IFT-4102 et IFT-7025

Techniques avancées en Intelligence Artificielle

Rapport du TP4

Équipe 20

Membres de l'équipe :

Emma KILBERTUS - 537305286

Maxence QUELENNEC - 536899537

Estrella PAOLI - 537026544

Travail présenté à :

Brahim Chaib-draa

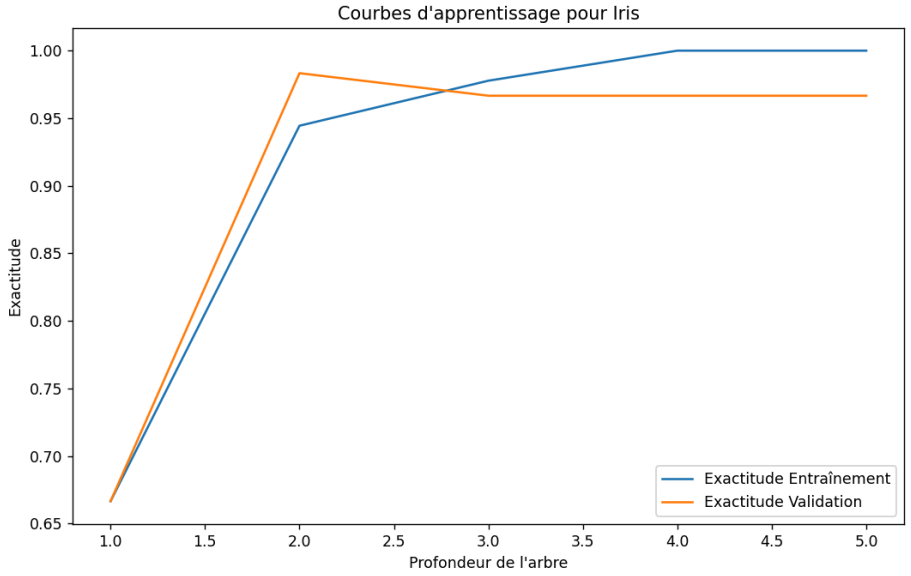
Remise :

22 décembre 2024

I. Arbres de décision

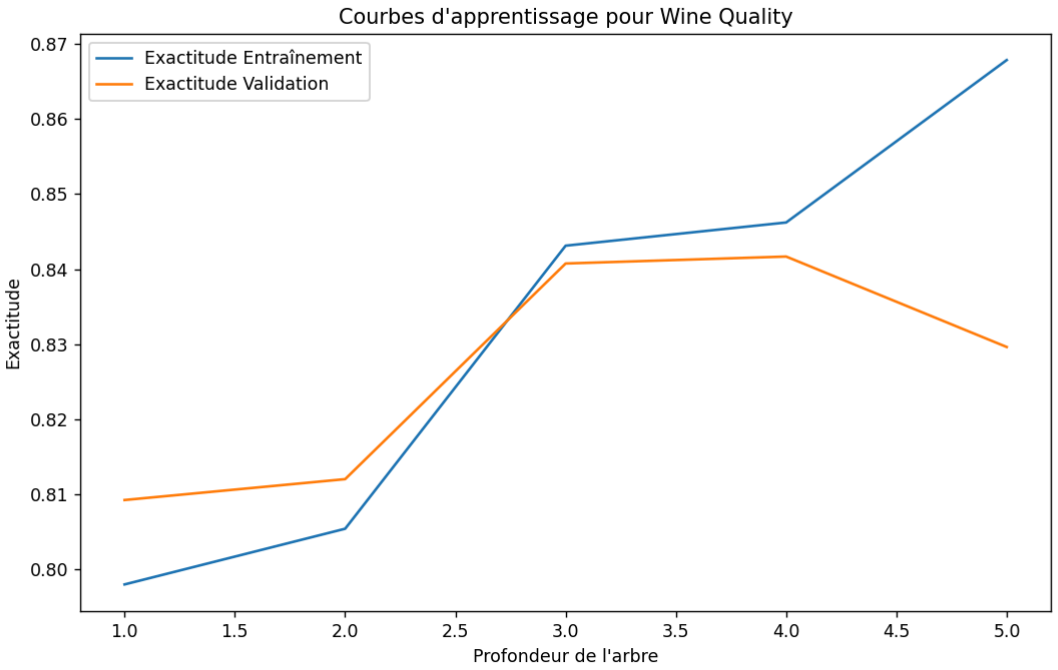
- *Courbes d'apprentissage :*

Dataset Iris :



La capture ci-dessus indique que l'exactitude du modèle augmente rapidement étroitement avec l'augmentation de la profondeur de l'arbre (dès l'ajout d'une deuxième couche). À partir de de la troisième couche, le modèle atteint un plateau. Donc on peut conclure que le nombre de couches le plus performant pour l'arbre décisionnel sera de 2 ou 3 pour ce dataset (couches = nœuds).

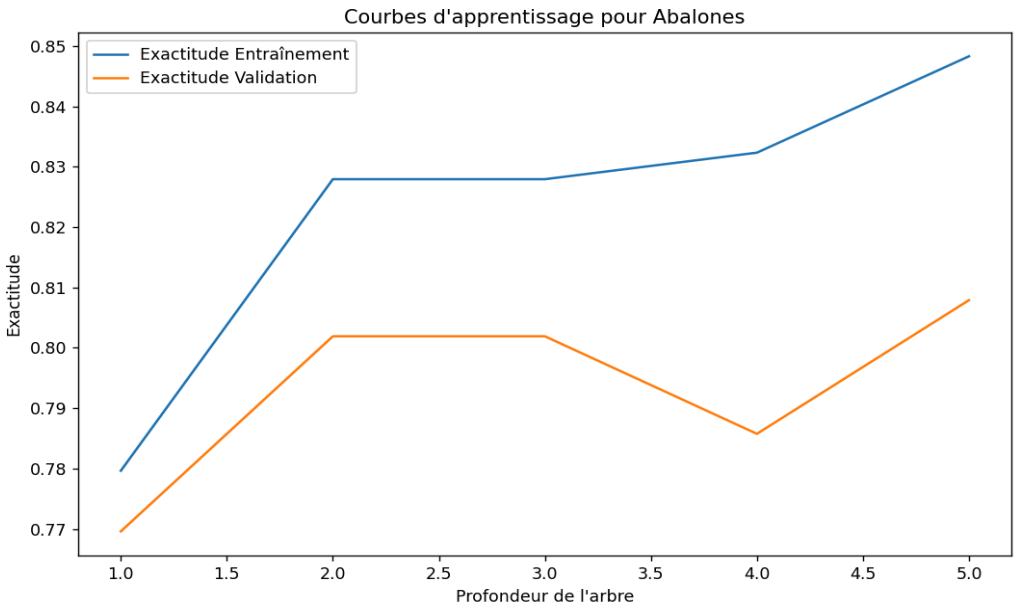
Dataset Wine Quality :



