement la première des approches valable comme étant la bonne étiquette. Une partie de la réflexion était qu'en cas de doute, nous pourrions trouver quel algorithme est le plus fiable dans ses prédictions et que nous aurions des résultats répétables. Cependant, comme nous l'avons expliqué, ce comportement

peut être en cause dans le déséquilibre du jeu de données nécessaire à l'entraînement du niveau L_0.

Il est possible de remarquer un comportement similaire en régression, mais le phénomène est un peu moins marqué. Ce qui explique un écart un peu plus grand avec le modèle dominant. En régression, le modèle qui semble être dominant est le modèle *PointNet*, qui selon l'ensemble d'entraînement du modèle au niveau L_1 est sélectionné plus de 80% du temps.

5.3.3 Performance de la prédiction par utilisation en métaapprentissage

Cette section se concentre sur l'exploration des résultats obtenus en métaapprentissage pour prédire le panier de produits. Cette technique est généralement connue sous le nom de *stacking* (Dzeroski et Zenko, 2002). Dans un premier temps, nous allons explorer les résultats en utilisant les arbres de décisions. Par la suite, nous ferons le même exercice avec les régressions linéaires telles que proposées par Dzeroski et Zenko (2002).

La figure 5.3 montre les résultats lorsque nous utilisons les arbres de décision comme *métamodèle*. Le premier constat que nous pouvons faire en regardant ces résultats est de remarquer que cette approche fournit des résultats similaires aux modèles individuels lorsqu'utilisé en classification. De manière générale, les résultats sont autour de 0,47 pour la métrique F1 moyenne et doivent se rapprocher de la moyenne pour toutes les approches. Cependant la classification montre un exemple très intéressant. La combinaison des arbres de décisions et *PointNet* a une qualité de prédiction très étendue alors que la majorité des approches ont des valeurs F1 moyenne similaire pour toutes les réplications. En effet la qualité de la prédiction pour combinaison des arbres de décisions est *PointNet* varie de 0,27 à 0,55. Ce qui peut la rendre la meilleure approche comme la pire selon la réplication. Malheureusement, nous n'avons pas été en mesure d'identifier la raison qui cause se comportement.

En ce qui concerne la régression, les résultats sont partagés. Aucune des approches n'offre de prédictions supérieures aux arbres de décisions avec une valeur de 0,5262 pour la métrique F1 moyenne. Cependant, il est possible de trouver quelques approches où la prédiction est meilleure en utilisant un *métamodèle*. Par exemple la combinaison de forêts d'arbres décisionnels et *PointNet* obtient une valeur de 0,5203 pour la métrique F1 moyenne alors que la valeur de cette métrique est de 0,4993 pour les forêts d'arbres décisionnels et de 0,4565 pour *PointNet*. Les autres combinaisons où nous pouvons remarquer ce phénomène sont les combinaisons *ResNet*/Pts, *ResNet*/RF et *ResNet*/Pts/RF. À la vue de ces résultats, nous pouvons affirmer que l'utilisation de *métamodèles* semble améliorer la qualité de la prédiction en régression.

La prochaine étape est d'analyser les résultats lorsque c'est la régression linéaire qui est utilisée comme *métamodèle*. Les résultats pour ces tests sont présentés dans la figure 5.4. Dans le cas des *métamodèles* linéaire, nous remarquons une baisse importante de la métrique F1 moyenne dans une grande majorité des cas. Seules les combinaisons de DT/Pts et *ResNet*/Pts

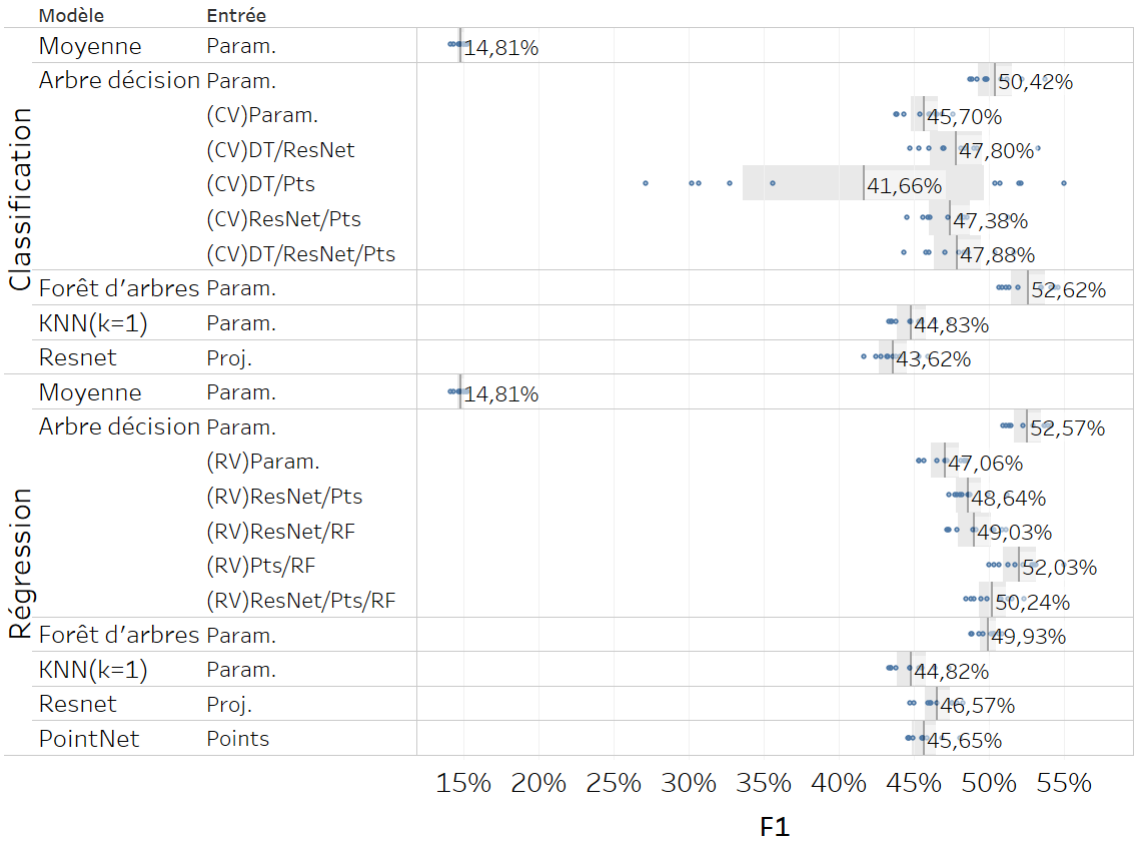

FIGURE 5.3 – Exploration de la qualité de la prédiction F1 moyenne (avec des intervalles de confiance à 95%) pour diverses combinaisons lorsque le *métamodèle* interprétant les prédictions au niveau L_0 est un arbre de décision ; les résultats présentent la moyenne de 10 réplications.

maintiennent une qualité de prédiction similaire aux algorithmes individuels lorsque la classification est utilisée. En régression, nous remarquons une baisse de la qualité pour toutes les approches utilisant la régression linéaire pour faire la prédiction finale.

Contrairement à ce que montre Dzeroski et Zenko (2002) l'utilisation de *métamodèles* n'aide pas nos prédictions. Il y a deux facteurs qui peuvent expliquer ce phénomène. Premièrement, dans l'article de Dzeroski et Zenko (2002), nous remarquons que les algorithmes au niveau L_0 ont généralement une dimension inférieure à 5. Dans notre cas, la dimension du vecteur de sortie au niveau L_0 sera soit de 85 ou 968 si nous sommes en régression ou en classification. La grande différence de taille pourrait être un indice qui explique la faiblesse de nos résultats.

Un second point est que pour entraîner les modèles au niveau L_1, nous n'avons accès qu'à 15% de la taille du jeu de données en entraînement. Ce qui nous donne uniquement 285 exemples pour faire l'apprentissage de nos modèles. Il serait crédible que ce nombre d'exemples soit trop petit pour bien apprendre. Il faut rappeler que dans le meilleur des cas, avec 285 exemples, on souhaite apprendre à faire un lien entre 170 éléments en entrée et 85 en sortie. Ce qui fait

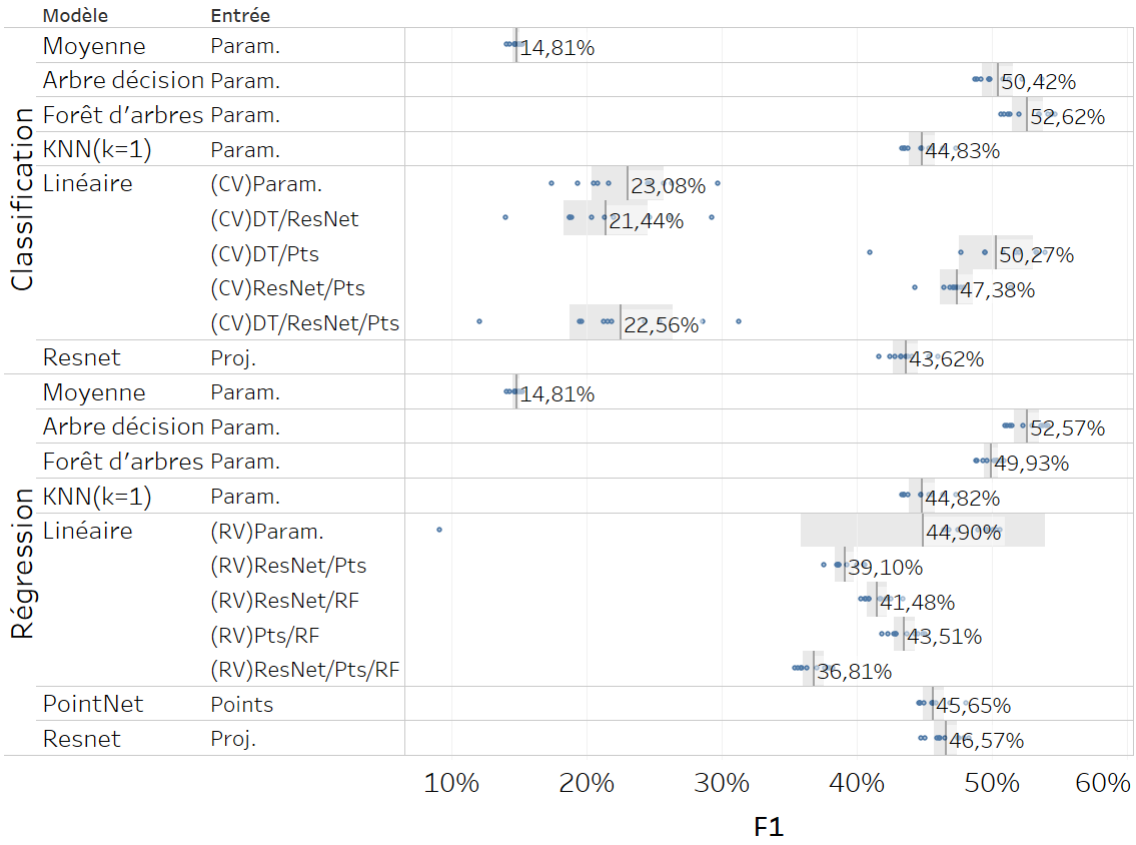

FIGURE 5.4 – Qualité de la prédiction F1 moyenne (avec des intervalles de confiance à 95%) lorsque nous utilisons l'approche par *métamodèle* utilisant la régression linaire pour faire la prédiction finale sur une variété de combinaisons ; les résultats présentent la moyenne de 10 réplications.

un espace large pour peu de données.

5.3.4 Variation de l'approche utilisant la prédiction par *métamodèles*

Dans cette section, nous comparons premièrement l'effet d'utiliser tout le jeu d'entraînement pour l'entraînement des modèles du niveau L_1. Par la suite, nous explorons l'effet de passer de la régression vers la classification pour valider notre hypothèse que cela pourrait améliorer nos résultats.

