TP apprentissage statistique

AKKOUH Maryam

TP 2 : ARBRES

1. CLASSIFICATION AVEC LES ARBRES

Dans ce TP nous allons explorer les méthodes numériques pour la classification avec les arbres ainsi que les méthodes de choix de paramètres en utilisant la sélection de modèle, notamment avec la validation croisée.

Nous allons pour cela considérer l'arbre de décision binaire *CART*. Il s'agit d'un algorithme de moyennage local par partition, dont la partition est construite par divisions successives au moyen d'hyperplans orthogonaux aux axes de $\mathbb{R}^p$. L'ensemble $\mathbb{R}^p$ constitue le noeud racine. Chaque division définit deux noeuds, les noeuds fils à gauche et à droite. Chacun sera : soit terminal, soit interne. Par le choix d'une variable explicative $X^{(j)}$ $(j = 1, ..., p)$ et d'une valeur seuil pour cette variable. Il faut, pour effectuer ce choix, maximiser une certaine fonction d'homogénéité H qu'on aura défini au préalable.

Dans le cadre de la régression, dans le but de mesurer l'homogénéité d'un modèle nous pouvons utiliser la variance. En effet, il s'agit d'un très bon indicateur, si la variance est élevée au sein des données cela veut dire que les données sont très éloignées les unes des autres. L'ensemble des données sera donc très hétérogène. A contrario, si la variance est faible, cela indique une homogénéité de l'ensemble des données.

En discrimination binaire, nous allons plutôt considérer des fonctions comme l'indice de Gini par exemple.

Exploration du package *tree*

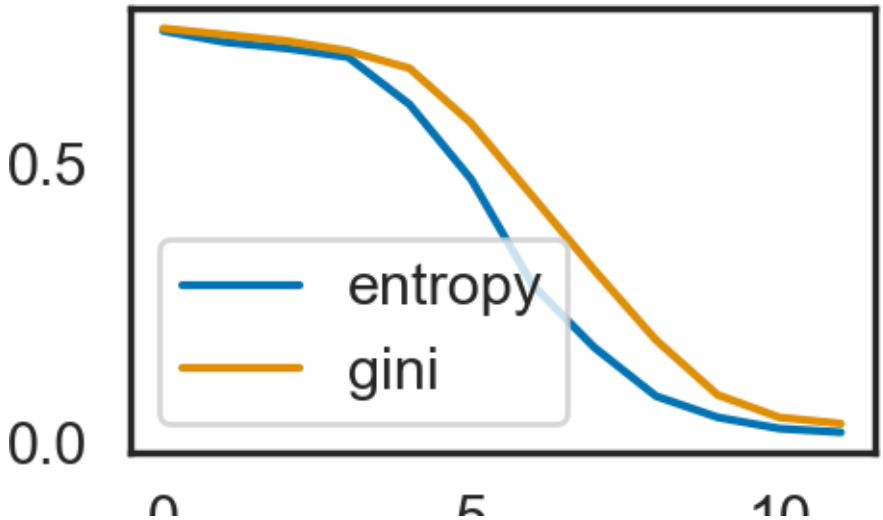
Avec *scikit-learn* nous pouvons construire des arbres de décision grâce au package *tree*.

En utilisant la fonction `rand_checkers`, on construit un échantillon de taille $n = 456$ en faisant attention à bien équilibrer les classes.

```
data = rand_checkers(n1=114,n2=114,n3=114,n4=114,sigma=0.1)
n_samples = len(data)
X_train = data[:, :2]
Y_train = data[:, 2].astype(int)
```

Ensuite, on crée deux courbes donnant le pourcentages d'erreurs commises en fonction de la profondeur maximale de l'arbre (en laissant les autres paramètres par défaut).

On obtient alors le graphique suivant :



On voit sur le graphique que les erreurs diminuent de manière similaire pour les deux critères pour s'approcher notablement de 0 lorsque la profondeur maximale de l'arbre atteint 10. Cependant, en utilisant le critère de Gini, l'erreur commise est légèrement plus importante que celle dans le cas de l'utilisation de l'entropie.