Pour le dataset sur la qualité du vin, la courbe d'apprentissage nous indique que le modèle subit un léger sous-apprentissage avec une ou deux couches car les exactitudes sont les plus faibles. De plus, l'exactitude de validation est meilleure que l'exactitude d'entraînement, ce qui n'est pas un critère qui prime quand on parle de sous-entraînement mais cela vient supporter cette idée. Lorsque le modèle possède quatre ou cinq couches, l'exactitude sur l'entraînement est supérieure à l'exactitude de validation, notamment en présence de cinq couches. Cela n'est pas une instance de surapprentissage, mais ce n'est pas le meilleur équilibre. Ce dernier se situe lorsque le modèle a 3 couches, puisque l'exactitude d'entraînement est élevée, l'exactitude de validation également et proche de l'entraînement.

L'arbre décisionnel trouve donc pour le dataset 1 et 2 que le meilleur nombre de couches est 3. Cela renvoie au travail pratique 3 dans lequel nous faisions remarquer que les deux premiers datasets étaient assez équilibrés, donc traités de manière équivalente intrinsèquement à chaque modèle. C'est donc le cas également pour l'arbre décisionnel.

<u>Dataset Abalones :</u>



La courbe d'apprentissage de la capture ci-dessus montre que le modèle d'arbre décisionnel ne trouve pas de bon équilibre pour le dataset compliqué des Ormeaux (Abalones). À 2 ou 3 couches, il y a un certain compromis intéressant, à 5 couches également, mais il n'y a pas d'équilibre idéal qui est atteint. L'analyse des métriques qui va suivre pourra venir ajouter des informations supplémentaires quant à la performance du modèle sur ce dataset.

- *Évaluation sur données tests :*

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.83	0.81
Matrice de confusion		$\begin{bmatrix} 21 & 0 & 0 \\ 0 & 18 & 2 \\ 0 & 0 & 19 \end{bmatrix}$	$\begin{bmatrix} 553 & 75 \\ 109 & 343 \end{bmatrix}$	$\begin{bmatrix} 73 & 113 & 0 \\ 25 & 1242 & 19 \\ 1 & 163 & 35 \end{bmatrix}$
Précision	Classe 0	1	0.835	0.7373
	Classe 1	1	0.82	0.8181
	Classe 2	0.90	NA	0.6481
Rappel	Classe 0	1	0.88	0.392
	Classe 1	0.9	0.76	0.966
	Classe 2	1	NA	0.176
F1-score	Classe 0	1	0.857	0.512
	Classe 1	0.947	0.788	0.886
	Classe 2	0.95	NA	0.277

Les courbes d'apprentissage pour le dataset Iris nous montraient des exactitudes élevées; les métriques viennent le confirmer. Les mesures de précision de rappel et

score F1 sont toutes proches de 1. Le dataset Iris a été efficacement traité par l'arbre malgré la faible quantité de données. Par efficacement traité, on entend que l'entraînement du modèle a été performant de sorte que ce dernier réussit fortement en phase de test.

Pour le dataset de qualité de vin, le modèle en phase de test est très bon puisque les métriques sont très fortes. La classe 1 est assurément bien identifiée par le modèle, mais la classe 2 un peu moins. Toutefois, il semble que ce modèle ait le mieux performé parmi les quatres modèles utilisés, supposition sur laquelle nous reviendrons dans la comparaison inter-modèle.

Pour le dataset des ormeaux, le modèle est significativement très performant avec la classe 2. Cette caractéristique est particulièrement intéressante car elle est potentiellement unique à ce modèle, nous devrons revenir sur cela lors de la comparaison inter-modèles. La mesure de rappel des classes 0 et 2 est nettement très faible, ce qui signifie que le modèle échoue profusément à identifier les vrais positifs. Ces échecs sont sans surprise car comme notre TP3 avait déjà souligné, les classes 0 et 2 sont clairement sous-représentées, d'où les problèmes d'identification. En cela, le modèle d'arbre décisionnel nous révèle déjà qu'il ne se distingue pas des autres modèles pour sa capacité à gérer les mauvaises distributions (malgré la présence de "poids" de classes) mais nous y reviendrons plus tard.

- *Comparaison avec scikit-learn :*

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.85	0.8
Matrice de confusion		$\begin{bmatrix} 21 & 0 & 0 \\ 0 & 18 & 2 \\ 0 & 0 & 19 \end{bmatrix}$	$\begin{bmatrix} 561 & 67 \\ 96 & 356 \end{bmatrix}$	$\begin{bmatrix} 85 & 100 & 1 \\ 36 & 1227 & 23 \\ 1 & 169 & 29 \end{bmatrix}$
Précision	Classe 0	1	0.85	0.70
	Classe 1	1	0.84	0.82
	Classe 2	0.9	NA	0.55
Rappel	Classe 0	1	0.89	0.46
	Classe 1	0.9	0.79	0.95
	Classe 2	1	NA	0.15
F1-score	Classe 0	1	0.87	0.55
	Classe 1	0.95	0.81	0.88
	Classe 2	0.95	NA	0.23

Pour le premier dataset, notre modèle a fonctionné pareillement au modèle de scikit-learn, c'est-à-dire très performant dans l'identification des nœuds. Pour le deuxième dataset, le modèle de scikit-learn réussit mieux, ce qui nous étonne puisque cela signifierait que nous sommes potentiellement non optimaux sur notre calcul des seuils. L'autre étonnement est que notre modèle réussit mieux que le modèle de scikit-learn pour le troisième dataset. Nous utilisons la formule d'entropie classique (celle vu en cours). Il est possible qu'elle soit plus adaptée pour le dataset complexe des ormeaux, sans que l'on s'y attende (i.e ces bons résultats sont le fruit du hasard dans notre cas).