Dans la figure 5.5, nous observons la qualité de la mesure F1 dans le but de valider l'hypothèse qu'ajouter des exemples dans l'ensemble d'entraînement aide les algorithmes à mieux apprendre. Nous pouvons observer que les résultats obtenus sont généralement supérieurs à ce qui a été présenté dans la section 5.3.3 à l'exception de l'approche qui combine toutes les données paramétriques en régression. Un point intéressant en régression est que les combinaisons *Pts*/RF (0,5375) et *ResNet*/Pts/RF (0,5277) utilisant les arbres de décisions comme *métamodèle* offrent maintenant une qualité de prédiction supérieure aux arbres de décisions

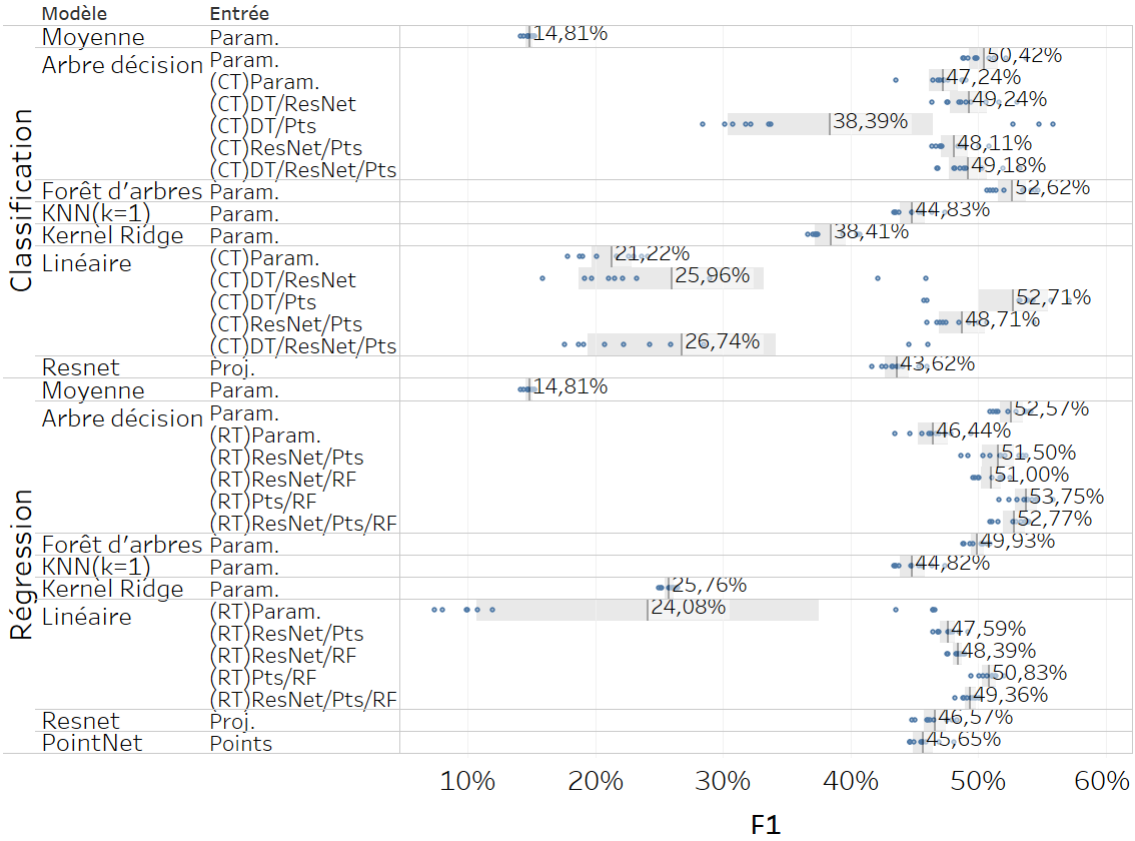

FIGURE 5.5 – Qualité de la prédiction F1 moyenne (avec des intervalles de confiance à 95%) lorsque nous utilisons l'approche par *métamodèle* utilisant l'ensemble d'entraînement au complet pour entraîner le modèle au niveau L_1 ; les résultats présentent la moyenne de 10 réplications.

seuls en régression (0,5257).

Dans le cas de la classification, il est possible de dire que l'approche n'améliore pas la prédiction. La combinaison des arbres de décision et des points avec un *métamodèle* linéaire reste l'approche la plus performante.

Dans la figure 5.6 nous pouvons observer l'effet de la qualité F1 moyenne lorsque nous permettons de passer de la classification à la régression et vice-versa. La figure contient aussi les valeurs lorsqu'il n'y a pas d'échange pour faciliter la comparaison. Pour faire la différence entre les approches provenant de la régression ou de la classification, il faut se référer à la première lettre du préfixe.

Nous remarquons qu'il n'y a généralement pas de gain à passer de classification à la régression ou l'inverse. Cependant, nous pouvons observer une exception dans les résultats. Effectivement, si nous prenons les résultats en régression pour le modèle L_0 et que nous utilisons un classificateur linéaire au niveau L_1, nous remarquons un gain de performance impression-

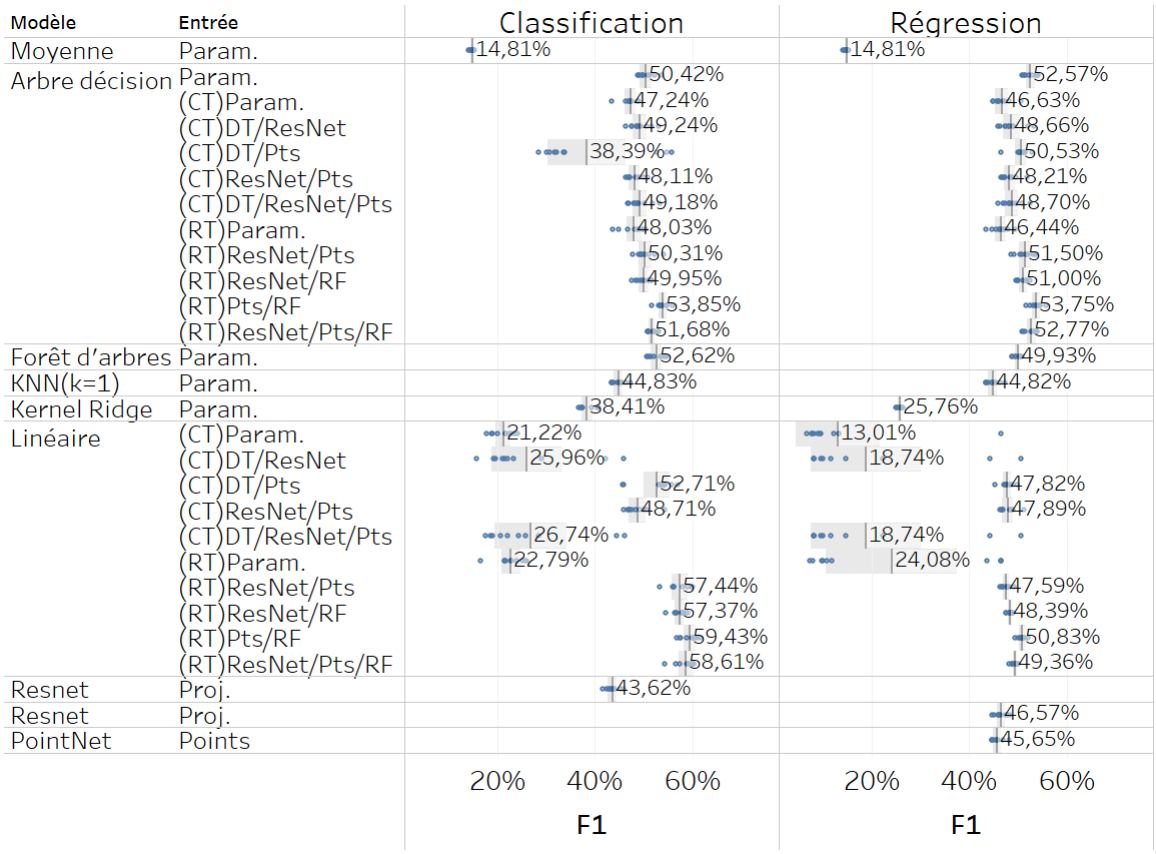

FIGURE 5.6 – Qualité de la prédiction F1 moyenne lorsque nous utilisons l'approche par méta modèle utilisant l'ensemble de d'entraînement étendu et que nous permettons d'échanger les types de prédictions différents au niveau L_0 et L_1 ; les résultats présentent la moyenne de 10 réplications.

nant. Cependant, les résultats combinant les données paramétriques ne montrent pas le même gain. Pour les autres combinaisons utilisant cette approche, nous observons des résultats entre 0,5744 et 0,5943. Ce qui dépasse les meilleurs résultats que nous avons obtenus avec les méthodes traditionnelles. Un autre détail est que dans ce cas, il semble que ce soit combiner les approches avec les classificateurs linéaires qui sont le plus performants. Ce résultat est intriguant puisque de manière générale, les arbres de décisions semblent plus performants que le classificateur linéaire au niveau L_1.

Tel que nous avons mentionné, il y a une exception à ce que nous venons de voir avec la combinaison qui semble plus prometteuse. Effectivement, si nous regardons le cas où nous utilisons uniquement les données paramétriques en entrée. Il y a une perte de performance importante. Dans ce cas, nous avons une valeur de 0,2279 pour la métrique F1 moyenne alors que le pire modèle individuel à une valeur de 0,4483 (k-NN) pour la métrique F1 moyenne. Ce résultat permet de nous demander si cette approche est vraiment une percée qui généralise bien pour tous les jeux de données ou si elle est seulement efficace avec notre jeu de données.

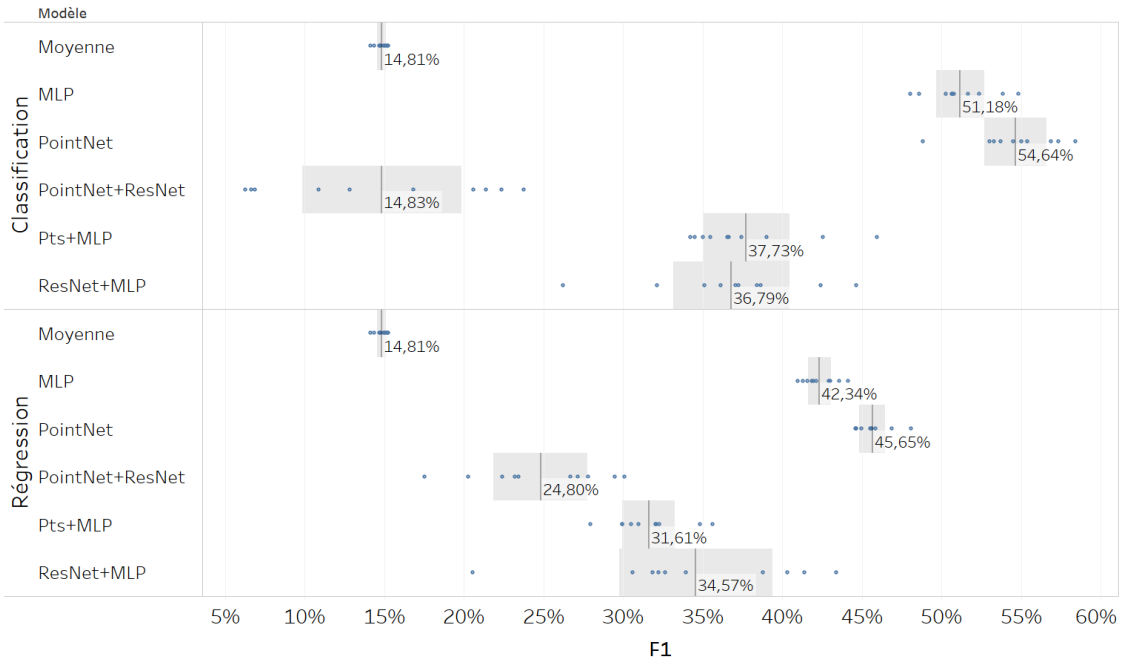

FIGURE 5.7 – Analyse de la qualité de la prédiction F1 moyenne (avec des intervalles de confiance à 95%) pour les modèles bout en bout qui combine les différentes représentations des billots en utilisant la concaténation ; les résultats présentent la moyenne de 10 réplications.

5.3.5 Performance de l'approche bout en bout

Le but de cette section est de valider si un modèle plus complexe est capable de faire l'apprentissage bout en bout pour la prédiction de panier de produits. Nous allons premièrement explorer l'approche utilisant la concaténation des modèles à ce qui correspond à la sortie du niveau L_0 pour les autres approches présentées précédemment. Par la suite, nous explorons l'approche additionnant le signal venant des diverses représentations.

