L2 MI Info232 - Mini Projet

Challenge *Hadaca* – Groupe *Cure*

Noms des membres : BENKHELIFA Mohamed groupe 3;
CARO Hugo groupe 4;
PÈRE Savanah groupe 3;
BEHIDJ Ramzi groupe 1;
VU Phuong Nam groupe 4.

Lien du Challenge : https://codalab.lri.fr/competitions/333
Lien du GitHub : https://github.com/Cure0/Cure0
Numéro de soumissions codalab : 19
Lien vidéo Youtube : https://www.youtube.com/watch?v=zxb3VtLXDGM
Lien des diapositives :https://github.com/Cure0/Cure0/blob/master/Projet-HADACA.pptx

Introduction :

De nos jours, le cancer fait des milliers, voir des millions, de morts chaque année et rien qu'en France il tue plus de 150 000 personnes par an. De plus, tous les ans on recense 400 000 nouveaux cas atteints de cancer juste en France. Il est un des principaux responsables de la mortalité humaine. C'est donc une priorité médicale de réussir à réduire ses effets et , rêvons un peu, d'un jour le guérir. Une des étapes pour guérir le cancer, est de mieux réussir à l'identifier et de savoir à quel stade en est le patient pour savoir quel est le traitement le plus adapté. L'intérêt que nous portons à cette maladie nous a poussés à choisir le challenge "HADACA" parmi tous les projets proposés.
Durant ce challenge, nous avions pour objectif d'essayer de déterminer le stade des cancers des patients grâce aux données obtenues par prélèvement ADN, avec l'aide du machine learning. Dans notre cas, nous avons essayé de trouver le stade des cancers des 5000 patients, et nous avons réparti ces mêmes patients en 10 classes (une pour chaque stade). Ces données présentaient 5000 caractéristiques.

Pour réaliser ce projet, nous avions établi un plan de travail en répartissant les tâches en 3 parties majeures :

-le Pré-processing, c'est-à-dire la transformation de données brutes en données utilisables par le classifieur;

-le Classifieur, qui va analyser les données d'entraînement ainsi que les résultats attendus sur ces données d'entraînement pour se calibrer;

- la Visualisation, qui a permis d'afficher les résultats de manière compréhensible et pertinente.

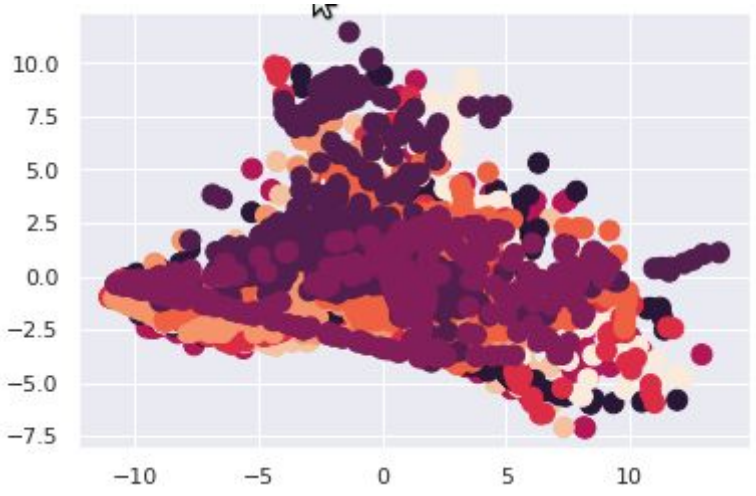
Pour évaluer les performances de notre modèle, nous avons eu besoin d'une métrique (l'indice de concordance). Cet indice a permis de valider la capacité prédictive du modèle.