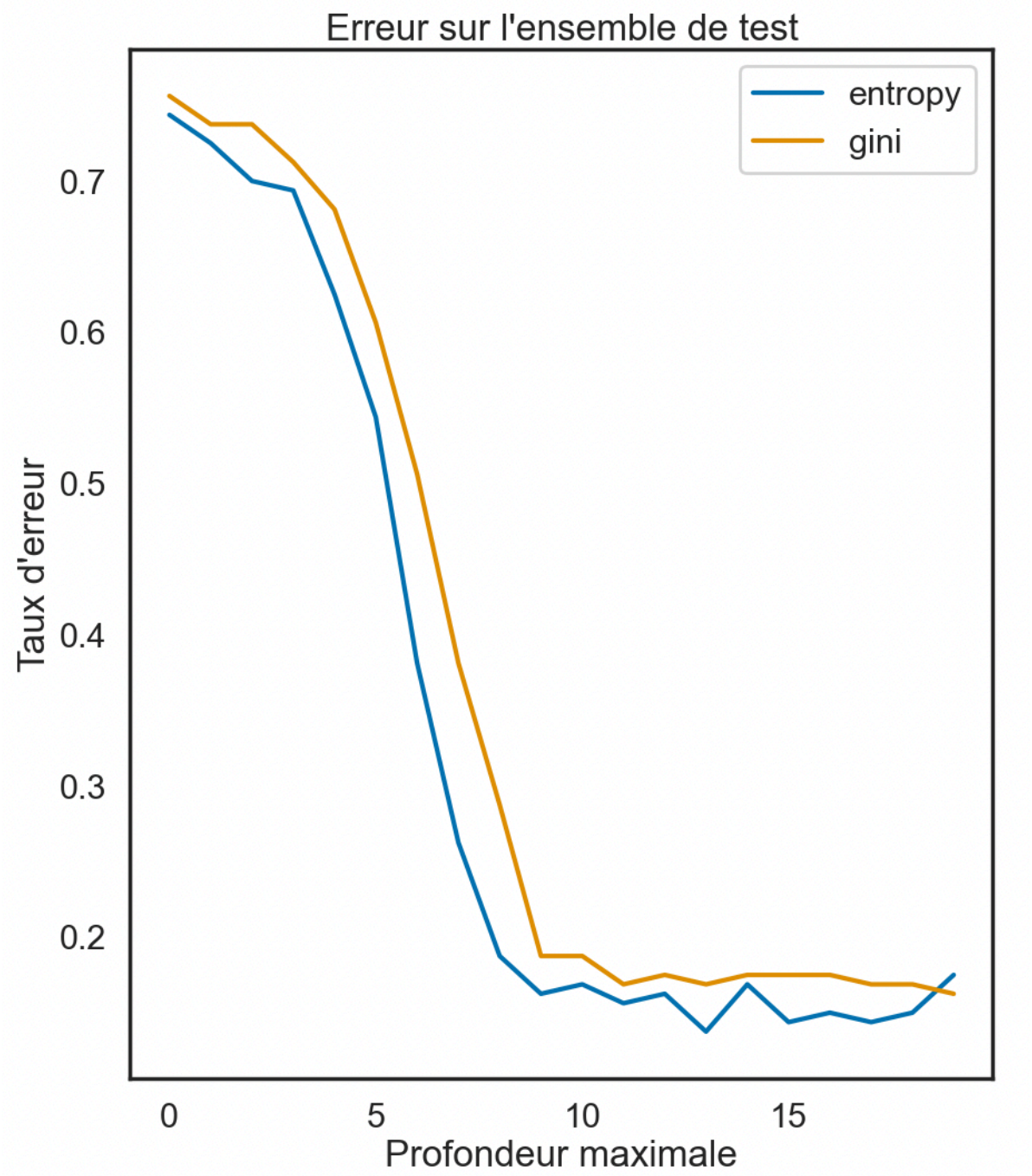
Il faut cependant rester vigilant face à ces erreurs quasi nulles à partir d'une profondeur égale à 10 et très faibles dès lors que la profondeur maximale atteint 8. En effet, l'erreur est ici calculée à partir des données d'apprentissage. Ces données ont dont servi pour ajuster le modèle et pour évaluer son efficacité, cela peut être dangereux car c'est un cas typique de sur-apprentissage. Il faudra alors trouver un moyen d'ajuster ce modèle tout en gardant un biais assez faible et trouver ainsi un compromis entre le sur-apprentissage et le sous-apprentissage du modèle.