La figure 5.7 présente une série de résultats obtenus avec l'approche bout en bout utilisant la concaténation pour combiner les différentes représentations. Avec cette approche, nous constatons que toutes les combinaisons avec les modèles de bout en bout sont inférieures aux prédictions individuelles. Comme il est possible de voir, la meilleure approche fournit une valeur F1 moyenne de 0,3773 pour la combinaison des données paramétriques et les nuages de points en classification. Cependant, lorsque ces approches sont entraînées individuellement, elles ont respectivement des valeurs de 0,5118 et 0,5464 pour la métrique F1 moyenne.

Un autre point intéressant de cette expérimentation que la combinaison de *PointNet* et *Resnet* obtient une valeur de 0,1483 pour la métrique F1 moyenne alors que la prédiction moyenne obtient une valeur de 0,1481 pour la métrique F1 moyenne.

La figure 5.8 présente les résultats obtenus avec l'approche bout en bout additionnant le signal sortant des blocs correspondant au niveau L_0 dans les approches précédentes. L'observation

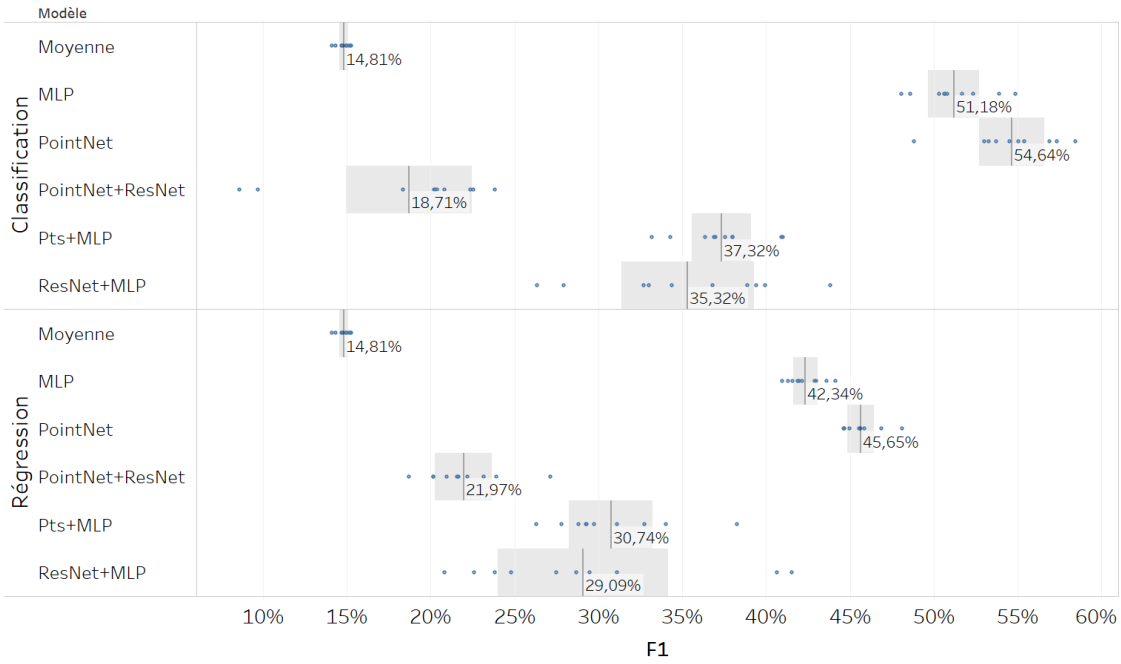

FIGURE 5.8 – Analyse de la qualité de la prédiction F1 moyenne (avec des intervalles de confiance à 95%) pour les modèles bout en bout qui combine les différentes représentations des billots en utilisant l'addition des signaux ; les résultats présentent la moyenne de 10 réplications.

qui ressort de cette approche est qu'encore une fois, tous les modèles bout en bout font moins bien que les modèles individuels. Il semble que la combinaison des nuages de points et des projections soit la pire combinaison tant en régression qu'en classification. Cependant, aucune des approches que nous venons de proposer ne se rapproche des prédictions individuelles. Certaines études ont d'ailleurs montré qu'il était possible de rencontrer une certaine limite avec les approches de bout en bout. Effectivement, faire un réseau de neurones plus complexe ne semble pas toujours être la solution (Glasmachers, 2017). Dans le contexte actuel, nous ne pouvons pas conclure que l'approche ne peut pas fonctionner, mais il est possible qu'il manque une information à extraire pour bien guider l'apprentissage.

En guise de piste de solution pour une approche de bout en bout fonctionnelle, nous nous permettons de mentionner qu'utiliser l'addition pour combiner les réseaux rend plus complexe pour l'algorithme d'optimisation de comprendre l'importance de chaque élément de la représentation du billot.

Un dernier point à mentionner est qu'il y avait une hypothèse qu'il serait plus efficace d'entraîner un seul gros modèle plutôt que plusieurs petits. Cependant, nos expérimentations nous montrent que c'est plutôt le contraire qui se produit. Par exemple, les réseaux de type *PointNet* sont relativement lents pour faire une *epoch*, mais demandent peu d'*epochs*. Les projections sont rapides pour itérer sur une *epoch*, mais demandent un grand nombre d'*epochs* pour conver-

ger. En combinant les deux modèles dans un seul modèle, nous avons remarqué que ce modèle demandait un grand nombre d'*epochs* pour converger et que le temps d'entraînement était plus long par *epoch*. Ce qui se trouve à être le pire scénario envisageable du point de vue de l'entraînement.

5.4 Conclusion

En conclusion, il est possible d'améliorer la qualité de prédiction de nos algorithmes en combinant certains modèles. En effet, nous avons trouvé une approche qui offre des résultats largement supérieurs aux autres (soit la variation de l'approche utilisant la prédiction par *métamodèle* qui passe de la régression vers la classification). Cependant, cette approche n'est pas documentée dans la littérature et il est possible que ce soit une approche qui ne généralise pas bien aux autres problèmes. Nous avons aussi trouvé d'autres approches où la combinaison de modèles offrait un gain de performance. Par contre, ces cas sont marginaux, il serait prudent de valider ces résultats au cas par cas plutôt que de conclure qu'il s'agit là de bonnes approches.

Un autre point important est que toutes ces expérimentations ont apporté un nombre non négligeable d'évidences empiriques liées à notre problème de prévision de la sortie d'une usine de sciage. Par exemple, certains modèles semblent avoir plus de difficulté à prédire des valeurs près des valeurs entières en régression. Il est possible de croire que nous pouvons obtenir des gains en régression à ce niveau. Cette exploration sera d'ailleurs conduite dans le prochain chapitre.

Il est aussi important de mentionner que les approches que nous avons utilisées font usage de l'augmentation de données. De plus, entre les nuages de points et les données paramétriques, il y a une certaine transformation d'appliquée. Il est possible que l'augmentation de données ne soit pas une bonne approche pour ce problème puisqu'elle augmente significativement la complexité en demandant aux modèles de faire la correspondance entre les deux entrées. Dans des travaux futurs, il pourrait s'avérer intéressant de refaire ces expérimentations sans l'augmentation de données pour voir s'il y a un gain de performance.

Finalement, avec la combinaison des modèles, nous constatons qu'il reste peu de données pour faire l'apprentissage des modèles au niveau L_1 et que ces modèles ont une plus grande complexité que nous pensions à l'origine. Nous avons testé une approche où il est possible d'obtenir plus de données pour l'entraînement des modèles au niveau L_1 et nous avons remarqué des gains importants. Dans ce contexte, il est possible que l'approche soit fonctionnelle, mais que nous ayons besoin de plus de données pour bien la valider.

Chapitre 6

Propositions pour l'amélioration des réseaux de neurones et des prédictions en régression pour la prédiction du panier de produits pour un billot

Le chapitre 4 montre que les résultats sont semblables pour l'ensemble des approches et des représentations du billot lorsque nous comparons les prédictions par billot. Ce chapitre explique les différentes approches que nous avons explorées pour améliorer la qualité de la prédiction. Ces explorations sont basées sur deux hypothèses. Dans un premier temps, nous savons que les données paramétriques ne sont pas en mesure de décrire adéquatement tous les billots dans notre jeu de données. Par exemple, la mesure de courbure n'est pas efficace pour décrire les billots avec une forme en S. Notre première hypothèse est donc que les projections et les nuages de points devraient surpasser les données paramétriques. La seconde hypothèse est que nous n'indiquons pas correctement aux modèles ce qu'ils doivent apprendre dans le sens où les fonctions de pertes sont inappropriées. Avec les réseaux de neurones, il est facile d'explorer différentes fonctions de pertes et nous avons développé l'intuition qu'il était possible d'en trouver une plus performante pour notre problème que les fonctions de pertes traditionnelles.

Dans ce chapitre, nous expliquons comment, dans le cadre de notre problème, il est possible d'améliorer la régression en arrondissant la sortie (section 6.1). Par la suite, nous introduisons trois fonctions de perte développées pour nos réseaux de neurones en régression (section 6.2). Les résultats de ce chapitre ont démontré qu'il était possible d'obtenir de meilleures prédictions en utilisant les réseaux en arrondissant la sortie du réseau de neurones et en utilisant une fonction de perte plus appropriée pour notre problématique (section 6.3). Avec ces changements, les réseaux de neurones sont maintenant notre approche la plus performante.

6.1 Réduction de l'erreur basée sur les valeurs entières

Comme mentionné dans le chapitre 2, la régression n'est pas contrainte à fournir des valeurs entières pour la prédiction des paniers de produits. Au fil des expérimentations, nous avons remarqué que les approches ont généralement une petite erreur sur la prédiction du compte de produits pour les quantités à prédire qui sont nulles. Au lieu de prédire une valeur de 0, les algorithmes prédisent une valeur près de 0. Or, ces erreurs pourraient avoir tendance à se cumuler et à dégrader les performances.

Nous avons exploré deux approches pour essayer de diminuer les erreurs sur les prédictions. Dans un premier temps, nous utilisons la partie entière inférieure (le plancher ou *floor* en anglais) pour chacun des éléments du vecteur de comptes de produits prédit. La seconde approche est d'arrondir à l'entier le plus proche les valeurs dans le vecteur de comptes prédit. Dû à la configuration de nos arbres de décisions qui ne possèdent pas de limites de profondeur ou de nombre de branches, il est attendu que ceux-ci, de même que l'approche k-NN lorsque $k = 1$, prédiront des nombres entiers en régression de sorte que les deux techniques mentionnées ci-haut ne devraient pas influencer leurs prédictions.

6.1.1 Exploration des résultats sur la transformation de la sortie en compte de produits entiers

La figure 6.1 démontre l'effet de convertir les prédictions en nombres entiers pour les produits de sciage lorsque ceux-ci viennent d'une prédiction faite en utilisant la régression. La figure compare les résultats sans modification (première colonne) aux deux autres approches que nous proposons.

Premièrement, nous observons que l'algorithme des arbres de décisions fournit une prédiction en nombres entiers lorsque la régression est utilisée. Ce résultat attendu pourrait expliquer pourquoi les arbres de décisions faisaient aussi bonne figure initialement. Nous avons le même comportement pour l'algorithme k-NN avec une valeur de $k = 1$ qui prédit aussi des valeurs entières. Malheureusement, cet algorithme n'offre pas d'aussi bons résultats que les arbres de décisions. Une seconde observation est qu'en utilisant le plancher la performance de la prédiction diminue significativement. Nous remarquons une diminution qui peut dépasser 50% si nous regardons k-NN avec une valeur de $k = 40$.

En revanche, lorsque nous utilisons l'arrondi pour transformer les paniers de produits, nous observons une augmentation de la mesure F1 moyenne pour tous les algorithmes qui ne prédisent pas des nombres entiers. À l'origine, le modèle qui performe le mieux est l'arbre de décision avec une valeur F1 moyenne de 0,5257. Avant d'appliquer l'arrondi, les forêts d'arbres décisionnels ont une valeur de 0,4993 pour la métrique F1 moyenne. Si nous appliquons l'arrondi, les forêts d'arbres décisionnels obtiennent une valeur de 0,5577 pour la métrique F1 moyenne. Ce qui représente une augmentation de 0,0320 par rapport aux arbres de décisions.