Pré-processing :

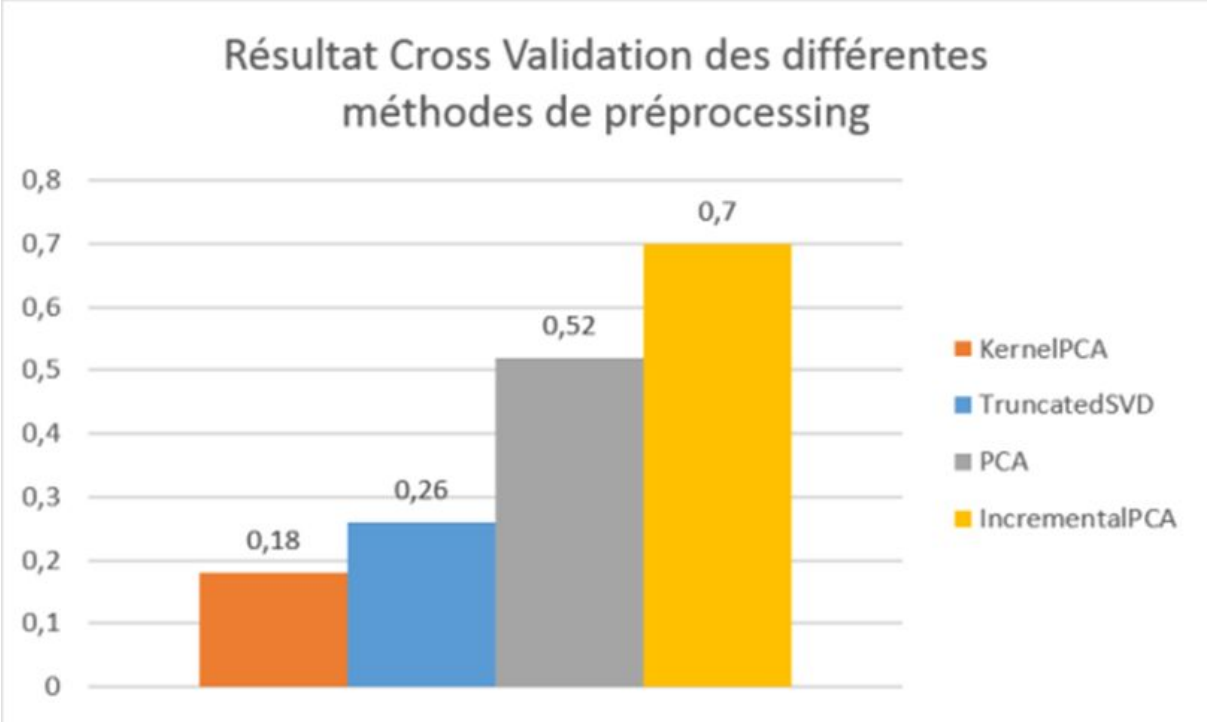
Au niveau du pré-processing, nous avons effectué des recherches sur scikit-learn afin de récupérer les différentes fonctions qui ont permises de factoriser les matrices. Afin de déterminer la meilleure méthode, nous avons testé et comparé les différentes fonctions et avons retenu celle du PCA. La raison pour laquelle nous avons retenue le PCA est tout simplement le score obtenue était meilleur que les autres lorsque nous utilisons la méthode du PCA

La méthode PCA permet une réduction du nombres de features, de dimensions, ce qui va permettre l'amélioration du score et une meilleure visualisation, afin de comprendre et d'interpréter les différents résultats obtenus. Cette méthode permet aussi de rassembler les données qui se ressemblent, ainsi que la construction de features, en faisant des opérations arithmétiques sur les données déjà existantes. PCA cherche les directions où les données sont les plus allongés, celles avec la plus grandes variances.

