

Projet Modèle de prévision

Pablo GUYEZ, Guillaume PARENT

April 2023



Contents

1	Introduction	3
1.1	Problématique	3
1.2	Présentation du dataset	3
2	Exploration des données	4
2.1	Valeurs manquantes	4
2.2	Visualisation des courbes	4
2.3	Corrélations	5
2.4	Premier modèle intuitif	6
3	Traitemet des données	7
3.1	Class balancing	7
3.2	Outliers	7
3.3	Smoothing	9
3.3.1	Lissage Exponentiel	9
3.3.2	Lissage Gaussien	10
3.4	Normalisation des données	11
4	Modélisation	11
4.1	Régression logistique	11
4.2	KNN	12
4.3	Decision tree	13
4.4	Approche Deep Learning	14
5	Résultats et conclusion	15

1 Introduction

1.1 Problématique

La recherche d'exoplanètes, c'est-à-dire de planètes en orbite autour d'étoiles situées en dehors de notre système solaire, est un enjeu majeur de la compréhension de l'espace. L'un des moyens de détecter la présence d'une exoplanète est d'observer l'étoile autour de laquelle elle orbite. Comme nous le savons, les planètes elles-mêmes n'émettent pas de lumière, mais les étoiles autour desquelles elles gravitent en émettent. Par conséquent, si nous observons une étoile pendant un certain temps, nous pouvons détecter un "affaiblissement" régulier du flux (intensité lumineuse), indiquant qu'un corps en orbite passe devant l'étoile. Cet affaiblissement régulier est appelé "transit".

Si un tel affaiblissement se produit, l'étoile en question est considérée comme un système "candidat" à la présence d'une exoplanète. Toutefois, des études complémentaires sont nécessaires pour confirmer que l'assombrissement est bien causé par une planète en orbite et non par un autre phénomène.

Le machine learning joue un rôle de plus en plus important dans la recherche d'exoplanètes. L'analyse des données de flux est l'un des moyens par lesquels l'apprentissage automatique peut contribuer à cet effort. Comme nous l'avons mentionné, l'affaiblissement du flux d'une étoile peut indiquer la présence d'une planète en orbite. Cependant, la détection de ces petites variations de flux peut s'avérer difficile, en particulier lorsqu'il s'agit de grandes quantités de données.

Des algorithmes d'apprentissage automatique peuvent être entraînés sur de vastes ensembles de données de mesures de flux afin d'identifier des schémas et des anomalies susceptibles d'indiquer le passage d'une exoplanète. Ces algorithmes, en plus de pouvoir analyser un grand nombre d'astres en très peu de temps, peuvent apprendre à distinguer les transits réels des faux positifs causés par d'autres facteurs, tels que le bruit de l'instrument ou la variabilité de la luminosité de l'étoile.

Notre but est donc de construire un algorithme permettant de détecter des étoiles candidates à la présence d'une exoplanète.

1.2 Présentation du dataset

Les données décrivent le changement de flux (intensité lumineuse) de plusieurs milliers d'étoiles. Chaque étoile porte une étiquette binaire de 0 ou 1. 0 indique qu'il est confirmé que l'étoile a au moins une exoplanète en orbite. certaines observations sont en fait des systèmes multi-planètes.

Trainset :

- 5087 observations
- 3198 colonnes
- La colonne 1 est le vecteur de label. Les colonnes 2 à 3198 sont les valeurs de flux dans le temps.
- 37 étoiles exoplanètes confirmées et 5050 étoiles non exoplanètes.

Jeu de tests :

- 570 lignes
- 3198 colonnes
- 5 étoiles exoplanètes confirmées et 565 étoiles non exoplanètes.