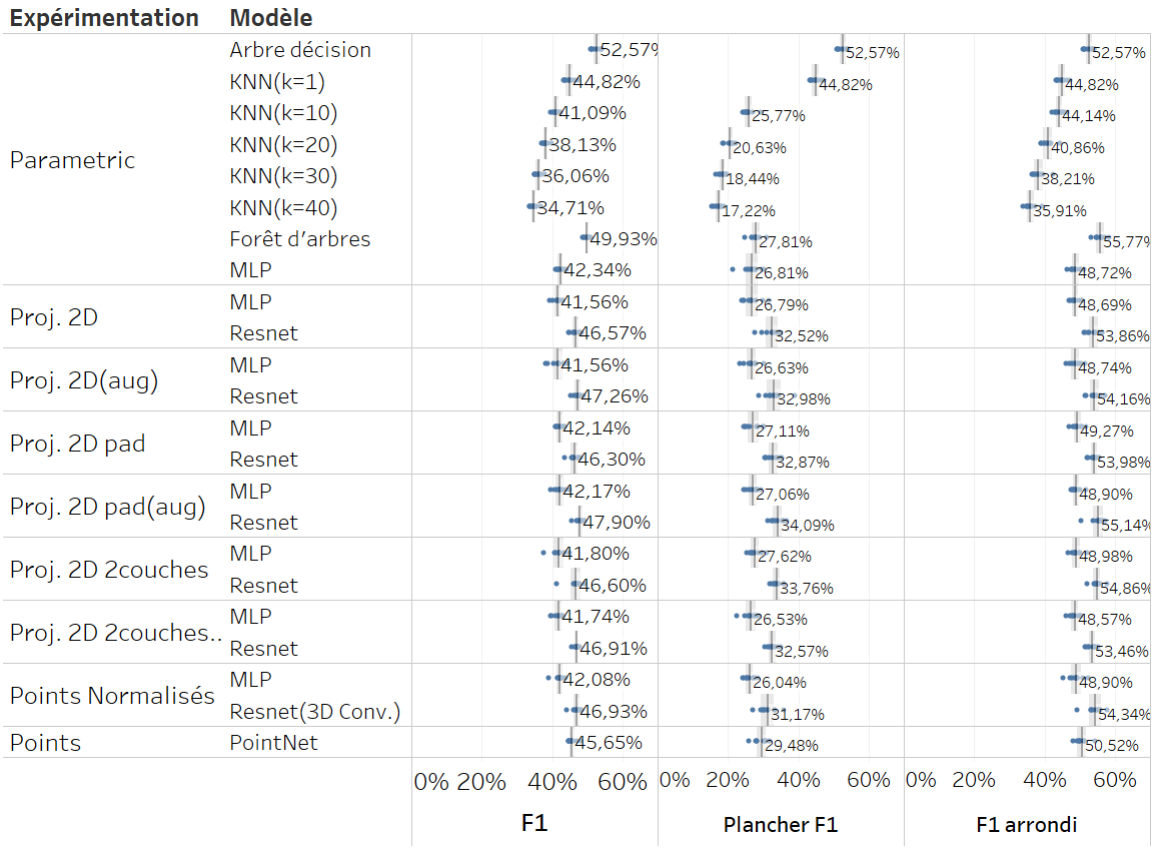

FIGURE 6.1 – Effet de transformer les prédictions obtenues par régression en panier de produits contenant des nombres entiers (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

En utilisant l'arrondi, tous les algorithmes utilisant une variante de *ResNet* offrent maintenant une performance supérieure aux arbres de décisions.

6.2 Exploration de fonctions de perte alternatives

Lors de la progression des travaux, nous avons émis l'hypothèse qu'il était plausible qu'il existe des fonctions de perte plus performantes que les fonctions de perte communes en contexte de classification ou de régression. Nous avons donc exploré une série de fonctions de perte dans le but d'améliorer la classification ou la régression.

Les options que nous présentons dans les sections suivantes sont :

— optimiser la valeur finale du panier de produits (section 6.2.1) ;

— une approche qui réplique la métrique F1 (section 6.2.2) ;

— une fonction de perte qui combine l'erreur quadratique moyenne à notre fonction de perte répliquant la métrique F1 (section 6.2.3) ;

Finalement, nous expliquons brièvement quelques approches que nous avons explorées, mais qui ne se sont pas avérées efficaces (section 6.2.4).

6.2.1 Réduction de l'erreur basée sur la valeur

La différence de valeur est une métrique ayant un sens en industrie. De plus, toutes les erreurs ne sont pas équivalentes pour tous les produits lorsqu'on considère la valeur. En effet, l'impact de ne pas prédire un produit de grande valeur est plus grand que l'impact de ne pas prédire un produit de faible valeur.

Nous avons implémenté cette idée en nous basant sur la valeur des produits de sciage qui sont définis dans la configuration de la scierie utilisée par le simulateur de débitage. Il est donc possible de créer la fonction de perte suivante pour minimiser l'erreur sur la valeur du billot :

$$MSE^{val}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i \cdot v_i - ReLU(\hat{y_i}) \cdot v_i)^2 \ , \tag{6.1}$$

où v est un vecteur de valeurs de produit, y les vecteurs de comptes représentant le panier de produits obtenus par le simulateur de débitage et $\hat{y}$ le panier de produits prédit par les divers algorithmes.

Nous utilisons la fonction $ReLU$ en tant que fonction d'activation à la sortie du réseau de neurones. Avec cette fonction d'activation, le réseau de neurones ne peut plus prédire des valeurs négatives (Nair et Hinton, 2010). Nous avons jugé important d'ignorer les quantités négatives en sortie de réseau parce que dans de tels cas on réduit la valeur finale du billot ce qui en pratique est illogique.

Résultats

La figure 6.2 montre les résultats qui comparent les résultats précédents à la nouvelle approche. Cette figure montre les résultats des approches précédentes dans les colonnes *CrossEntropy* qui est la fonction de perte utilisée en classification et la colonne *MSE* qui est utilisée en régression et qui représente l'erreur quadratique moyenne (en anglais *mean squared error*). Les résultats de la nouvelle fonction de perte se trouvent dans la colonne *ValueLoss*.

Comme nous pouvons le constater, la nouvelle approche performe moins bien pour toutes les approches antérieures. Nous pouvons observer une particularité surprenante. Les modèles utilisant les perceptrons multicouches ont une dégradation moins importante que les autres architectures. En effet, les architectures de types *ResNet* causent une dégradation des performances lorsque nous les utilisons. Cependant, les approches *ResNet* sont plus performantes malgré la plus grande diminution de la performance.

De plus, nous remarquons qu'avec cette approche les modèles ont une petite tendance à se tromper dans la prédiction des grades. Lorsque nous explorons la valeur des produits de sciage,

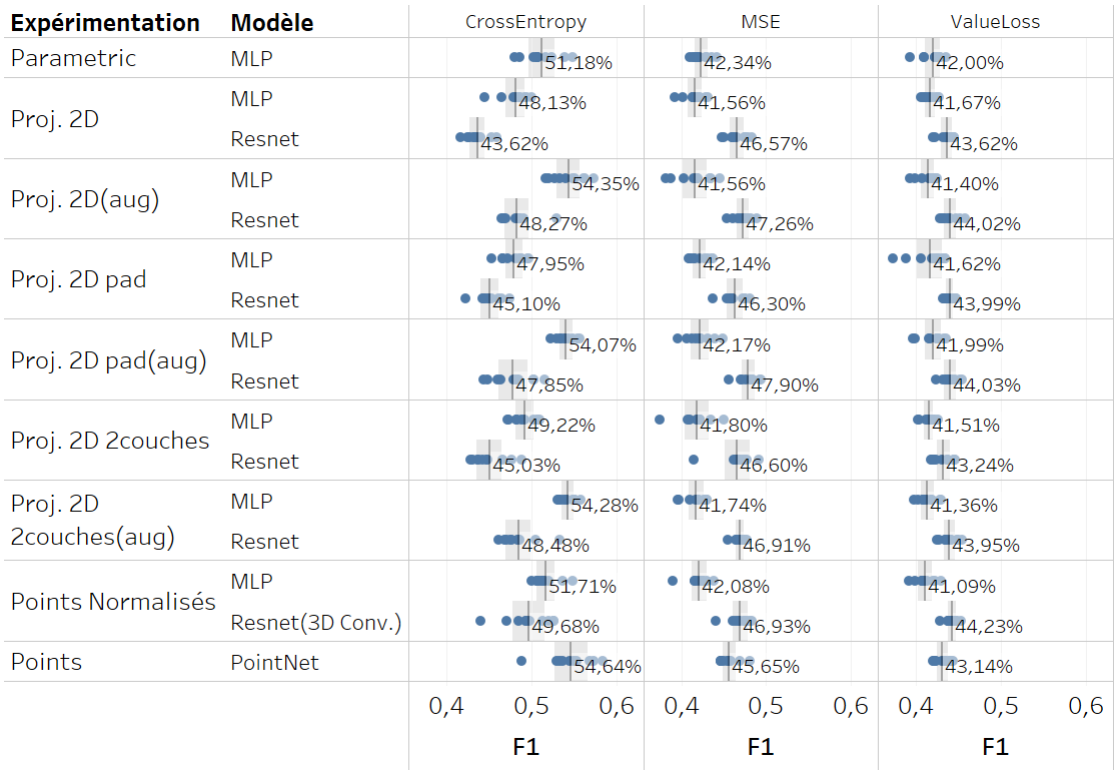

FIGURE 6.2 – Résultats comparant la fonction de perte utilisant l'erreur quadratique moyenne de valeur des paniers de produits et l'erreur quadratique moyenne sur les comptes de produits (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

nous observons que pour beaucoup de produits similaires, la valeur marchande qui nous a été fournie est la même. Par exemple, les produits de grade supérieur et grade 1 ont toujours la même valeur. Le simulateur de débitage semble en mesure de faire la distinction entre les deux types de produits puisqu'il est possible d'observer les deux grades dans le jeu de données. Dans ce contexte, avec notre fonction de perte, si nous échangions un produit de grade supérieur et un produit de grade 1, nous aurions une erreur de 0. Cependant, notre métrique F1 moyenne considérera cela comme une erreur pouvant être importante.

Après expérimentations, nous concluons que cette fonction de perte n'est pas pertinente pour la métrique F1 moyenne que nous utilisons pour calculer la performance de nos modèles. Comme il est important de bien prédire le grade dans nos expérimentations, nous avons choisi de ne pas pousser les expérimentations plus loin sur la fonction de perte utilisant la valeur des billots.

6.2.2 Réduction de l'erreur basée sur la métrique F1

Lors de nos prédictions en mode régression, nous prédisons une quantité pour chacun des produits possibles pour la scierie. Dans notre cas, nous avons 85 produits de sciage différents.

Selon nos analyses du jeu de données, les billots ont jusqu'à 10 produits. Cependant, nous comptons 7 produits différents au maximum. Intuitivement, ceci nous indique qu'une majorité de comptes devrait être nuls dans le vecteur de compte représentant le panier de produits. En régression la fonction de perte est l'erreur quadratique moyenne décrite de la manière suivante :

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \ . \tag{6.2}$$

Dans notre contexte, la différence entre la prédiction ($\hat{y_i}$) et la valeur réelle (y_i) devrait être nulle ou contenir une très petite valeur pour la majorité des indices dans le vecteur de comptes représentant les produits de sciage. Faire une moyenne dont la majorité des éléments sont nuls résulte en une petite valeur lorsque le nombre est aussi grand que notre nombre de produits de sciage. Dans ce contexte, nous suspectons que l'erreur en sortie est trop petite pour que les réseaux de neurones soient en mesure de bien apprendre. Pour résoudre ce problème, nous devions obtenir une formule qui n'utilise que l'erreur concernant les éléments « intéressants » dans notre prédiction. Or, la métrique F1 permet exactement de montrer cette importance. Par conséquent, nous avons donc choisi d'adapter cette métrique pour en créer une fonction de perte que nous pouvons utiliser dans un réseau de neurones, c'est-à-dire une fonction qu'il est possible de dériver de bout en bout.