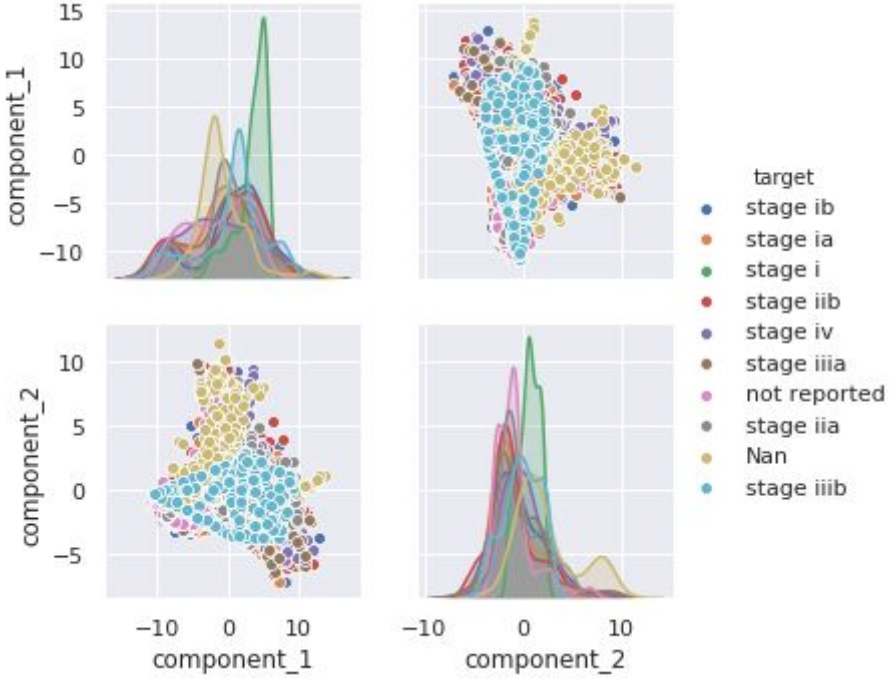
Au lieu de choisir arbitrairement un nombre de dimensions auxquelles on va réduire, on veut de préférence choisir un nombre de dimension représentant une contribution suffisamment importante à la variance. Pour cela on peut fixer l'attribut *n_components* avec une valeur comprise entre 0.0 et 1.0. Cependant lorsque nous appliquons cette méthode pour une précision de 95%, le nombre de features retenu est de 209, cependant le score est bien moins bon que lorsque nous sélectionnons seulement 10 features. La raison probable à un tel résultat et que les données qui nous ont été fournis ont déjà été trier auparavant. Cette méthode ne fonctionne donc pas dans notre situation. Nous nous sommes alors contenté de choisir un *n_components* compris entre 10 et 209.



Après pré-processing (figure 1)



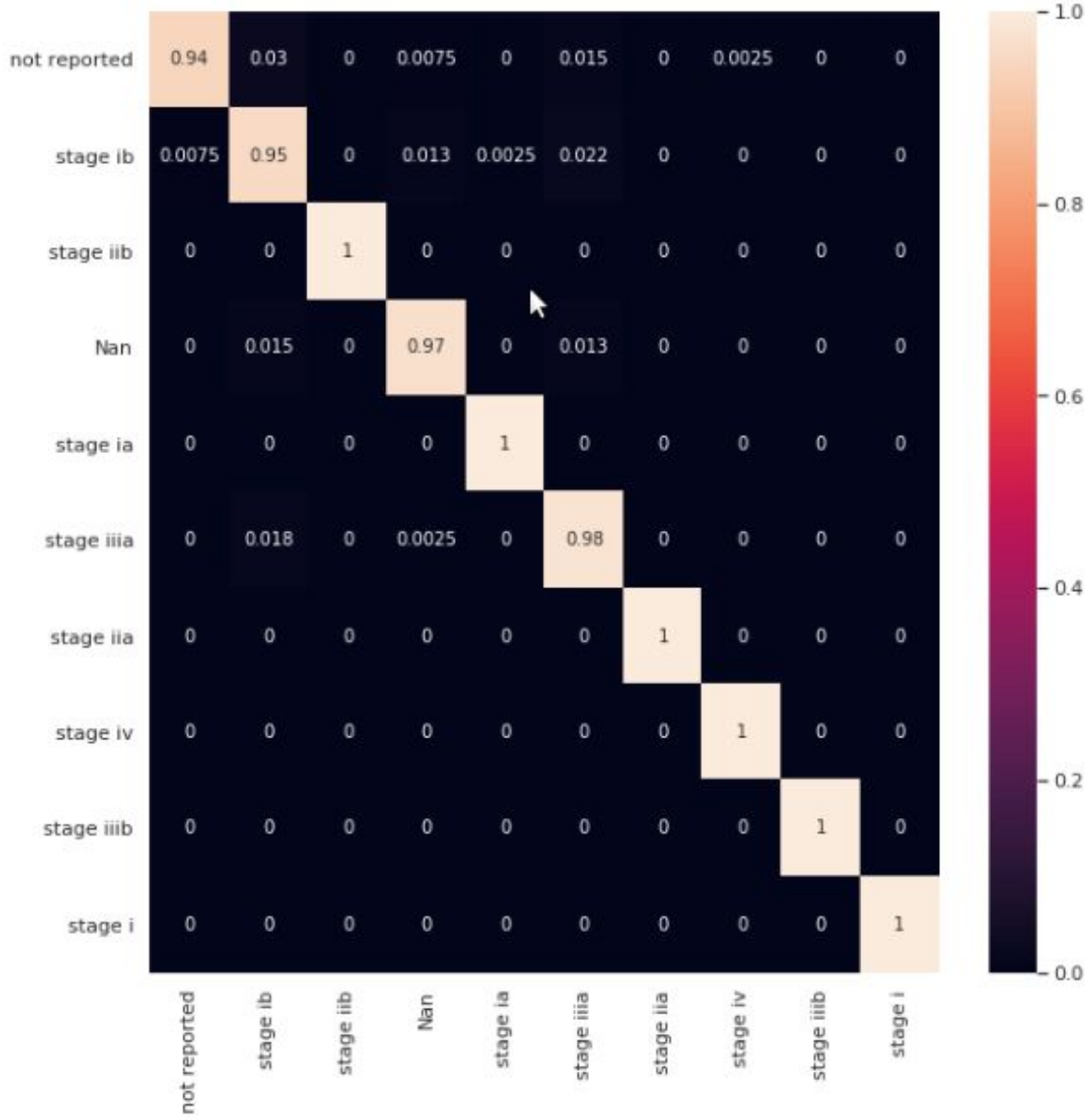
Comparaison du score en C.V. des différentes méthodes de préprocessing (figure 2)