- Analyse de l'élagage

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.6	0.79
Matrice de confusion		$\begin{bmatrix} 21 & 0 & 0 \\ 1 & 18 & 1 \\ 0 & 0 & 19 \end{bmatrix}$	$\begin{bmatrix} 615 & 13 \\ 424 & 28 \end{bmatrix}$	$\begin{bmatrix} 42 & 144 & 0 \\ 6 & 1280 & 0 \\ 0 & 199 & 0 \end{bmatrix}$
Précision	Classe 0	0.95	0.59	0.875
	Classe 1	1	0.68	0.789
	Classe 2	0.95	NA	0
Rappel	Classe 0	1	0.98	0.226
	Classe 1	0.9	0.06	0.995
	Classe 2	1	NA	0
F1-score	Classe 0	0.977	0.74	0.359
	Classe 1	0.947	0.11	0.88
	Classe 2	0.974	NA	0

Dans notre cas, l'élagage offre de moins bonne performance. Le modèle reste pareil pour le dataset Iris, mais pour les deux autres datasets, il devient très bon pour certaines classes (classe 0 du Wine Quality et classe 2 du Abalones) et très mauvais pour les autres. Notre élagage est classique puisque nous simplifions l'arbre en supprimant l'arbre droit et l'arbre gauche d'un nœud, qui devient une feuille, sans réduire la capacité de généralisation de l'arbre. Cette baisse de performance peut être expliquée par le fait qu'on ne prenne pas en compte la pondération des différentes classes, bien que ce ne soit pas un problème majeur du corpus de Wine Quality. Comme autre piste d'amélioration possible, une mise en place de validation croisée pour l'élagage est tout à fait pertinente.

II. Réseaux de neurones artificiels

Notre implémentation se base sur un réseau de neurones avec une couche cachée variable. Le taux d'apprentissage, le nombre d'épisodes (epoch) et le nombre de neurones utilisés dans la couche cachée sont différents pour chaque dataset afin de maximiser la performance du modèle. Les choix que nous avons faits pour chaque dataset sont présentés plus bas dans la partie d'évaluation sur les données de test.

- *Initialisation des poids et biais :*

La technique d'initialisation des poids choisie est l'initialisation Glorot (ou Xavier). Chaque poids est initialisé selon une distribution normale dont la variance dépend du nombre de neurones d'entrée et de sortie. Pour chaque poids w, nous utilisons :

$$w \sim N(0, \frac{2}{input_{neurons} + output_{neurons}})$$

Nous avons choisi cette approche car elle garantit une échelle adaptée pour les valeurs des poids durant les phases de propagation avant (forward) et arrière (backward), maintenant une variance équilibrée. De plus, elle est particulièrement adaptée aux fonctions d'activation symétriques comme la sigmoïde utilisée dans notre réseau.

Nous avons choisi d'initialiser les biais à 0 pour éviter d'introduire des biais initiaux dans l'apprentissage du réseau.

- *Paramètres d'apprentissage :*

Les paramètres d'initialisation du modèle retenus sont :

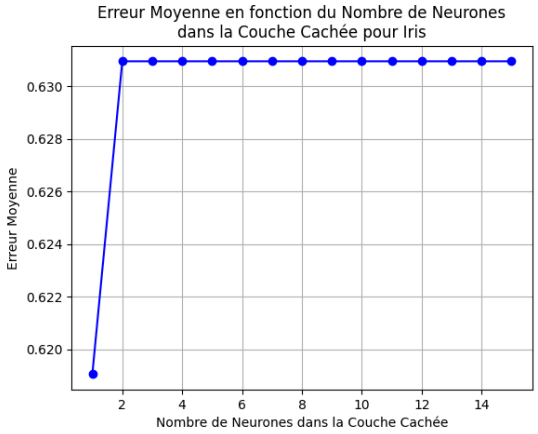
- Un taux d'apprentissage α de 0.01
- Un nombre de neurones d'entrée égal au nombre de caractéristiques
- Un nombre de neurones de sortie égal au nombre de classes
- Un nombre de neurones cachés calculé avec la formule suivante : $hiddenNeurons = \frac{inputNeurons + outputNeurons}{2}$

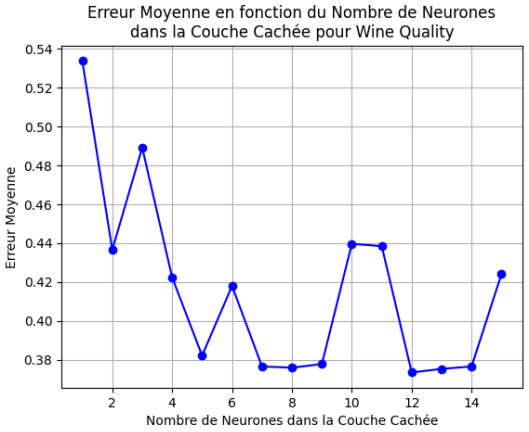
- *Analyse des hyperparamètres :*

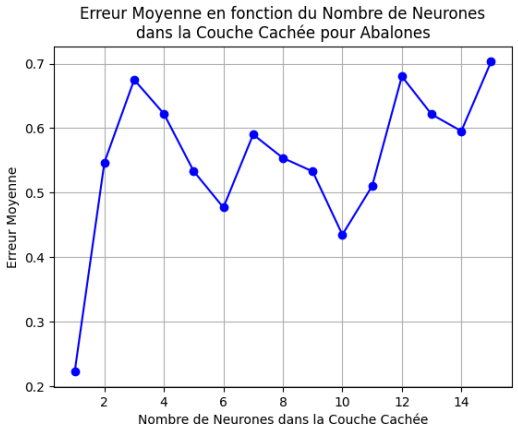
Pour la validation croisée, nous utilisons k=7 folds avec un taux d'apprentissage α=0.01, identique à notre implémentation personnelle.

Nombre de neurones dans la couche cachée :
Nous avons testé des valeurs allant de 1 à 15 pour l'hyperparamètre du nombre de neurones dans une couche cachée afin de déterminer le nombre optimal. Les résultats des erreurs obtenues pour chaque dataset sont représentés par les graphiques ci-dessous.