Afin de créer cette fonction de perte, nous modifions le calcul de la métrique F1 ainsi :

$$TP(y, \hat{y}) = ReLU(\hat{y_i}) - ReLU(\hat{y_i} - y_i) \,, \tag{6.3}$$

$$TN(y, \hat{y}) = ReLU(\hat{y_i}) - TP(y, \hat{y_i}) \,, \tag{6.4}$$

$$FP(y, \hat{y}) = y_i - TP(y, \hat{y_i}) \,, \tag{6.5}$$

$$F1(y, \hat{y}) = \frac{2\,TP(y, \hat{y_i})}{2\,TP(y, \hat{y_i}) + FP(y, \hat{y_i}) + FN(y, \hat{y_i})} \,. \tag{6.6}$$

Le but de cette modification est de transformer la métrique F1 de manière à ce qu'elle soit utilisable par un réseau de neurones. Une fois que nous avons calculé la valeur F1 pour nos prédictions par rapport au résultat attendu, il nous reste à inverser la valeur. La raison est que la fonction de perte tente de minimiser l'erreur et cette erreur correspond à l'inverse de la métrique F1. La fonction de perte devient alors :

$$F1Loss(y, \hat{y}) = 1.0 - F1(y, \hat{y}) \,. \tag{6.7}$$

Résultats avec utilisation de la fonction de perte similaire à la métrique F1

La figure 6.3 montre les résultats de la fonction de perte basée sur la métrique F1. Le format de la figure est tel que nous comparons notre nouvelle fonction de perte par rapport à la classification et à la régression que nous avons présentée dans le chapitre 3. Notre nouvelle approche

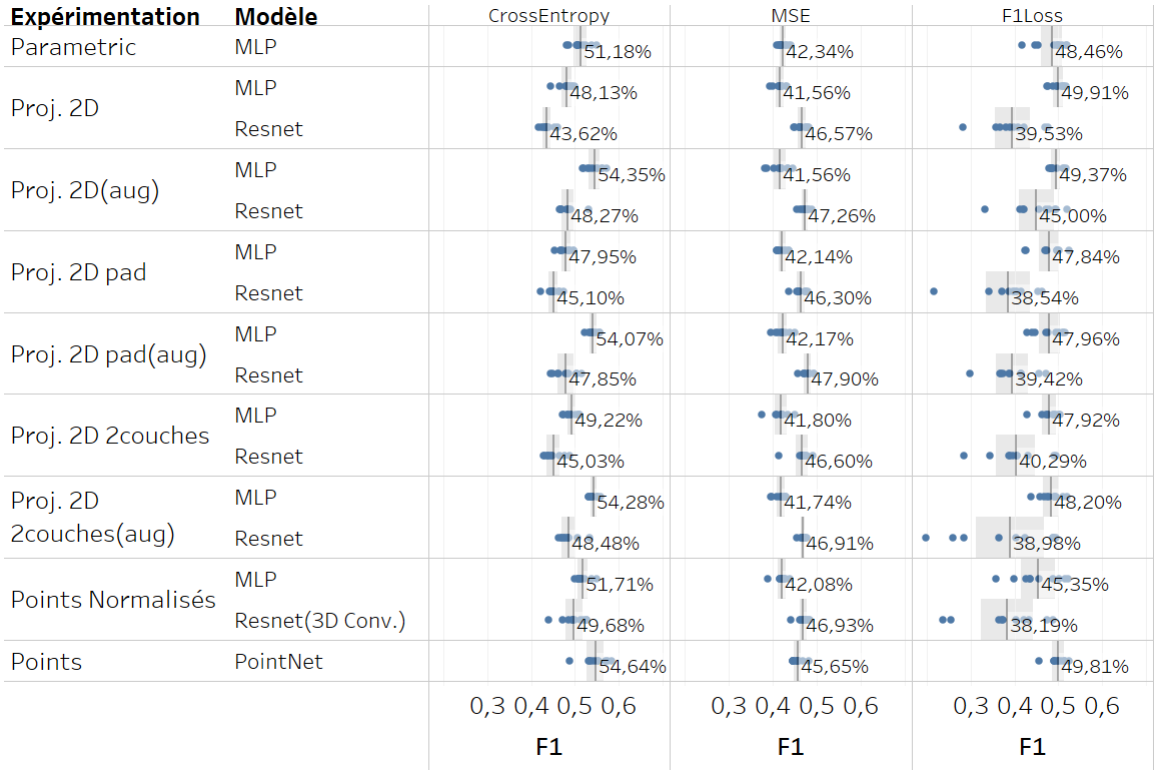

FIGURE 6.3 – Résultats comparant le score F1 moyen lorsque la fonction de perte est l'erreur quadratique moyenne basée sur le compte de produits et lorsqu'elle est basée sur le *F1Loss* (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

est présentée dans la colonne *F1Loss*. Les perceptrons multicouches sont significativement supérieurs à l'approche classique utilisée en régression. Par exemple, nous remarquons un gain de 19,8% pour les projections 2D avec augmentation de données. Cette augmentation de la qualité de la prédiction s'observe pour tous les modèles utilisant les perceptrons multicouches et ce peu importe la représentation du billot que nous utilisons. Nous pouvons aussi observer une amélioration pour les réseaux de type *PointNet*. Avec l'utilisation de *PointNet*, nous remarquons une amélioration du score F1 moyenne de 9,1%. Un dernier constat concernant la régression est que la nouvelle fonction de perte fait moins bien que l'erreur quadratique moyenne pour les modèles de type *ResNet*. Nous remarquons une baisse d'environ 15% pour ces approches. Nous pouvons aussi observer ce phénomène pour notre variation du modèle *ResNet* que nous avons utilisé pour les points normalisés.

Si nous comparons à la classification, nous remarquons que pour la projection 2D utilisant le perceptron multicouche, il est possible d'avoir une approche de régression qui fait mieux que les approches par classification. Cependant, toutes les autres expérimentations performent de manière similaire ou moins bien que la classification en utilisant l'entropie croisée.

Une explication plausible des résultats obtenus avec la nouvelle fonction de perte se base sur l'observation que le réseau de neurones a tendance à prédire de grandes valeurs négatives pour certains produits de sciage. Ce comportement pourrait être causé par l'utilisation de la fonction $ReLU$ qui est insensible aux grandes valeurs négatives. Lorsque ce phénomène se produit, le réseau semble incapable de bien prédire le produit de sciage correspondant par la suite.

Bien que la fonction de perte montre des gains par rapport à l'erreur quadratique moyenne utilisée pour la régression, les réseaux de neurones ne performent pas encore à la hauteur des arbres de décisions en régression (0,5257).

6.2.3 Réduction de l'erreur basée sur la métrique F1 et l'erreur quadratique moyenne

La fonction de perte présentée dans cette section est une amélioration de la fonction perte similaire à la fonction basée sur la métrique F1 (section 6.2.2). Considérant que le problème de cette fonction de perte est qu'elle laisse le réseau produire de grandes valeurs négatives, nous croyons qu'il est possible de contraindre le réseau à produire des valeurs qui sont plus proches de 0. Par exemple, la fonction de perte utilisant l'erreur quadratique moyenne sur le compte de produits a une tendance à prédire des valeurs plus proches de 0. D'ailleurs, comme nous l'avons observé précédemment, elle a aussi une tendance à sous-prédire des produits de sciage. L'idée est de combiner les deux approches en utilisant un paramètre constant α qui permettra de quantifier l'importance des deux termes de la fonction de perte, celui correspondant à l'erreur quadratique moyenne effectuée sur tout le panier de produits et celui correspondant à la fonction *F1Loss*. Ainsi, nous avons :

$$F1MSELoss(y, \hat{y}) = \alpha \cdot (F1Loss(y, \hat{y})) + (1 - \alpha) \cdot (MSELoss(y, \hat{y})) \ . \tag{6.8}$$

La raison pour laquelle nous utilisons un paramètre α est parce que les deux fonctions de pertes ont des échelles et des intervalles de valeurs différents. Par exemple, la section *F1Loss* est nécessairement inférieure à 1. En ce qui concerne la perte sur l'erreur quadratique moyenne, elle peut avoir des valeurs beaucoup plus grandes que 1. Il est donc possible qu'on ait besoin de balancer l'importance des deux fonctions de pertes que nous utilisons. Nous explorons aussi l'effet que ce paramètre a sur notre nouvelle fonction de perte.

Résultats de l'utilisation de la fonction de perte basée sur la métrique F1 et l'erreur quadratique moyenne

Nous commençons par évaluer la qualité de la métrique $F1$ pour notre nouvelle fonction de perte en la comparant aux résultats initiaux. Par la suite, nous explorons différentes valeurs pour la variable α.

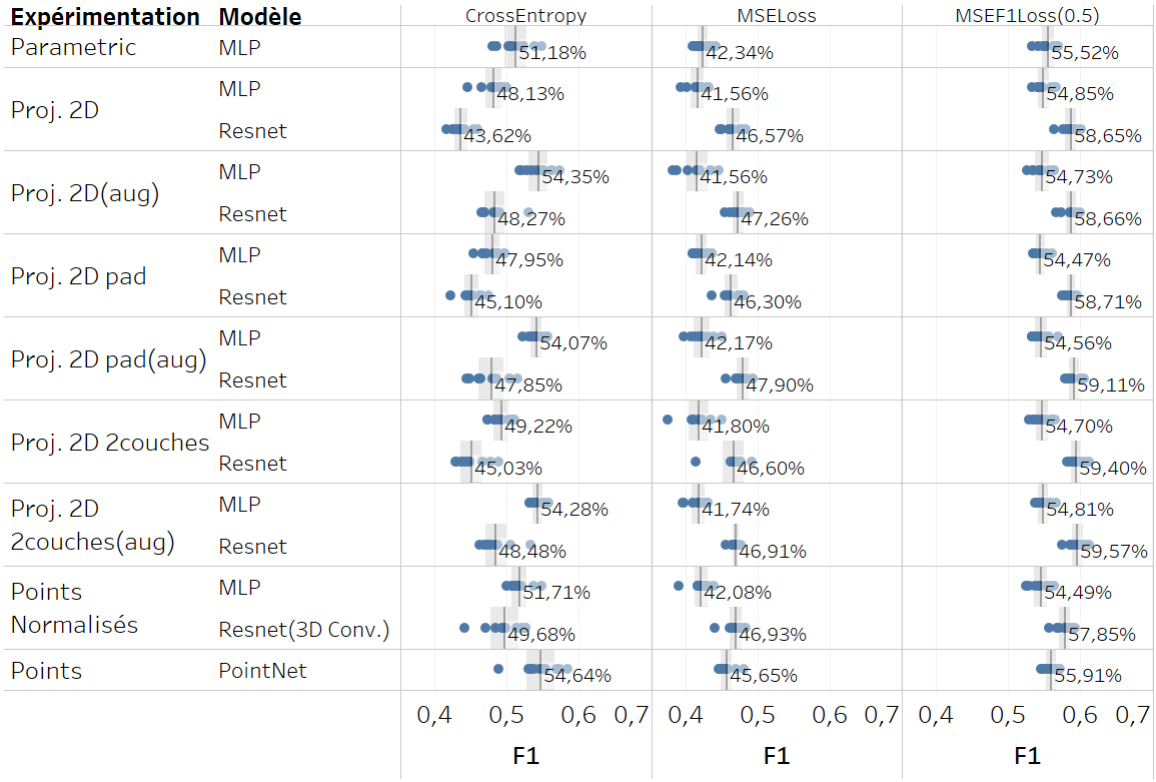

FIGURE 6.4 – F1 moyenne (%) pour l'utilisation de la fonction de perte qui combine *F1Loss* et l'erreur quadratique moyenne (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

La figure 6.4 permet de visualiser les valeurs pour la métrique F1 utilisant la nouvelle fonction de perte. Les résultats sont présentés dans la colonne *MSEF1Loss*(0.5). Comme premier test, nous avons choisi de fixer la valeur de $\alpha = 0.5$.

Dans tous les cas, les résultats moyens sont supérieurs aux approches précédentes et il est possible de conclure que la régression offre de meilleurs résultats que la classification pour sur l'ensemble. Cependant, nous remarquons que ce sont maintenant les projections 2D qui offrent les meilleurs résultats contrairement aux nuages de points et la combinaison des nuages de points avec *PointNet*.