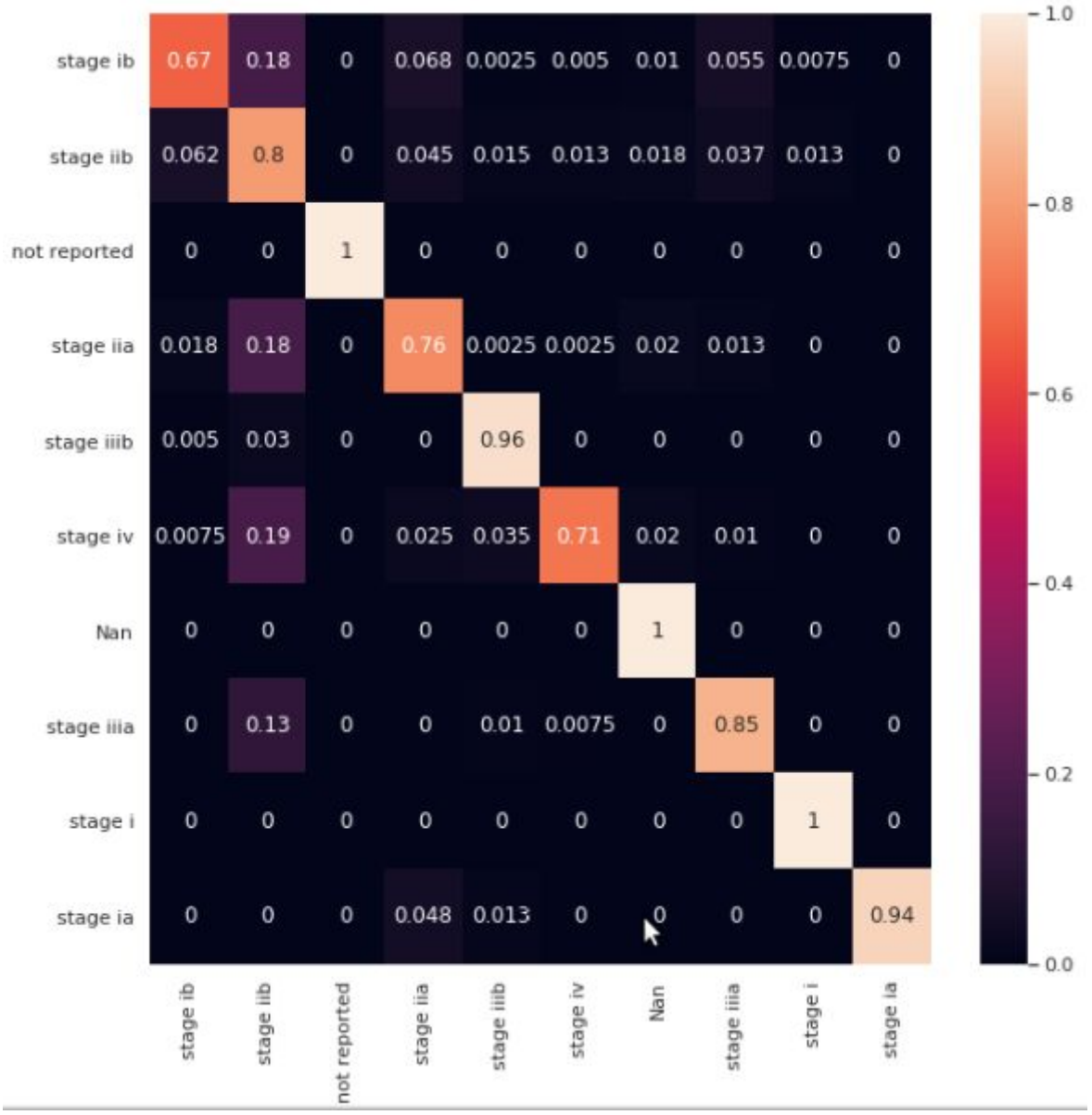
Différentes façons d'appréhender les données (figure 3)