Cependant, on remarque sur le graphique que les deux courbes sont très proches, même si nous ne pouvons baser toutes nos conclusions sur ce graphique pour les raisons citées précédemment, nous pouvons conclure quant à la pertinence de l'utilisation d'un classifieur dans notre étude.

En effet, les erreurs commises sont très faibles et similaires pour les deux critères utilisés pour l'homogénéité du modèle.

Nous pouvons regarder plus précisément la classification obtenue en utilisant la profondeur qui minimise le pourcentage d'erreurs obtenues avec l'entropie. On utilise pour cela les fonctions `plot_2d`et `frontière`.

On obtient alors le graphique suivant :

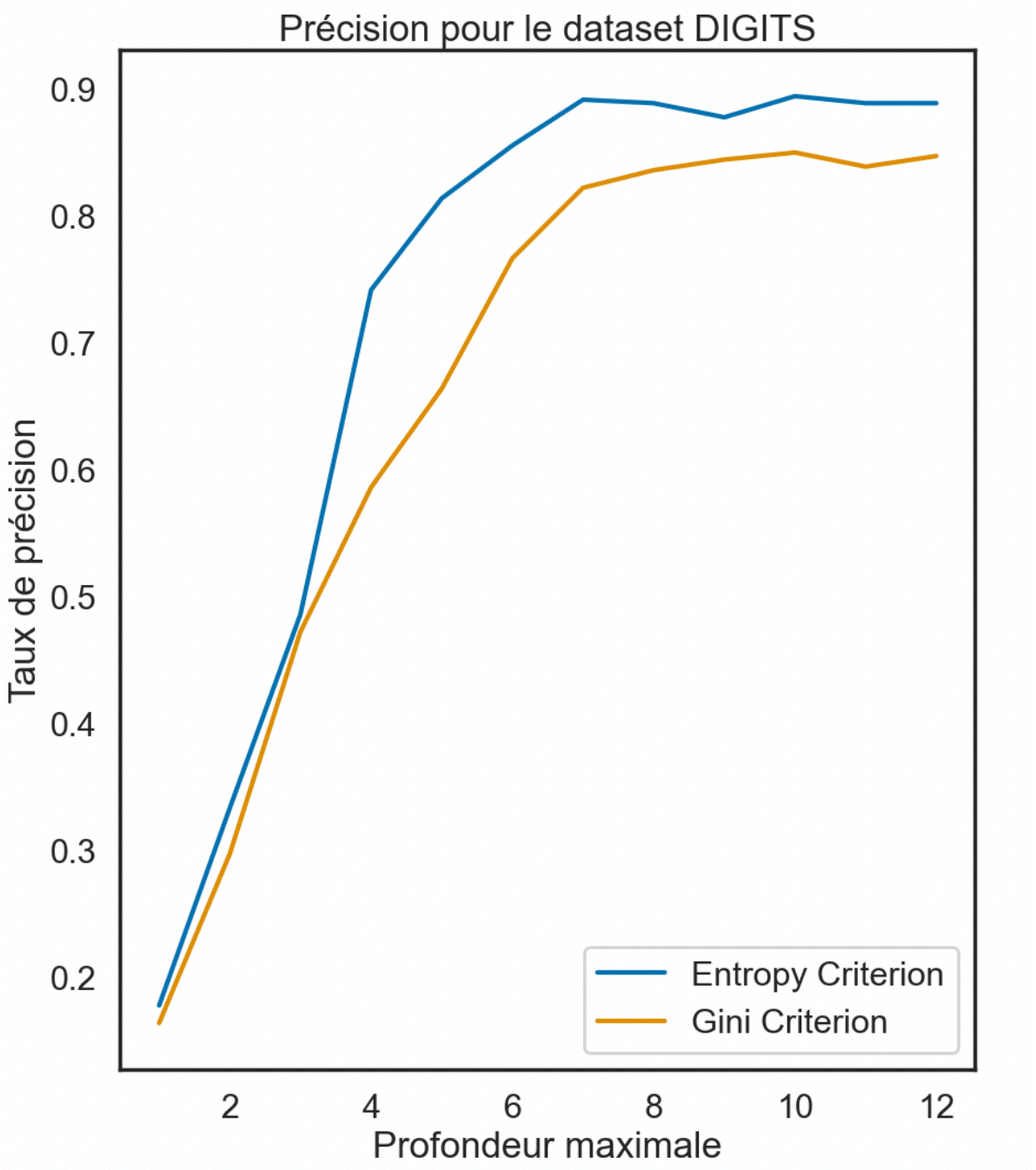


De manière similaire à ce qui a été observé précédemment, les erreurs associées aux deux critères évoluent de manière cohérente. Il est notable qu'avec un ensemble de données de taille $n = 160$ (qui est inférieur à $n = 456$), les erreurs diminuent plus rapidement à mesure que la profondeur maximale de l'arbre de décision augmente. Cependant, dès que la profondeur max-