Un second résultat que nous présentons est l'effet du paramètre α. La figure 6.2.3 présente divers niveaux pour le paramètre α dans la fonction de perte. Pour des raisons de ressources nécessaires, nous présentons seulement trois valeurs de $\alpha \in \{0.25, 0.5, 0.75\}$. Dans cette figure, nous observons que de manière générale, la qualité de la prédiction F1 moyenne augmente lorsque nous augmentons la valeur du paramètre α. Le réseau bénéficie de mettre l'emphase sur la fonction de perte qui se rapproche de la métrique *F1Loss*. Cependant, il est évident que l'ajout de l'erreur quadratique moyenne donne un gain significatif par rapport à seulement la fonction de perte *F1Loss*.

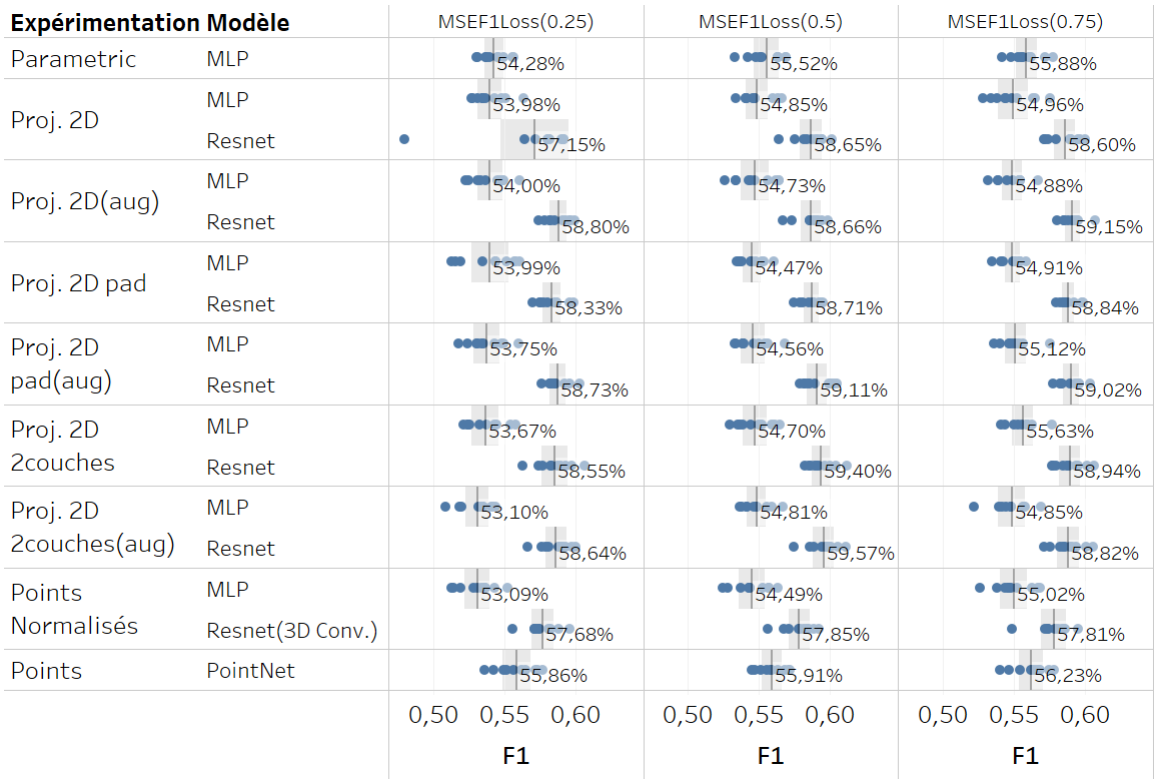

FIGURE 6.5 – Résultats comparant différentes valeurs de α lors de la combinaison de l'erreur quadratique moyenne et perte de type *F1Loss* (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

Lorsque nous utilisons *ResNet* avec les projections 2D comme représentation du billot, nous remarquons que la performance est similaire pour $\alpha = 0.5$ et $\alpha = 0.75$. En fait, il arrive parfois que $\alpha = 0.5$ soit le plus performant. Selon cette observation, il serait envisageable que la meilleure valeur pour le paramètre α se situe entre ces deux valeurs. Bien sûr, des gains substantiels pourraient peut-être encore survenir par une optimisation rigoureuse de l'hyperparamètre α (Luo, 2016). Il s'agit toutefois d'une étude que nous réservons pour des travaux futurs.

Cette approche nous a permis de réduire l'impact du problème lié au fait que le réseau de neurones ne prédit principalement des valeurs négatives de la fonction de perte précédente (*F1Loss*). De plus, nous abordons le problème de sous-prédiction associé à l'erreur quadratique moyenne.

6.2.4 Autres approches

Dans cette section, nous décrivons d'autres approches testées en cours de projet. Les résultats préliminaires ont montré que ces approches n'étaient pas efficaces. Nous présentons les approches, mais sans leurs résultats qui auraient alourdi le texte puisque rien ne portait à

croire qu'elles dépasseraient l'état de l'art. Cette section documente, en quelque sorte, une série de résultats négatifs. Néanmoins, la documentation de ces approches ajoute à l'ensemble d'évidences empiriques sur l'application des réseaux de neurones au problème de prévision de la sortie d'une usine de sciage.

Matrice expert

Cette approche vient de l'idée que toutes les erreurs ne sont pas équivalentes. Par exemple, il serait possible pour une scierie d'envoyer un produit d'un grade supérieur à son client sans problème. Bien qu'un tel comportement n'est pas souhaitable (la scierie et diminuerait son bénéfice), ce serait acceptable du point de vue du client. Il devient donc important de trouver une balance entre les deux partis. Dans ce qui suit, nous détaillons comment nous avons simulé de tels échanges de produits en appliquant une certaine pénalité lors des échanges.

Ayant accès à des experts avec nos partenaires, nous leur avons demandé de fournir un facteur correspondant à la gravité de l'erreur entre deux produits. Lorsque la valeur est nulle, cela veut dire qu'il n'y a aucune pénalité à échanger les deux produits. Plus la valeur est grande, plus il est grave d'échanger les produits. Considérant qu'une scierie produit n différents produits, l'expert doit fournir une matrice de $n \times n$. Une particularité de cette matrice est que la diagonale est composée de 0 puisque la diagonale représente la conversion d'un produit en lui-même.

Avant de pouvoir calculer l'erreur basée sur la conversion des produits, il est important de créer une matrice qui permet de réassigner les produits de sciage vers un produit de sciage différent lorsque nécessaire. Cette opération crée une seconde matrice de taille $(n + 1) \times (n + 1)$. La ligne et la colonne supplémentaires sont nécessaires lorsqu'il n'est pas possible de convertir un produit de sciage vers un autre produit de sciage. Pour obtenir l'erreur, il suffit de multiplier les éléments de chaque matrice. Pour la ligne et la colonne $n+1$, nous multiplions cette erreur par l'erreur maximale de la ligne ou de la colonne correspondante.

Cette expérimentation nous a permis de comprendre la complexité de construire cette matrice à partir de la connaissance d'un expert. De plus, nous comprenons que dans la réalité, nous pouvons principalement échanger des produits en fonction du grade. Lorsqu'il s'agit d'une erreur sur une des dimensions comme la largeur, la longueur ou l'épaisseur ce n'est pas possible.

En explorant cette approche, nous avons observé que la source d'erreur principale n'est pas le grade. C'est de cette expérimentation que provient l'analyse des erreurs en fonction de la dimension présentée au chapitre 2.

Matrice par valeur

Cette approche est similaire à l'approche précédente. La différence principale est que nous utilisons les connaissances de l'expert indirectement. Lorsqu'un expert choisit la liste de pro-

duits possibles pour une scierie, il assigne une valeur à chacun des produits. Le simulateur de débitage vise à maximiser cette valeur pour chaque billot. Avec cette approche, nous utilisons donc les valeurs fournies au simulateur de débitage pour construire la matrice de taille $n \times n$ où n représente le nombre de produits possibles en sortie d'usine.

Avec cette approche, nous remarquons qu'il est possible d'avoir des résultats similaires à l'approche précédente sans avoir besoin d'un expert pour remplir la matrice. Cependant, le constat est similaire quant aux résultats. Les résultats sont significativement moins bons qu'avec les approches traditionnelles et nous avons jugé qu'il n'était pas pertinent de continuer dans cette direction.

Régression par classification

Cette approche a été notre premier essai pour obtenir des nombres entiers en prédisant la quantité de chacun des produits de sciage. Cependant, dans cette approche, nous utilisons la classification pour obtenir notre panier de produits.

Pour réaliser cette approche, nous créons un réseau qui possède n sorties de x valeurs où n correspond au nombre de produits de sciage possible pour la scierie et x permet de représenter le compte de chacun des produits. Pour chacune des sorties du réseau, nous utilisons la classification pour prédire le nombre de chacun des produits de sciage. Selon notre analyse du jeu de données, nous voyons que les billots n'ont pas plus de 10 produits. Il faut noter que le nombre maximal de produits identiques est de 7. Afin de donner un peu de marge sur le nombre maximal de produits pour la prédiction de produits, nous utilisons, pour chacune des sorties, un vecteur de 10 éléments sur lequel on calcule l'entropie croisée. Dans notre cas précis n a une valeur de 85 et x a une valeur de 10. Ce qui implique que le réseau doit avoir une taille de sortie de 850 valeurs.

Ce qui ressort de cette expérience est qu'une grande proportion des sorties ne sont jamais utilisées lors de l'entraînement. Par exemple, les produits ayant un grand volume sont rarement présents plusieurs fois. Les réseaux de neurones ont peu de valeurs pour mettre à jour les poids pour ces sorties. Dans de tels cas, il serait possible de réduire la taille de x pour certains produits. Cependant, les résultats préliminaires sont largement sous les résultats que nous avions avec les méthodes traditionnelles.

Considérant que l'approche a plusieurs inconvénients et que les résultats n'étaient pas prometteurs, nous avons choisi de ne pas poursuivre dans cette direction. Parmi les inconvénients de cette approche, nous remarquons qu'il est difficile de bien couvrir tous les comptes de produits de manière adéquate. De plus, l'approche ne se généralise pas bien d'une scierie à l'autre puisqu'il faudrait ajuster la taille de chacune des sorties (x) en fonction de la capacité de chaque scierie.

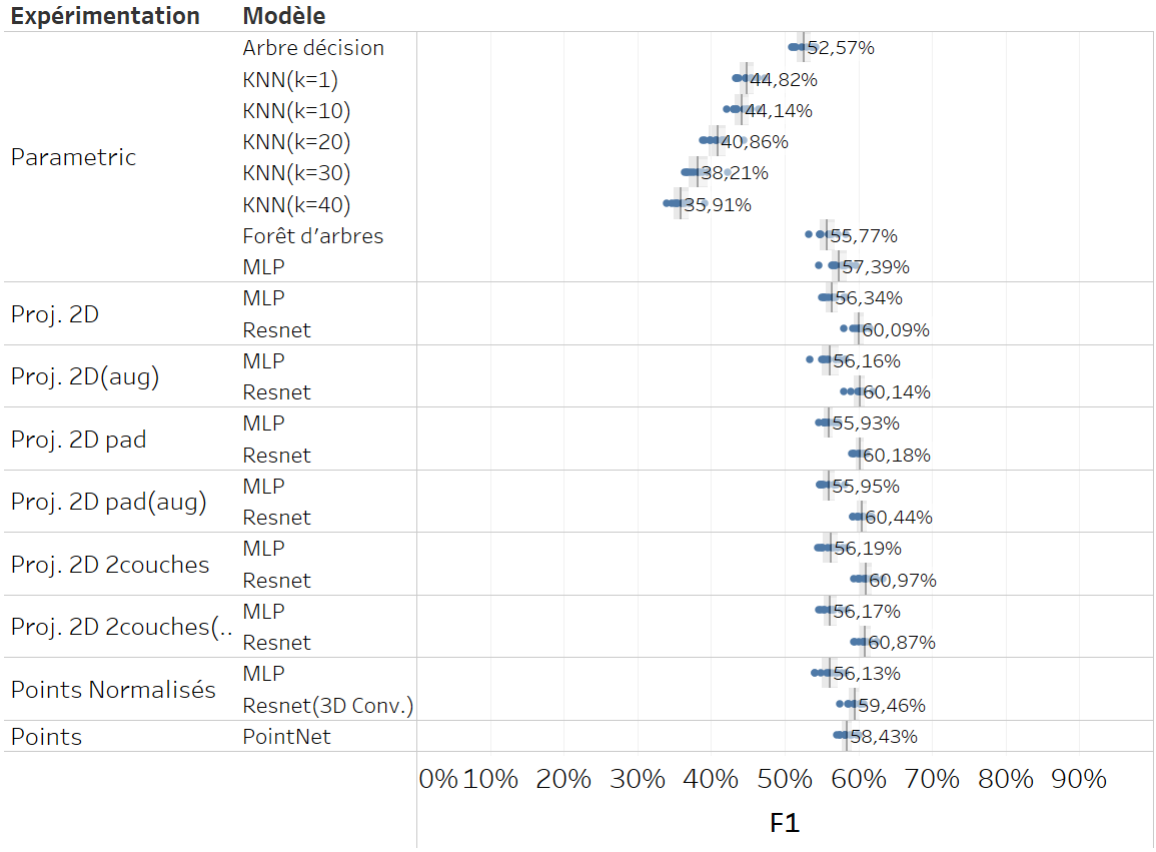

FIGURE 6.6 – Résultats démontrant la qualité de la prédiction lorsque nous combinons l'effet d'arrondir la prédiction et d'utiliser une fonction de perte qui combine un calcul similaire à la métrique F1 et l'erreur quadratique moyenne (avec des intervalles de confiance à 95%) ; les résultats présentent la moyenne de 10 réplications.