Classifieur :

Puisque nos données ont plusieurs labels, nous utilisons un classifieurs multi-classes, le modèle SVC. Les SVC peuvent être utilisés pour résoudre des problèmes de discrimination, c'est-à-dire décider à quelle classe appartient un échantillon, ou de régression, c'est-à-dire prédire la valeur numérique d'une variable. La résolution de ces deux problèmes passe par la construction d'une fonction qui à un vecteur d'entrée fait correspondre une sortie. Dans le cas du multi classes Plusieurs méthodes ont été proposées pour étendre le schéma ci-dessus au cas où plus de deux classes sont à séparer. Ces schémas sont applicables à tout classifieur binaire, et ne sont donc pas spécifiques aux SVC. Les deux plus connues sont appelées *one versus all* et *one versus one*. Formellement, les échantillons d'apprentissage et de test peuvent ici être classés dans 10 classes. Après un test rigoureux des deux méthodes, on a remarqué qu'en utilisant la méthode One versus One, la précision du classifieur était meilleur, cette méthode consiste à construire M/(M-1)/2 classifieurs binaires ou M correspond au nombre de classes (10 dans notre cas) en confrontant chacune des M classes. En phase de test, l'échantillon à classer est analysé par chaque classifieur et un vote majoritaire permet de déterminer sa classe. Si l'on note Xt l'échantillon à classer et Hi le classifieur SVC séparant la classe Ci et la classe Cj et renvoyant le label attribué à l'échantillon à classer. C'est la classe qui sera le plus souvent attribuée à Xt quand il aura été analysé par tous les Hi. Pour déterminer le degré de précision de notre classifieur, nous avons décidé de mettre en place une matrice de confusion (figure 4) en prenant en compte la PCA. On a remarqué après analyse de la matrice que la précision est maximale entre 0.97 et 1 ce qui nous a fait penser a un sur-apprentissage de la part de notre classifieur car la précision et un taux de succès trop élevé pendant l'entraînement et cela se fera au détriment des performances réelles. Cela nous a conduit a retirer la PCA et d'établir une nouvelle matrice de confusion (figure 5) qui se montre plus concluante après son analyse car elle présente un bon taux de succès sans être trop élevé ni trop bas . Cela nous a donc permis de choisir un classifieur plus adapté sans PCA.