imale atteint 9, il devient évident que poursuivre l'augmentation de la profondeur n'apportera que peu ou pas d'amélioration significative en termes de réduction de l'erreur, car l'erreur demeure pratiquement constante à ce stade.

On refait les étapes précédentes pour le dataset `DIGITS`. On commence par diviser ces données en deux groupes avec la fonction `train_test_split`. Cela nous donnera les données d'apprentissage et les données de test.

On obtient le graphique suivant :



On remarque ici que pour les données d'entraînement, le graphe renseigne une erreur quasi nulle au bout d'une profondeur maximale égale à 8. Ceci est cohérent avec le travail d'avant.

En pratique, nous ne disposons pas d'un ensemble de test. Il faut diviser nos données en deux

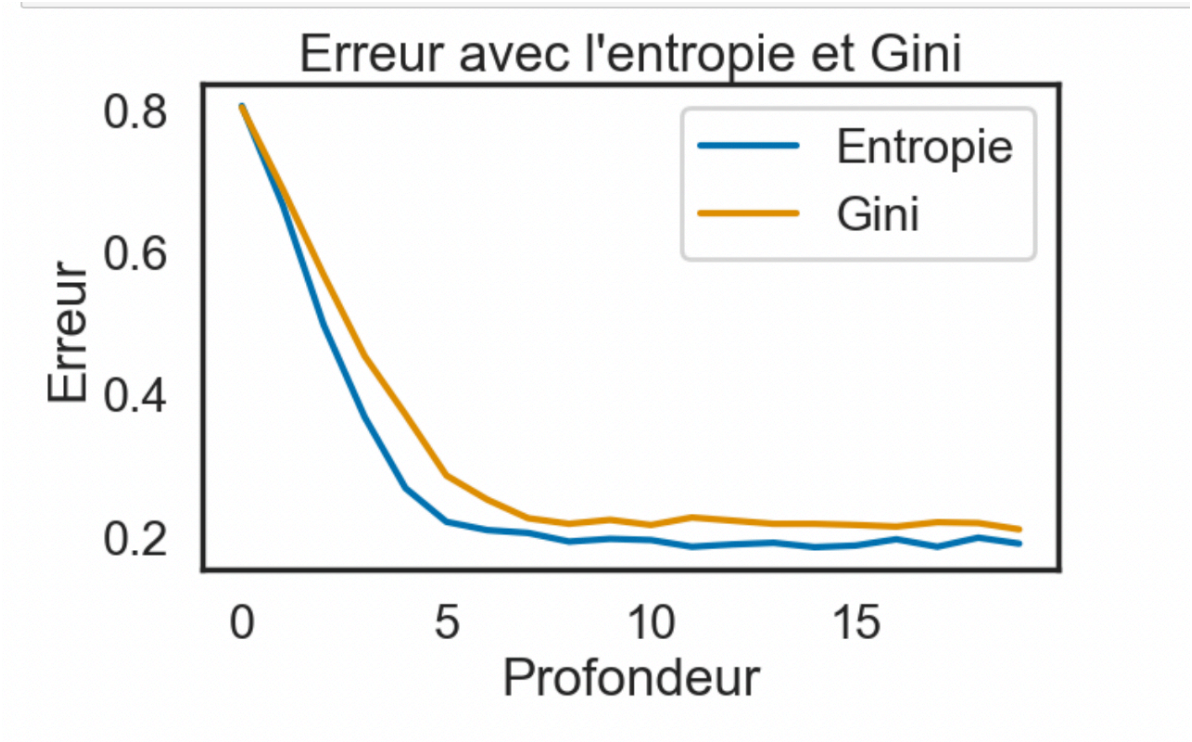
parties : un ensemble de test et un ensemble d'apprentissage. Il faut réserver un bon taux de données pour l'apprentissage car le modèle sera d'autant plus précis qu'il a de données pour apprendre. Pour sélectionner un modèle ou un paramètre tout en considérant le plus grand nombre d'exemples possibles pour l'apprentissage, on utilise une sélection par validation croisée.