6.3 Mise à jour des résultats en utilisant les améliorations

Dans cette section, nous présentons les résultats provenant de la combinaison des diverses approches que nous avons trouvées pour améliorer les approches utilisant les réseaux de neurones et la régression. Selon nos expérimentations précédentes, nous avons remarqué qu'arrondir à l'entier le plus près la quantité de produits de sciage et utiliser une nouvelle fonction de perte améliore la qualité de la prédiction. La fonction de perte que nous retenons est la fonction qui combine un calcul similaire à la métrique F1 et l'erreur quadratique moyenne.

La figure 6.6 montre les résultats mis à jour. Comme nous pouvons le voir, la majorité des réseaux de neurones vont maintenant mieux en régression que ce que nous avions obtenu originalement que ce soit en classification ou en régression. Ce résultat est extrêmement encourageant dans la mesure où cette approche nous permet d'obtenir de bonnes prédictions sans avoir à préalablement définir tous les paniers de produits possibles pour une scierie comme il est souhaitable en classification. Effectivement, en classification, si un nouveau panier de

produits était découvert, il faudrait modifier la configuration du réseau et entraîner le réseau à nouveau pour supporter cette nouvelle sortie. Avec l'approche présentée en régression, il n'est plus nécessaire de modifier la configuration du réseau pour supporter de nouveaux paniers de produits tant qu'on ne modifie pas le nombre de produits possibles pour une scierie.

De plus, nous remarquons que maintenant notre nouvelle approche donne une amélioration de 11,5% par rapport à la meilleure approche que nous avions initialement avec *PointNet* en classification. Une autre observation que nous pouvons faire est qu'il semble y avoir un léger avantage à utiliser les projections 2D plus complexes que nous avons développé lorsque *ResNet* est utilisé. La projection 2D avec remplissage à 2 couches alternées montre une amélioration d'environ 1% par rapport aux projections de base alors que les projections avec remplissage circulaire montrent un gain de 0.5% par rapport aux projections de bases. Il est intéressant de mentionner que les perceptrons multicouches ont des performances similaires, peu importe le type de projections.

Finalement, dans les résultats, nous remarquons que les arbres de décision et l'algorithme k-NN avec $k = 1$ n'ont aucune amélioration. Nous rappelons que ces approches prédisent par défaut des nombres entiers. Donc nous ne pouvons pas bénéficier de l'arrondi. Ils ne sont pas des réseaux de neurones et nous n'avons pas pu utiliser notre fonction de pertes pour ces algorithmes.

6.4 Conclusion

Ce que nous remarquons de nos expérimentations sur les améliorations des réseaux de neurones est qu'il est possible d'améliorer la qualité de la prédiction. En fait, nous remarquons qu'arrondir la prédiction des réseaux de neurones et qu'utiliser une fonction de perte combinant l'erreur quadratique et la métrique F1 permet d'augmenter la qualité de la prédiction beaucoup plus qu'en travaillant sur la représentation du billot.

Nous remarquons aussi que pour trouver une fonction de perte pertinente à notre problème, il est important de pouvoir essayer plusieurs variations et de bien analyser les erreurs de chacune des fonctions de perte. Ce travail a montré être efficace pour notre problème et nous a aussi permis d'acquérir plusieurs connaissances qui sont nécessaires pour améliorer la qualité de la prédiction et nous permettre de progresser sur le problème de la prédiction de la transformation d'un billot en produits de sciage.

Finalement, nous proposons une approche qui semble beaucoup plus stable et constante que ce que nous avons trouvé dans le chapitre 5 où nous combinons les approches que nous avions développées. En plus d'être plus stables dans ses résultats, nous pouvons aussi dire qu'il est beaucoup plus simple d'expliquer les résultats.

Conclusion

Dans ce mémoire, nous avons confirmé qu'il est possible d'utiliser l'apprentissage automatique en remplacement des simulateurs de débitage dans le but d'effectuer la prédiction du panier de produits pour un billot. De plus, il a été montré qu'il était possible d'utiliser les réseaux de neurones pour tirer avantage de représentations plus complètes du billot que les données paramétriques communément utilisées et pour leur facilité à être adapté à notre problématique.

Premièrement, comme nos expérimentations portaient sur un nouveau jeu de données, nous avons choisi d'utiliser les mêmes approches d'apprentissage supervisé présentées par Morin, Gaudreault et al. (2020) qui représentait l'état de l'art. Nous avons refait tout le développement nécessaire pour utiliser ces algorithmes dans le but d'avoir un point de références pour comparer les résultats des nouvelles approches que nous avons développées. En plus de pouvoir reproduire les résultats sur un nouveau jeu de données, nous avons découvert que les métriques utilisées précédemment étaient un peu généreuses dans certains contextes. Nous avons donc développé une nouvelle métrique basée sur la métrique F1 plus appropriée au problème. De plus, nous avons montré qu'il est possible d'utiliser l'apprentissage automatique pour améliorer la prédiction sur les lots par rapport à la production moyenne d'une scierie. Ces travaux ont fait l'objet du chapitre 2 et ont été publiés à CIGI-Qualita21 (Martineau, Gaudreault et al., 2021).

Par la suite, dans le chapitre 3, nous avons exploré les approches utilisant les réseaux de neurones. Avec cette nouvelle approche, il est maintenant possible d'utiliser plusieurs représentations d'un billot. Nous avons développé des approches basées sur les nuages de points qui sont normalement donnés en entrée aux simulateurs de débitage. Nous avons aussi développé une approche qui permet de transformer le nuage de points en une projection 2D. Le but de cette représentation était de maintenir le même niveau d'information tout en réduisant le nombre de données à fournir au réseau de neurones. De cette manière, nous avions accès à de nouveaux algorithmes et il était possible de diminuer les ressources nécessaires pour obtenir une prédiction. Bien entendu, nous avons aussi cherché une approche de réseau de neurones pouvant exploiter les données paramétriques. Ces travaux ont mené à la parution d'un article présenté à la conférence Canadian AI 2021 (Martineau, Morin et al., 2021). Cependant, bien que l'approche soit intéressante, nos premiers résultats ont montré que les réseaux de neu-

rones sont performants, mais ne produisent pas un gain important par rapport aux approches présentées dans le chapitre 2.

Suite aux résultats des réseaux de neurones, nous avons développé une série d'approches qui permettent de combiner les approches que nous avons développées. L'idée étant que, dans le chapitre 2, nous avons observé que certains modèles avaient une tendance à surprédire alors que d'autres avaient une tendance à sous-prédire. Cette observation nous laisse croire que les modèles n'ont pas les mêmes forces et les mêmes faiblesses et qu'il était envisageable d'exploiter les forces de chacun des modèles en les combinant. Cette intuition nous a poussés à étudier la littérature afin de trouver des techniques nous permettant de combiner les approches que nous avons faites précédemment. Cependant, ces travaux n'ont pas montré de bons résultats ou des résultats consistants. Nos analyses nous ont laissés penser qu'encore une fois, une partie de notre problème était un manque de données. Nous sommes alors sortis des sentiers battus en développant une variation de l'approche utilisant la prédiction par *métamodèles*. Avec cette nouvelle approche, nous avons trouvé une manière de combiner la régression et la classification pour améliorer nos prédictions. Cette nouvelle approche a montré des résultats impressionnants pour une seule combinaison de prédicteurs et montre des résultats similaires à nos meilleurs résultats obtenus avec les réseaux de neurones et les améliorations aux réseaux de neurones (discutées ci-dessous). Cependant, cette approche ne fonctionne que dans une combinaison particulière, ce qui nous fait douter qu'elle soit généralisable, ex., à d'autres jeux de données.

Notre dernière étape visait à améliorer la qualité de la prédiction en utilisant les réseaux de neurones. Les résultats obtenus dans le chapitre 4 démontrent que la régression semble moins prometteuse que la classification pour prédire le panier de produits associé à un billot. Cependant, nous savions qu'il était plus facile d'utiliser cette approche pour créer de nouvelles fonctions de perte. Malgré le fait que nos premières approches aient semblé peu prometteuses, chacune d'elle nous a apporté une série de connaissances qui nous ont permis de développer des fonctions de pertes de plus en plus efficaces. Nous avons commencé par permettre l'échange de produits de sciage similaires en nous basant sur les connaissances d'experts. Bien que l'approche n'ait pas donné de bons résultats, elle nous a permis de comprendre la complexité d'exploiter le savoir des experts du domaine. Par la suite, nous avons développé une fonction de perte se basant sur la valeur d'optimisation pour prédire le panier de produits associé à un billot. Cette fois, nous avons appris que cette approche n'était pas pertinente puisque plusieurs produits avaient la même valeur d'optimisation. Finalement, nous avons développé deux fonctions de pertes utilisant un calcul similaire à la métrique F1, dont une utilisant l'erreur quadratique moyenne. La variante utilisant l'erreur quadratique moyenne a démontré une amélioration de plus de 10% par rapport à l'état de l'art dans ce domaine qui utilisait les arbres de décisions. Un dernier point important est que les nouvelles fonctions de pertes que nous avons développées semblent stables dans la mesure où elles montrent des améliorations

pour toutes les combinaisons de représentation et d'architecture de réseau de neurones et il ne s'agit pas de combinaisons très précises comme nous l'avons observé dans le chapitre 5.

Comme nous pouvons le constater, ces travaux de recherches touchent à plusieurs aspects du problème. Effectivement, nous avons observé des gains en explorant différentes représentations du billot et plusieurs approches d'apprentissage automatique. Sans considérer le problème de la prédiction du panier de produits associé à un billot comme étant complètement résolu, nous pouvons affirmer que nos approches sont maintenant l'état de l'art en la matière. Avec ces résultats, il serait intéressant de valider si nos travaux permettent d'améliorer la planification stratégique pour l'industrie forestière. Si tel était le cas, on pourrait parler d'économies importantes pour les entreprises de produits forestiers.

Bibliographie

Abd Elrahman, S. M. et A. Abraham (2013). « A review of class imbalance problem ». Dans : *Journal of Network and Innovative Computing* 1.2013, p. 332-340.

Altman, N. S. (1992). « An introduction to kernel and nearest-neighbor nonparametric regression ». Dans : *The American Statistician* 46.3, p. 175-185.

Barandela, R., J. S. Sánchez, V. Garcıa et E. Rangel (2003). « Strategies for learning in class imbalance problems ». Dans : *Pattern Recognition* 36.3, p. 849-851.

Barse, E. L., H. Kvarnstrom et E. Jonsson (2003). « Synthesizing test data for fraud detection systems ». Dans : *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE, p. 384-394.

Besl, P. J. et N. D. McKay (1992). « Method for registration of 3-D shapes ». Dans : *Sensor fusion IV : control paradigms and data structures.* T. 1611. International Society for Optics et Photonics, p. 586-606.

Borchani, H., G. Varando, C. Bielza et P. Larranaga (2015). « A survey on multi-output regression ». Dans : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 5.5, p. 216-233.