Les résultats des erreurs obtenues pour chaque dataset sont représentés par les graphiques ci-dessous :







Dataset Iris :

Sur le premier graphique, on observe que l'erreur moyenne est minimale lorsqu'il y a un seul neurone dans la couche cachée. De plus, les erreurs moyennes restent très similaires lorsqu'on ajoute plus d'un neurone dans la couche cachée. Cela signifie que la performance du modèle n'est pas améliorée en ajoutant des neurones cachés. Ce résultat pourrait indiquer que le dataset est petit et ne contient pas suffisamment de données pour entraîner correctement le modèle, ou que les données sont simples et qu'un seul neurone caché suffit pour capturer la structure sous-jacente des données.

Conclusion pour Iris : L'erreur moyenne est minimisée avec 1 neurone dans la couche cachée. Nous choisissons donc d'utiliser 1 neurone pour ce dataset.

Dataset Wine Quality :

Sur le deuxième graphique, on observe que l'erreur moyenne est minimale lorsqu'il y a entre 7 et 9 ou entre 12 et 14 neurones. L'erreur minimale est obtenue avec 12 neurones dans la couche cachée. Cela signifie que le modèle est le plus capable d'extraire des caractéristiques du dataset lorsqu'il utilise 12 neurones cachés. Lorsque le modèle utilise moins de 12 neurones, il est trop simple et ne peut pas apprendre correctement les relations dans les données, ce qui conduit à une erreur élevée par sous-apprentissage. Au contraire, lorsqu'il utilise plus de 12 neurones, il devient trop spécifique aux données d'entraînement, ce qui conduit à une erreur élevée par sur-apprentissage.

Conclusion pour Wine Quality : L'erreur moyenne est minimisée avec 12 neurones dans la couche cachée. Nous choisissons donc d'utiliser 12 neurones pour ce dataset.

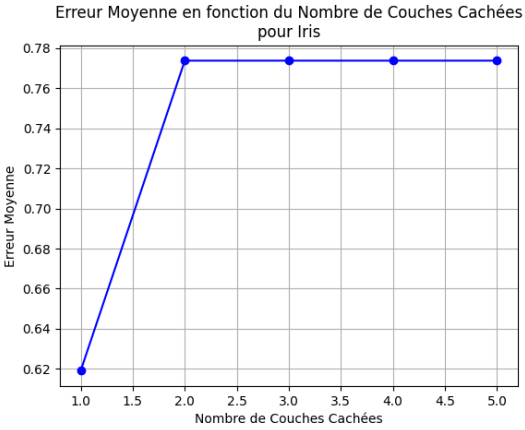
Dataset Abalones :

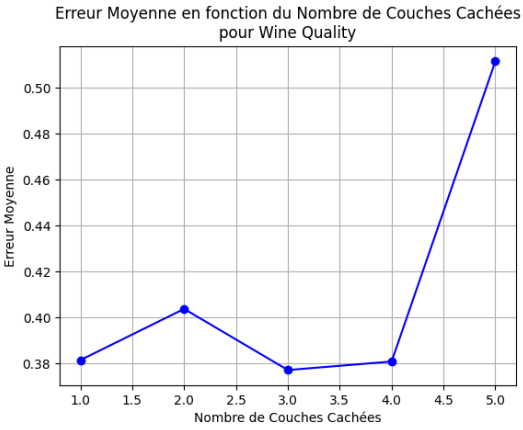
Sur le troisième graphique, il est possible d'observer que l'erreur moyenne est minimale lorsqu'il y a un seul neurone dans la couche cachée. Cependant les performances sont plus variables en augmentant le nombre de neurones que pour le premier dataset. De même que pour le premier graphique, cela signifie que la performance du modèle n'est pas améliorée en ajoutant des neurones cachés, ce qui pourrait indiquer que le dataset est petit et qu'il ne contient pas suffisamment de données pour entraîner correctement le modèle ou que les données qu'il contient sont simples et qu'un seul neurone caché suffit pour capturer la structure sous-jacente des données.

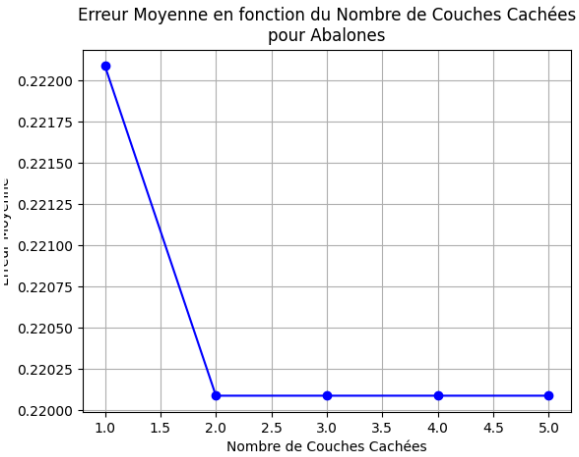
Conclusion pour Abalones : L'erreur moyenne est minimisée avec 1 neurone dans la couche cachée. Nous choisissons donc d'utiliser 1 neurone pour ce dataset.

Nombre de couches cachées :

Nous avons testé des valeurs allant de 1 à 5 pour l'hyperparamètre du nombre de couches cachées, en utilisant le nombre optimal de neurones trouvé précédemment pour chaque dataset. Les résultats des erreurs obtenues pour chaque dataset sont représentés par les graphiques ci-dessous.







Les résultats des erreurs obtenues pour chaque dataset sont représentés par les graphiques ci-dessous :

<u>Dataset Iris :</u>

Sur le premier graphique, on observe que l'erreur moyenne est minimale avec une seule couche cachée. De plus, les erreurs moyennes restent très similaires lorsqu'on ajoute des couches supplémentaires. Cela indique que l'ajout de couches supplémentaires ne contribue pas à une amélioration de la performance du modèle. Ce résultat pourrait être dû au fait que les données sont suffisamment simples pour être capturées par une seule couche cachée, ou que le modèle ne dispose pas de suffisamment de données d'entraînement pour exploiter l'augmentation de la profondeur. Il est également possible que le modèle fasse du sur-apprentissage avec plusieurs couches cachées, ce qui expliquerait pourquoi l'erreur moyenne reste

stable.
Conclusion pour Iris : L'erreur moyenne est minimisée avec 1 couche cachée. Nous choisissons donc d'utiliser 1 couche cachée avec 1 neurone.