2. METHODES DE CHOIX DE PARAMETRES - SELECTION DE MODELE

VALIDATION CROISEE

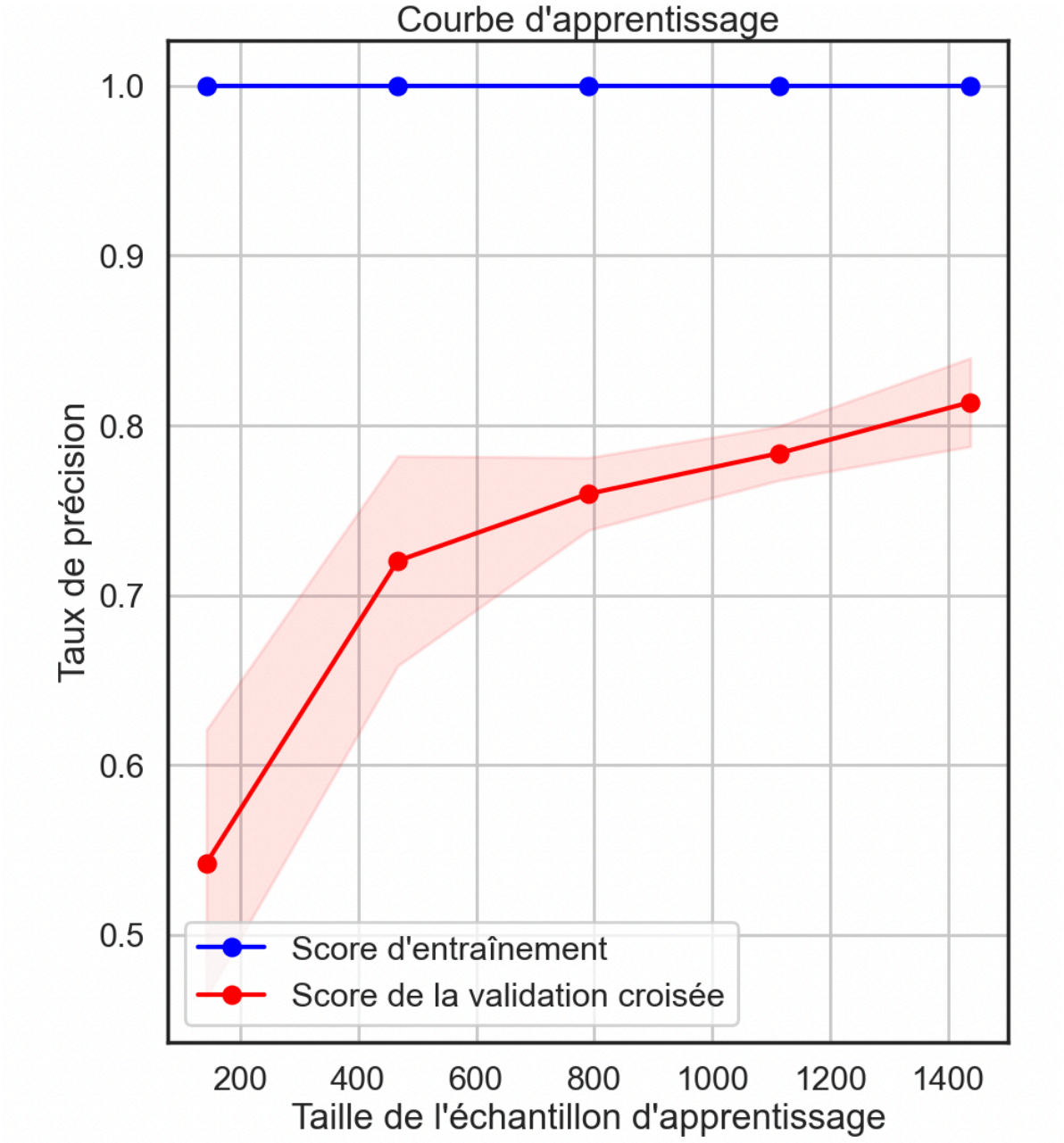
On utilise la fonction **sklearn.cross_validation.cross_val_score**. Cette fonction sert à évaluer la performance d'un modèle d'arbre de décision à différentes profondeurs en utilisant la validation croisée. On définit pour cela la plage de valeurs de profondeur. Elle effectue une boucle sur les différentes profondeurs et calcule les précisions moyennes. Ensuite **cross_val_score** est utilisée pour calculer les scores de validation croisée avec 5 plis. Elle identifie ainsi la meilleure profondeur en trouvant celle qui donne la précision moyenne la plus élevée (**best_depth**). On affiche alors la meilleure profondeur et la meilleure précision moyenne.