Matrice de confusion avec PCA (figure 4)



Matrice de confusion sans PCA (figure 5)

Visualisation :

Dans cette partie, nous avons fait en sorte de rendre les résultats accessibles et exploitables par tous. Pour cela, nous avons cherché sur différents supports (scikit-learn, Jupyter, …) quel serait le(s) meilleur(s) graphe(s) pour chaque partie du code (pré-processing et classifieur). Pour le pré-processing, nous avons choisis de montrer les données avant et après le passage du code sous la forme d'un nuage de points (figure 1) pour voir si les données avaient bien été triées ainsi qu'un graphique en bâtonnet (figure 2) afin de comparer les différentes méthodes de pre-processing et d'en sélectionner la meilleure (Cross-Validation). Pour le classifieur, nous avons décidé de mettre en place deux matrices de confusion avec et sans PCA (figures 4 & 5) pour voir quelle(s) classe(s) n'obéissent pas à notre prédiction. La mise en place d'une interface graphique adéquate et complète est essentielle au bon déroulement d'une présentation de projet qui d'une part garderai l'audience captivé et d'une autre permettra une meilleur compréhension de toutes les notions abordé qui ne sont peut être pas assimilé par tout le monde.

Conclusion :

Toute les choses mentionnées n'ont pas encore été incorporées ou réalisées dans ce rapport ou dans les codes soumis mais le seront dans la semaine normalement, de plus ce rapport sera complété de même dans la semaine, merci de votre compréhension.
Pour conclure, ce projet fut intéressant car cela nous a permis de rechercher par nous même des codes ou des inspirations et nous allons continuer de rechercher de meilleurs codes pour avoir des résultats plus pertinents en classifieur.

Références :

1/Cours de L2 « Intoduction à la vie Artificielle ». Aurélien Decelle. 2018

2/README.ipynb fourni avec le starting kit.

3/Cours de L2 « Mini-projet ». Isabelle Guyon. 2018

4/https://scikit-learn.org/stable/auto_examples/compose/plot_compare_reduction.html#sphx-glr-auto-examples-compose-plot-compare-reduction-py

5/https://scikit-learn.org/stable/modules/svm.html#svm-classification

6/model.py

7/Livre "Machine Learning avec Scikit-Learn" , Aurélien Géron.

8/https://fr.wikipedia.org/wiki/Machine_à_vecteurs_de_support

9/https://www.python-course.eu/principal_component_analysis.php

Partie Bonus :

Sur-apprentissage :

Un réseau de neurones apprend grâce à des exemples (jeux d'entraînements) qui lui sont soumis. Le but de cet apprentissage est de permettre au réseau de tirer de ces exemples des généralités et de pouvoir les appliquer à de nouvelles données par la suite.
En intelligence artificielle, on parle de surapprentissage ou de overfitting quand un modèle a trop appris les particularités de chacun des exemples fournis en exemple. Il présente alors un taux de succès très important sur les données d'entraînement (pouvant atteindre jusqu'à 100%), au détriment de ses performances générales réelles.

Cross-validation :

La validation croisée est, en apprentissage automatique, une méthode d'estimation de fiabilité d'un modèle fondé sur une technique échantillonnage, En fait, il y a au moins trois techniques de validation croisée : « testset validation » ou « holdout method », « k-fold cross-validation » et « leave-one-out cross validation »,
Supposons posséder un modèle statistique avec un ou plusieurs paramètres inconnus, et un ensemble de données d'apprentissage sur lequel on peut entraîner le modèle. Le processus d'apprentissage optimise les paramètres du modèle afin que celui-ci corresponde aux données le mieux possible. Si on prend ensuite un échantillon de validation indépendant issu de la même population d'entraînement, il s'avérera en général que le modèle ne réagit pas aussi bien à la validation que durant l'entraînement. La validation croisée est un moyen de prédire l'efficacité d'un modèle sur un ensemble de validation hypothétique lorsqu'un ensemble de validation indépendant et explicite n'est pas disponible.