Dataset Wine Quality :
Sur le deuxième graphique, on observe que l'erreur moyenne est minimale avec 3 couches cachées. De plus, l'erreur moyenne augmente fortement avec 5 couches cachées. Cela signifie qu'à partir de 5 couches cachées, le modèle devient trop complexe pour le problème à résoudre, ce qui peut être dû à un sur-apprentissage ou à une difficulté d'entraînement à cause de la complexité du réseau. En utilisant moins de 3 couches cachées, le modèle semble être en sous-apprentissage car son réseau est trop simple pour capturer l'ensemble des relations entre les caractéristiques dans ce dataset.
Conclusion pour Wine Quality : L'erreur moyenne est minimisée avec 3 couches cachées. Nous choisissons donc d'utiliser 3 couches cachées ayant chacune 12 neurones.

Dataset Abalones :
Sur le troisième graphique, on observe que l'erreur moyenne est minimale avec 2 couches cachées. L'erreur moyenne diminue fortement lorsque le modèle utilise plus d'une couche cachée, et ces erreurs restent similaires entre 2 et 5 couches cachées. Cela suggère qu'à partir de 2 couches cachées, l'ajout de couches supplémentaires n'apporte plus de bénéfice en termes de performance. En utilisant une seule couche cachée, le modèle semble être en sous-apprentissage car son réseau est trop simple pour capturer l'ensemble des relations entre les caractéristiques dans ce dataset.
Conclusion pour Abalones : L'erreur moyenne est minimisée avec 2 couches cachées. Nous choisissons donc d'utiliser 2 couches cachées ayant chacune 1 neurone.

Vanishing Gradient :

Le phénomène de "vanishing gradient" est un problème majeur dans les réseaux de neurones profonds. Il se produit lorsque les gradients deviennent exponentiellement petits lors de la rétropropagation à travers de nombreuses couches.

Dans nos expérimentations, ce phénomène est particulièrement visible sur le dataset Wine Quality, où l'erreur augmente significativement avec 5 couches cachées. Les gradients s'atténuent progressivement lors de la rétropropagation, rendant l'apprentissage inefficace pour les premières couches du réseau.
Cette dégradation est due à la multiplication répétée de petites valeurs lors du calcul des gradients. Les dérivées de la fonction sigmoïde, comprises entre 0 et 1, se rapprochent de 0 pour des entrées éloignées de 0. Cela entraîne une atténuation des gradients à chaque couche traversée. Par conséquent, les poids des premières couches ne sont presque plus mis à jour, empêchant le réseau d'apprendre efficacement.
Ce phénomène explique pourquoi l'architecture optimale trouvée pour Wine Quality s'arrête à 3 couches cachées, et pourquoi Iris et Abalones performent mieux avec des architectures plus simples.

- *Évaluation sur données test :*

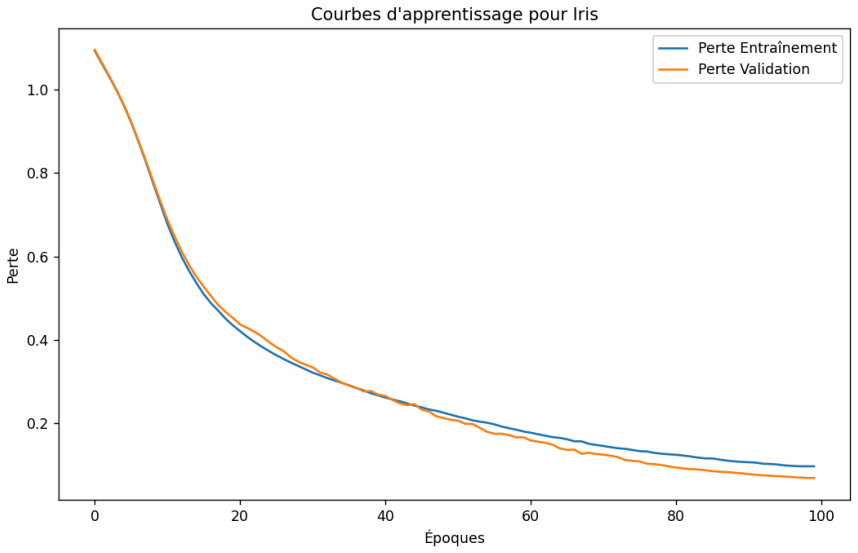
Premièrement, nous avons utilisé des paramètres différents pour le RN (réseau de neurones) en phase de tests que dans la phase d'entraînement qui précède. Les paramètres de la section précédente, c'est-à-dire le nombre d'épisodes, le taux d'entraînement alpha et le nombre de couches cachées ont été instanciées de sorte à maximiser l'entraînement des modèles (puisque la fonction de perte que nous présentons permet d'optimiser les poids du modèles en entraînement). En phase de tests, il s'avère que ces paramètres peuvent être amenés à être changés afin d'optimiser la capacité de prédiction du modèle.

Deuxièmement, dans notre cas, l'évaluation du modèle RN se distingue des trois autres modèles d'apprentissage élaborés précédemment (Knn, NB, DT) puisque les résultats changent à chaque exécution. Pour procéder à l'évaluation, nous avons lancé plusieurs exécution du modèle sur les dataset et avons pris une exécution qui donne des performances typiques. Parfois, la séparation des données en lot d'entraînement et en lot de test était telle que l'ensemble d'entraînement avait beaucoup de bruit dans les données et donc il avait du mal en phase de test. Ces cas ne sont pas majoritaires, mais surviennent fréquemment et c'est évidemment

une source de problème qui devrait être adressée. Notre conjecture (au sens d'hypothèse) est que le réseau de neurones n'est pas adapté à l'application présente. Cette inaptitude provient de sa nature: c'est un modèle d'apprentissage profond qui est conçu pour les tâches difficiles mais dont on possède une quantité de données massive (et alors il sera très performant). Ce n'est pas le cas ici, les données d'entraînement sont au plus de l'ordre du millier. Cela engendre des cas de "bruit" pour lesquels les données d'entraînement sont parsemées de fluctuations non pertinentes que le modèle mémorise.