Bowles, C., L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. Valdés Hernández, J. Wardlaw et D. Rueckert (2018). « GAN Augmentation : Augmenting Training Data using Generative Adversarial Networks ». Dans : *arXiv e-prints*, arXiv-1810.

Breiman, L. (1996). « Bagging predictors ». Dans : *Machine learning* 24.2, p. 123-140.

– (2001). « Random forests ». Dans : *Machine learning* 45.1, p. 5-32.

Breiman, L., J. Friedman, C. J. Stone et R. A. Olshen (1984). *Classification and regression trees.* CRC Press.

Brownlee, J. (2020). *Data preparation for machine learning : data cleaning, feature selection, and data transforms in Python.* Machine Learning Mastery.

Cai, L., Y. Yu, S. Zhang, Y. Song, Z. Xiong et T. Zhou (2020). « A Sample-Rebalanced Outlier-Rejected k-Nearest Neighbor Regression Model for Short-Term Traffic Flow Forecasting ». Dans : *IEEE access* 8, p. 22686-22696.

Chabanet, S., V. Chazelle, P. Thomas et H. B. El-Haouzi (2021). « Dissimilarity to Class Medoids as Features for 3D Point Cloud Classification ». Dans : *IFIP International Conference on Advances in Production Management Systems.* Springer, p. 573-581.

Chabanet, S., H. B. El-Haouzi et P. Thomas (2021). « Coupling digital simulation and machine learning metamodel through an active learning approach in Industry 4.0 context ». Dans : *Computers in Industry* 133, p. 103529.

Chabanet, S., P. Thomas, H. B. El-Haouzi, M. Morin et J. Gaudreault (2021). « A kNN approach based on ICP metrics for 3D scans matching : an application to the sawing process ». Dans : *17th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2021*.

Chen, Y. et G. Medioni (1992). « Object modelling by registration of multiple range images ». Dans : *Image and vision computing* 10.3, p. 145-155.

Ciresan, D. C., U. Meier, J. Masci, L. M. Gambardella et J. Schmidhuber (2011). « Flexible, high performance convolutional neural networks for image classification ». Dans : *Twenty-second international joint conference on artificial intelligence*.

Dekking, F. M., C. Kraaikamp, H. P. Lopuhaä et L. E. Meester (2005). *A Modern Introduction to Probability and Statistics : Understanding why and how*. T. 488. Springer.

Dierckx, P. (1995). *Curve and surface fitting with splines*. Oxford University Press.

Dzeroski, S. et B. Zenko (2002). « Is combining classifiers better than selecting the best one ? ». Dans : *ICML*. T. 2002. Citeseer, 123e30.

Fix, E. (1985). *Discriminatory analysis : nonparametric discrimination, consistency properties*. T. 1. USAF school of Aviation Medicine.

Foley, J. D., A. van Dam, J. F. Hughes et S. K. Feiner (1990). « Spatial-partitioning representations ; Surface detail ». Dans : *Computer Graphics : Principles and Practice*.

FPInnovations (2014). *Optitek 10. User's Manual*.

Glasmachers, T. (2017). « Limits of end-to-end learning ». Dans : *Asian Conference on Machine Learning*. PMLR, p. 17-32.

Glorot, X., A. Bordes et Y. Bengio (2011). « Deep sparse rectifier neural networks ». Dans : *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop et Conference Proceedings, p. 315-323.

Graves, A., G. Wayne et I. Danihelka (2014). « Neural turing machines ». Dans : *arXiv preprint arXiv :1410.5401*.

Green, T. R. G. et M. Petre (1996). « Usability analysis of visual programming environments : a 'cognitive dimensions' framework ». Dans : *Journal of Visual Languages & Computing* 7.2, p. 131-174.

Gujjar, H. S. (2018). « A Comparative Study of VoxelNet and PointNet for 3D Object Detection in Car by Using KITTI Benchmark ». Dans : *International Journal of Information Communication Technologies and Human Development (IJICTHD)* 10.3, p. 28-38.

Guo, X., Y. Yin, C. Dong, G. Yang et G. Zhou (2008). « On the class imbalance problem ». Dans : *2008 Fourth international conference on natural computation*. T. 4. IEEE, p. 192-201.

HALCO (2016). *HALCO Software Systems Ltd.* http://www.halcosoftware.com. [2021-02].

Hastie, T., R. Tibshirani et J. Friedman (2009). « Overview of supervised learning ». Dans : *The elements of statistical learning*. Springer, p. 9-41.

He, K., X. Zhang, S. Ren et J. Sun (2016). « Deep residual learning for image recognition ». Dans : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770-778.

Hsieh, J. (2003). *Computed tomography : principles, design, artifacts, and recent advances*. T. 114. SPIE Press.

Huang, G., Z. Liu, L. Van Der Maaten et K. Q. Weinberger (2017). « Densely connected convolutional networks ». Dans : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 4700-4708.

Hubel, D. H. et T. N. Wiesel (1968). « Receptive fields and functional architecture of monkey striate cortex ». Dans : *The Journal of physiology* 195.1, p. 215-243.

Ji, S., W. Xu, M. Yang et K. Yu (2012). « 3D convolutional neural networks for human action recognition ». Dans : *IEEE transactions on pattern analysis and machine intelligence* 35.1, p. 221-231.

Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar et L. Fei-Fei (2014). « Large-scale video classification with convolutional neural networks ». Dans : *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, p. 1725-1732.

Koehrsen, W. (2018). « Overfitting vs. underfitting : A complete example ». Dans : *Towards Data Science*.

Lan, T., H. Hu, C. Jiang, G. Yang et Z. Zhao (2020). « A comparative study of decision tree, random forest, and convolutional neural network for spread-F identification ». Dans : *Advances in Space Research* 65.8, p. 2052-2061.

Law, A. M. et W. D. Kelton (2007). *Simulation Modeling and Analysis, The McGraaw-Hill Companies*.

LeCun, Y. (1998). « The MNIST database of handwritten digits ». Dans : *http ://yann. lecun. com/exdb/mnist/*.

LeCun, Y. et al. (2015). « LeNet-5, convolutional neural networks ». Dans : *URL : http ://yann. lecun. com/exdb/lenet* 20.5, p. 14.

LeCun, Y., L. Bottou, Y. Bengio et P. Haffner (1998). « Gradient-based learning applied to document recognition ». Dans : *Proceedings of the IEEE* 86.11, p. 2278-2324.

Levoy, M. et T. Whitted (1985). *The use of points as a display primitive*. Citeseer.

Lin, W., J. Wang et R. E. Thomas (2010). « A three-dimensional optimal sawing system for small sawmills in central Appalachia ». Dans : *17th Central Hardwood Forest Conference*. T. 78, p. 67.

Liu, M., M. Wang, J. Wang et D. Li (2013). « Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification : Application to the recognition of orange beverage and Chinese vinegar ». Dans : *Sensors and Actuators B : Chemical* 177, p. 970-980.

Luo, G. (2016). « A review of automatic selection methods for machine learning algorithms and hyper-parameter values ». Dans : *Network Modeling Analysis in Health Informatics and Bioinformatics* 5.1, p. 1-16.

Makhzani, A., J. Shlens, N. Jaitly, I. Goodfellow et B. Frey (2015). « Adversarial autoencoders ». Dans : *arXiv preprint arXiv :1511.05644*.

Mani, I. et I. Zhang (2003). « kNN approach to unbalanced data distributions : a case study involving information extraction ». Dans : *Proceedings of workshop on learning from imbalanced datasets*. T. 126. ICML United States.

Martineau, V., J. Gaudreault, M. Morin et S. Vallerand (2021). « Quality of sawmilling output predictions according to the size of the lot-The size matters ! ». Dans : *International Conference on Industrial Engineering and Quality*.

Martineau, V., M. Morin, J. Gaudreault, P. Thomas et H. El-Haouzi (2021). « Neural network architectures and feature extraction for lumber production prediction ». Dans : *The 34th Canadian Conference on Artificial Intelligence*.

Mitchell, T. (1997). *Machine learning*. McGraw Hill Burr Ridge.

Morin, M., J. Gaudreault, E. Brotherton, F. Paradis, A. Rolland, J. Wery et F. Laviolette (2020). « Machine learning-based models of sawmills for better wood allocation planning ». Dans : *International Journal of Production Economics* 222, p. 107508.

Morin, M., F. Paradis, A. Rolland, J. Wery, F. Laviolette et F. Laviolette (2015). « Machine learning-based metamodels for sawing simulation ». Dans : *2015 Winter Simulation Conference (WSC)*. IEEE, p. 2160-2171.

Nair, V. et G. E. Hinton (2010). « Rectified linear units improve restricted boltzmann machines ». Dans : *ICML*.

Nitze, I., U. Schulthess et H. Asche (2012). « Comparison of machine learning algorithms random forest, artificial neural network and support vector machine to maximum likelihood for supervised crop type classification ». Dans : *Proceedings of the 4th GEOBIA, Rio de Janeiro, Brazil* 79, p. 3540.

Pascanu, R., T. Mikolov et Y. Bengio (2013). « On the difficulty of training recurrent neural networks ». Dans : *International conference on machine learning*. PMLR, p. 1310-1318.

Powers, D. M. (2020). « Evaluation : from precision, recall and F-measure to ROC, informedness, markedness and correlation ». Dans : *arXiv preprint arXiv :2010.16061*.

Qi, C. R., H. Su, K. Mo et L. J. Guibas (2017). « Pointnet : Deep learning on point sets for 3d classification and segmentation ». Dans : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 652-660.

Quinlan, J. R. (1986). « Induction of Decision Trees ». Dans : *MACH. LEARN* 1, p. 81-106.

Rijsbergen, C. van (1979). *Information Retrieval, 2nd edButterworths*.

Rönnqvist, M. (2003). « Optimization in forestry ». Dans : *Mathematical programming* 97.1, p. 267-284.

Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Rapp. tech. Cornell Aeronautical Lab Inc Buffalo NY.

Rumelhart, D. E., G. E. Hinton et R. J. Williams (1985). *Learning internal representations by error propagation*. Rapp. tech. California Univ San Diego La Jolla Inst for Cognitive Science.

Russell, S. et P. Norvig (2016). *Artificial Intelligence : A Modern Approach*. CreateSpace Independent Publishing Platform.

Schmidhuber, J. (2015). « Deep learning in neural networks : An overview ». Dans : *Neural networks* 61, p. 85-117.

Schonlau, M. et R. Y. Zou (2020). « The random forest algorithm for statistical learning ». Dans : *The Stata Journal* 20.1, p. 3-29.

Selma, C., H. B. El Haouzi, P. Thomas, J. Gaudreault et M. Morin (2018). « An iterative closest point method for measuring the level of similarity of 3D log scans in wood industry ». Dans : *Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, p. 433-444.

Shannon, R. E. (1975). *Systems simulation ; the art and science*. Rapp. tech.

– (1998). « Introduction to the art and science of simulation ». Dans : *1998 winter simulation conference. proceedings (cat. no. 98ch36274)*. T. 1. IEEE, p. 7-14.

Srivastava, R. K., K. Greff et J. Schmidhuber (p. d.). « Highway Networks ». Dans : ().

Thomas, R. E. (2013). « RAYSAW : A log sawing simulator for 3D laser-scanned hardwood logs ». Dans : *Proceedings, 18th Central Hardwood Forest Conference*. T. 117, p. 325-334.

Todoroki, C. et al. (1990). « AUTOSAW system for sawing simulation ». Dans : *New Zealand Journal of Forestry Science* 20.3, p. 332-348.

Ursella, E., F. Giudiceandrea et M. Boschetti (2018). « A Fast and Continuous CT scanner for the optimization of logs in a sawmill ». Dans : *8th Conference on Industrial Computed Tomography (iCT 2018) at Wels, Austria*. T. 2.

Wang, J., L. Perez et al. (2017). « The effectiveness of data augmentation in image classification using deep learning ». Dans : *Convolutional Neural Networks Vision Recognition* 11, p. 1-8.

Wolpert, D. H. (1992). « Stacked generalization ». Dans : *Neural networks* 5.2, p. 241-259.

Zhou, Y. et O. Tuzel (2018). « Voxelnet : End-to-end learning for point cloud based 3d object detection ». Dans : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 4490-4499.