Après exécution du code nous obtenons que la profondeur optimale est 15 :



En effet, selon les deux critères (entropie et Gini), en affichant le graphique de l'erreur en fonction de la profondeur, nous voyons que 15 semble être un choix convenable. Cependant, on voit également sur le graphique qu'à partir d'une profondeur de 5, selon les deux critères, l'erreur est proche de 0.

On veut à présent afficher la courbe d'apprentissage pour les arbres de décision sur le même jeu de données. Elle nous permettra d'évaluer la performance de l'arbre en fonction de la taille de l'ensemble de données d'apprentissage. Elle donne une estimation de la précision du modèle. On aura alors la courbe de test (learning curve) qui utilise l'ensemble d'entraînement pour construire le modèle puis évalue ce modèle en utilisant ce même ensemble d'entraînement. On aura également la courbe de la validation croisée qui utilise l'ensemble de validation pour évaluer le modèle.



La courbe bleue représente le score des données d'entraînement. Cette courbe est celle de l'équation $y = 1$. En effet, le modèle a bien appris sur les données et l'erreur commise est nulle.

La courbe rouge correspond au score des données de la validation croisée. Généralement, la performance du modèle diminue à mesure que la taille de l'ensemble de validation augmente. Ici, ce n'est pas le cas pour la validation croisée. En effet, la performance augmente lorsque

la taille de l'ensemble de validation augmente. Cela peut signifier plusieurs choses, comme par exemple un sous-ajustement du modèle. En effet, le modèle n'a pas encore bien appris à partir des données d'entraînement. Ce biais élevé du modèle peut être réduit en augmentant la complexité du modèle ou en fournissant davantage de données d'entraînement afin d'améliorer sa performance.

3. CONCLUSION

Pour conclure, nous avons constaté que les deux critères de partitionnement ont des performances similaires, avec des taux d'erreur qui diminuent à mesure que la profondeur maximale de l'arbre augmente. Cependant, nous avons également noté que des taux d'erreur très faibles peuvent résulter d'un sur-apprentissage du modèle aux données d'entraînement, ce qui nécessite une attention particulière pour éviter le sur-ajustement.

La validation croisée nous a permis de sélectionner la meilleure profondeur de l'arbre de décision pour notre modèle, en minimisant les taux d'erreur moyens sur différents plis de validation. Cette approche nous a aidés à déterminer que la profondeur optimale était d'environ 15 pour notre ensemble de données.

Enfin, nous avons tracé des courbes d'apprentissage pour évaluer la performance de notre modèle en fonction de la taille de l'ensemble de données d'apprentissage. Nous avons observé que la performance du modèle n'augmentait pas nécessairement avec l'augmentation de la taille de l'ensemble de validation, ce qui peut indiquer un sous-ajustement du modèle.

En résumé, ce TP nous a permis de comprendre les bases des arbres de décision, les critères de partitionnement, la sélection de modèle par validation croisée, et l'importance de surveiller le sur-ajustement lors de la construction de modèles d'arbres de décision. Ces concepts sont essentiels pour le développement de modèles d'apprentissage automatique efficaces et précis.