Cette conjecture étant faite, ci-dessous suit les courbes d'apprentissages du RN pour les différents datasets lorsque les données d'entraînement n'ont pas de bruit :

➢ Dataset Iris



Le graphique des courbes d'apprentissage de la capture ci-dessus présente le nombre d'époques (epoch) utilisé lors de l'entraînement du modèle en fonction de la perte du modèle, c'est-à-dire la mesure de la différence des prédictions du modèle avec les valeurs réelles. Ce graphique est très pratique pour se lancer dans un processus itératif dans lequel on choisit des paramètres initiaux (taux d'apprentissage, nombre d'époques, architecture de couches cachées), on regarde l'allure du graphique et on ajuste les paramètres en conséquence jusqu'à obtenir les meilleures courbes. Pour le dataset Iris, nous avons pu aller jusqu'à la centaine d'époques avant de voir une stabilisation de la perte (un plateau). Cette perte du

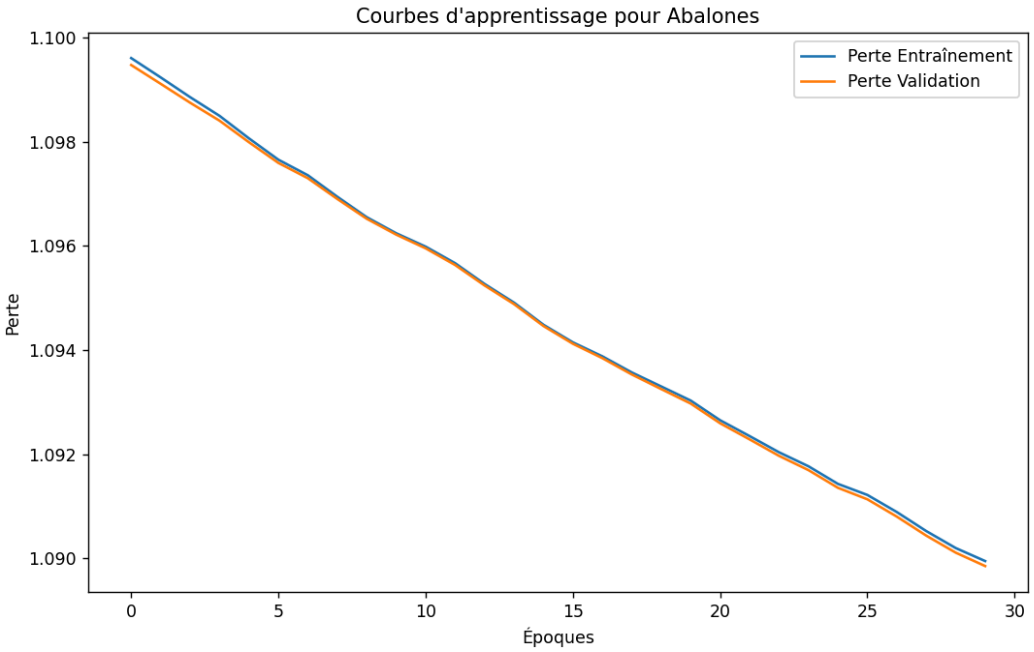
modèle est d'ailleurs très faible puisqu'elle est visuellement inférieure à 0.2, nous donnant une première impression que ce dernier a prédit efficacement les données de test du dataset Iris.

À noter que la décroissance des courbes est due à la fonction de gradients qui améliore à chaque époque les poids, ces derniers étant initialement aléatoires (selon la méthode Glorot).

➢ dataset Wine Quality

Le dataset Wine Quality est, contrairement au dataset Iris, déjà annonciateur de difficultés pour le modèle. Nous obtenons un plateau après 20 époques car les données sont plus complexes et donc le modèle atteint sa capacité limite de séparation linéaire plus rapidement. À partir de 25 époques, le modèle commence à subir du surapprentissage, ce qui est la suite logique lorsque le modèle a atteint un plateau. Il ne faut donc pas choisir plus de 30 époques, mais pas non plus se limiter à 20 car ce serait manquer de flexibilité. Au mieux, on atteint un perte de 0.4; en finalité la perte est de 0.45 ce qui est somme toute très bon. On peut donc s'attendre à une réussite convenable du modèle pour le dataset Wine Quality. L'analyse des métriques viendra valider ou infirmer cette attente.

➢ dataset Abalones



Le graphique ci-dessus montre clairement que le modèle a rencontré des difficultés non négligeables avec le dataset des ormeaux. Cette affirmation provient de la diminution minime de la perte malgré l'avancement des époques (1.1 à 1.09). Nous estimons que les hyperparamètres sont correctement instanciés, car la configuration choisie offrait le moins de bruit (la décroissance est faible, mais nettement linéaire). Cette faible diminution de la perte peut donc s'expliquer hypothétiquement par un mauvais choix de discriminants ou fonction de perte, et/ou par des données très complexes en quantité limitée. En tout cas, le modèle n'est pas en surapprentissage.

Bien que le modèle semble avoir rencontré des difficultés, la perte n'est que de 1, et nous avions des pertes parfois allant jusqu'à 15 ou 20 avec des configurations d'hyperparamètres différentes, donc on peut supposer que le modèle a modérément trimé.

Les métriques du modèle pour les différents dataset:

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.95	0.82	0.44
Matrice de confusion		$\begin{bmatrix} 21 & 0 & 0 \\ 0 & 17 & 3 \\ 0 & 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 503 & 125 \\ 64 & 388 \end{bmatrix}$	$\begin{bmatrix} 177 & 5 & 4 \\ 354 & 417 & 515 \\ 9 & 43 & 147 \end{bmatrix}$
Précision	Classe 0	1	0.887	0.44
	Classe 1	1	0.756	0.87
	Classe 2	0.86	NA	0.20
Rappel	Classe 0	1	0.80	0.92
	Classe 1	0.85	0.858	0.54
	Classe 2	1	NA	0.52
F1-score	Classe 0	1	0.84	0.60
	Classe 1	0.919	0.80	0.67
	Classe 2	0.927	NA	0.29

Les métriques confirment que le modèle a très bien performé avec le dataset Iris. Les RN sont une solution vraiment excellente pour les problèmes avec données complexes (linéairement non séparables et multidimensionnelles) puisque leur structure est très modelable. Toutefois, ils peuvent être très bon aussi avec les données simples et linéaires, ce qui en fait un type de modèle très polyvalent.

Pour le dataset Wine Quality, nous avions estimé que le RN aurait une bonne performance à partir des courbes d'apprentissage. Les scores F1 montrent que le modèle a été performant à la fois dans l'identification des vrais positifs et le taux de vrais positifs parmi ceux identifiés comme tels. Nous supposons que ce modèle est parmi les meilleurs pour ce dataset.

Pour le dataset des ormeaux, nous avions émis au moment des courbes d'apprentissage l'hypothèse que le modèle avait rencontré modérément des difficultés. L'exactitude de 0.44 ne vient pas améliorer ce cas, les scores F1 non plus. Il semblerait que ce modèle ait, contrairement à ce qu'on avait pensé en regardant les courbes d'apprentissage, rencontré le plus de difficultés sur l'ensemble des modèles. Nous rappelons donc nos explications sur le potentiel mauvais choix de la fonction de perte ou la potentielle mauvaise qualité de l'ensemble des données.

- *Comparaison avec scikit-learn :*

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.82	0.83
Matrice de confusion		$\begin{bmatrix} 21 & 0 & 0 \\ 0 & 18 & 2 \\ 0 & 0 & 19 \end{bmatrix}$	$\begin{bmatrix} 564 & 64 \\ 126 & 326 \end{bmatrix}$	$\begin{bmatrix} 106 & 79 & 1 \\ 43 & 1193 & 50 \\ 0 & 119 & 80 \end{bmatrix}$
Précision	Classe 0	1	0.82	0.71
	Classe 1	1	0.84	0.86
	Classe 2	0.9	NA	0.61
Rappel	Classe 0	1	0.90	0.57
	Classe 1	0.9	0.72	0.93
	Classe 2	1	NA	0.40
F1-score	Classe 0	1	0.86	0.63
	Classe 1	0.95	0.77	0.89
	Classe 2	0.95	NA	0.48

La classe MLP de scikit-learn a légèrement mieux réussi sur le premier dataset mais c'est assez négligeable. Sur le deuxième dataset, notre modèle a fait aussi bien, sinon un peu mieux, ce qui est excellent; cela peut s'expliquer par notre approche itérative d'optimisation manuelle des hyperparamètres, approche que la classe standard MLP s'abstient de faire. Pour le troisième dataset, le modèle de scikit-learn réussit nettement mieux; cela peut s'expliquer par la capacité de ce-dernier à choisir le meilleur discriminant, la meilleure initialisation des poids, la meilleure fonction de perte, etc.

III. Discussion des résultats

➔ *Classification arbre de décision :*

Notre conception de l'arbre décisionnel est réussie car nous obtenons de très bons résultats, surtout compte tenu de la grande similarité avec les résultats de la classe de scikit-learn. Point de vue améliorations, nous avons abordé les possibilités d'optimisation des calculs de seuil ou du processus d'élagage. Nous avons également fait face à une réussite due au hasard, puisque notre modèle a mieux réussi que celui de scikit-learn sur le dataset complexe des ormeaux.

➔ *Classification réseaux de neurones artificiels :*

Notre conception du RN est très bonne car nous obtenons de très bons résultats pour les premiers datasets. Cela se justifie par les scores F1 très élevés, et la proximité des métriques avec la classe de scikit-learn. Nous devons toutefois améliorer la capacité de notre modèle en lui donnant la possibilité d'alterner entre différents discriminants ou différentes fonctions d'erreur, ceci dans le but de le rendre plus polyvalent, et par extension plus performant, lorsque le dataset est compliqué.

IV. Comparaison entre les algorithmes expérimentés dans le cours

Modèle		Knn	NB	DT	RN
Exactitude sur l'ensemble de tests	Iris	0.9667	0.9833	0.97	0.95
	WQ	0.7806	0.8037	0.83	0.82
	Abalones	0.8061	0.5739	0.81	0.44
Temps de prédiction d'un seul exemple (secondes)	Iris	0.0000	0.0000	0.0000	0.0000
	WQ	0.0000	0.0000	0.0000	0.0000
	Abalones	0.0000	0.0000	0.0000	0.0000
Temps d'apprentissage du modèle (secondes)	Iris	0.00	0.01	0.08	0.09
	WQ	0.00	0 .86	3.45	0.33
	Abalones	0.00	2.01	7.39	0.89

De manière générale, point de vue exactitude, le meilleur modèle est l'arbre décisionnel suivi du Knn. Toutefois, c'est le bayésien naïf qui semble avoir le mieux performé pour la dataset iris. Le réseau de neurones a le moins bien performé sauf pour le dataset de la qualité des vins. Il a particulièrement moins réussi le dataset des ormeaux. Nous expliquons cette moins bonne performance par le fait que nous regardons seulement l'exactitude, et que les données ne sont pas en suffisamment grand nombre pour que le RN excelle. L'arbre décisionnel est quant à lui sans surprise excellent pour les données en faible quantité, de par sa méthode de sélection des meilleurs attributs. Cette méthode n'est pas viable sur un ensemble massif de données, car trop lourd en calcul, mais se prête bien à cette application.

Les modèles ont tous un excellent temps de prédiction de chaque nouvel exemple, donc pas de hiérarchisation possible sur ce critère. Les valeurs ne sont pas exactement de 0, mais tellement petites que l'approximation laisse apparaître uniquement des 0.

Pour les temps d'apprentissage des modèles, le k voisins plus proches ressort comme étant le plus rapide. L'arbre décisionnel étant le plus lent, alors que les données sont en petite quantité, cet aspect vient renforcer son impraticabilité en présence de données massives.

V. Conclusion

Cette conclusion vient boucler les travaux pratiques 3 et 4. Les enseignements pratiques et théoriques que nous tirons sont multiples et très pertinents.

Du point de vue pratique, nous avons vu quatre approches différentes de classifications pour trois types de corpus différents. Cela nous a permis d'enrichir notre culture sur les modèles d'apprentissage existants, comment ils fonctionnent, comment les utiliser, comment les évaluer. Nous sommes trois étudiants en maîtrise IA, et ce bagage de connaissances nous sera assurément utile dans notre parcours scolaire ou professionnel; nous avons développé des bases solides pour des tâches de classification du futur.

Du point de vue théorique, nous avons appris à réfléchir sur les optimisations possibles de nos modèles, leurs points forts ainsi que leurs points faibles, à remettre en questions nos procédés pour chercher à progresser. Par exemple, nous avons rencontré l'importance d'adapter un modèle au corpus de données; si ce dernier est déséquilibré, il faut agir en conséquence. Nous avons vu des modèles qui optimisent leur phase d'entraînements (Knn), et d'autres qui favorisent leur phase de tests (Naïf bayésien). Nous avons des modèles qui excellent en présence de faibles données (arbre décisionnel) et d'autres qui sont meilleurs pour des données massives multidimensionnelles (réseau de neurones). Nous avons donc été sensibilisé à bien comprendre l'application avec laquelle nous travaillons, les données à disposition, afin de réfléchir et identifier le meilleur modèle avec sa meilleure structure pour accomplir la tâche de classification.

Notre équipe a donc bénéficié de ces deux travaux d'équipes, avec chaque étudiant ayant une participation active et efficace, puisque nous sommes persuadés que nous sommes désormais bien plus compétents dans le domaine d'apprentissage profond.

VI. Annexe

(les matrices de confusion n'ont pu être exportées)

(nos Knn et NB ont deux tableaux car on évaluait la phase d'entraînement et la phase de test alors que pour DT et MLP on évalue simplement la phase de test)

Knn en train (notre modèle)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.9778	0.8826	0.8587
Matrice de confusion				
Précision	Classe 0	1	0.9	0.82
	Classe 1	0.967	0.84	0.87
	Classe 2	0.967	NA	0.77
Rappel	Classe 0	1	0.9	0.67
	Classe 1	0.967	0.84	0.97
	Classe 2	0.968	NA	0.31
F1-score	Classe 0	1	0.9	0.74
	Classe 1	0.967	0.84	0.91
	Classe 2	0.968	NA	0.44

Knn en train (notre modèle)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.9667	0.7806	0.8061
Matrice de confusion				
Précision	Classe 0	1	0.79	0.66
	Classe 1	1	0.76	0.83
	Classe 2	0.9	NA	0.57
Rappel	Classe 0	1	0.84	0.54
	Classe 1	0.9	0.69	0.93
	Classe 2	1	NA	0.22
F1-score	Classe 0	1	0.82	0.60
	Classe 1	0.947	0.72	0.88
	Classe 2	0.95	NA	0.32

NaiveBayes en Train (notre modèle)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.9444	0.7869	0.5804
Matrice de confusion				
Précision	Classe 0	1	0.875	0.44
	Classe 1	0.90	0.68	0.87
	Classe 2	0.93	NA	0.20
Rappel	Classe 0	1	0.77	0.92
	Classe 1	0.93	0.82	0.54
	Classe 2	0.90	NA	0.52
F1-score	Classe 0	1	0.82	0.60
	Classe 1	0.918	0.74	0.67
	Classe 2	0.918	NA	0.29

NaiveBayes en Test (notre modèle)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.9833	0.8037	0.5739
Matrice de confusion				
Précision	Classe 0	1	0.86	0.43
	Classe 1	1	0.74	0.86
	Classe 2	0.95	NA	0.22
Rappel	Classe 0	1	0.79	0.91
	Classe 1	0.95	0.82	0.53
	Classe 2	1	NA	0.52
F1-score	Classe 0	1	0.82	0.58
	Classe 1	0.974	0.78	0.66
	Classe 2	0.974	NA	0.30

Knn (scikit-learn)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.95	0.7703	0.8043
Matrice de confusion				
Précision	Classe 0	1	0.79	0.65
	Classe 1	1	0.75	0.84
	Classe 2	0.86	NA	0.56
Rappel	Classe 0	1	0.83	0.53
	Classe 1	0.85	0.68	0.93
	Classe 2	1	NA	0.28
F1-score	Classe 0	1	0.81	0.58
	Classe 1	0.92	0.71	0.88
	Classe 2	0.93	NA	0.37

NB (scikit-learn)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.95	0.7703	0.8043
Matrice de confusion				
Précision	Classe 0	1	0.86	0.65
	Classe 1	1	0.74	0.84
	Classe 2	0.95	NA	0.56
Rappel	Classe 0	1	0.80	0.53
	Classe 1	0.95	0.81	0.93
	Classe 2	1	NA	0.28
F1-score	Classe 0	1	0.83	0.58
	Classe 1	0.97	0.78	0.88
	Classe 2	0.97	NA	0.37

DecisionTree (notre modèle)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.83	0.81
Matrice de confusion				
Précision	Classe 0	1	0.835	0.7373
	Classe 1	1	0.82	0.8181
	Classe 2	0.90	NA	0.6481
Rappel	Classe 0	1	0.88	0.392
	Classe 1	0.9	0.76	0.966
	Classe 2	1	NA	0.176
F1-score	Classe 0	1	0.857	0.512
	Classe 1	0.947	0.788	0.886
	Classe 2	0.95	NA	0.277

Réseau de neurones (notre modèle)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.95	0.82	0.44
Matrice de confusion				
Précision	Classe 0	1	0.887	0.44
	Classe 1	1	0.756	0.87
	Classe 2	0.86	NA	0.20
Rappel	Classe 0	1	0.80	0.92
	Classe 1	0.85	0.858	0.54
	Classe 2	1	NA	0.52
F1-score	Classe 0	1	0.84	0.60
	Classe 1	0.919	0.80	0.67
	Classe 2	0.927	NA	0.29

DT (scikit-learn)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.85	0.8
Matrice de confusion				
Précision	Classe 0	1	0.85	0.70
	Classe 1	1	0.84	0.82
	Classe 2	0.9	NA	0.55
Rappel	Classe 0	1	0.89	0.46
	Classe 1	0.9	0.79	0.95
	Classe 2	1	NA	0.15
F1-score	Classe 0	1	0.87	0.55
	Classe 1	0.95	0.81	0.88
	Classe 2	0.95	NA	0.23

MLP (scikit-learn)

Dataset		Iris	Wine Quality	Abalones
Exactitude		0.97	0.82	0.83
Matrice de confusion				
Précision	Classe 0	1	0.82	0.71
	Classe 1	1	0.84	0.86
	Classe 2	0.9	NA	0.61
Rappel	Classe 0	1	0.90	0.57
	Classe 1	0.9	0.72	0.93
	Classe 2	1	NA	0.40
F1-score	Classe 0	1	0.86	0.63
	Classe 1	0.95	0.77	0.89
	Classe 2	0.95	NA	0.48