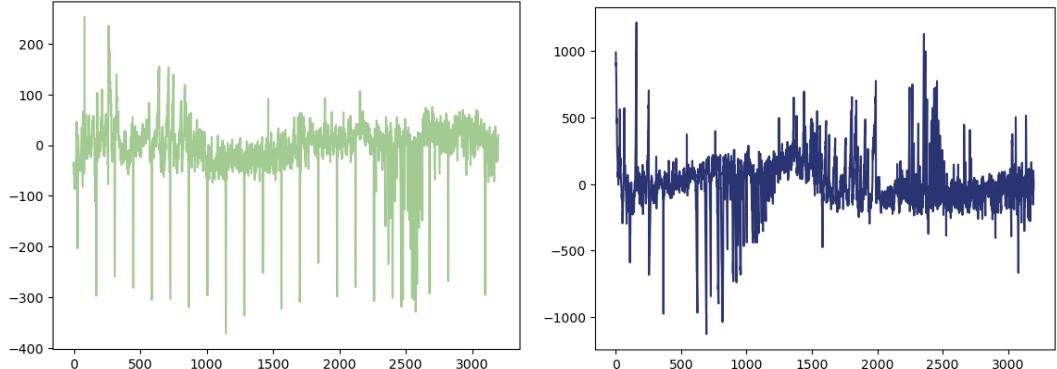
2 Exploration des données

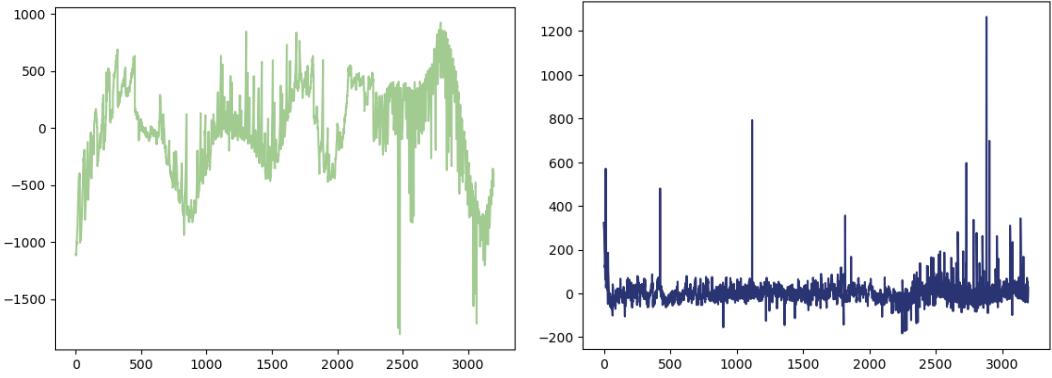
2.1 Valeurs manquantes

Le dataset ne contient pas de valeurs manquantes, aucun traitement n'est donc à effectuer

2.2 Visualisation des courbes

Nous allons visualiser différents plots de l'évolution du flux pour observer les éventuelles particularités d'un type d'étoile ou l'autre. En vert si c'est une étoile pouvant avoir une exoplanète en orbite, en bleu sinon :

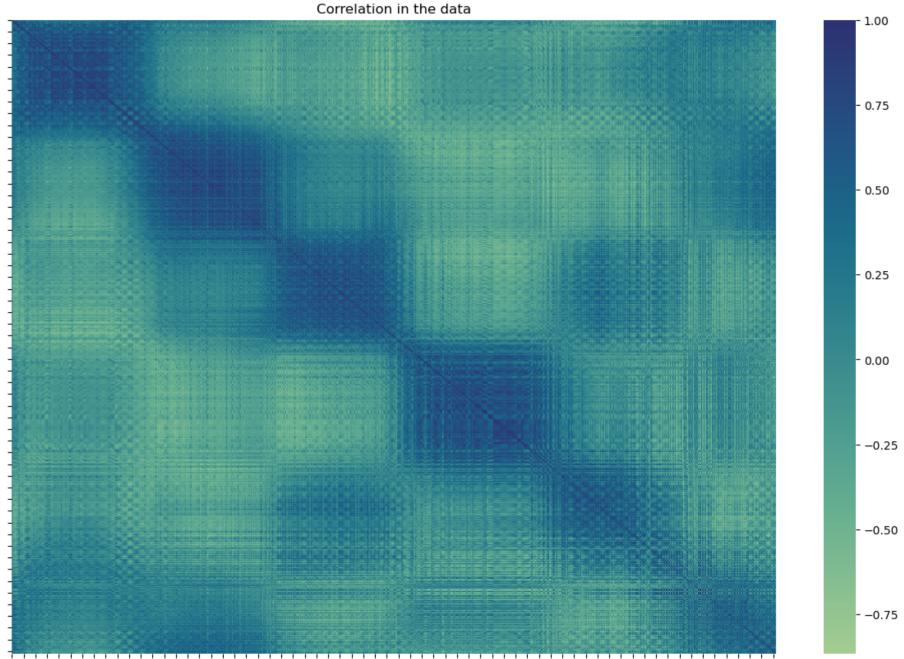




On remarque sur le graphique 1 (astre candidat) une chute brutale du flux à intervalle régulier, ce qui pourrait être un schéma révélateur d'une exoplanète. Cependant ces graphiques ne permettent pas une conclusion formelle. En revanche, on remarque que les courbes semblent assez parasitées par un bruit dans les données, que l'on pourra essayer de traiter par la suite.

2.3 Corrélations

Nous avons construit une heatmap des corrélations afin de mieux comprendre celles-ci :



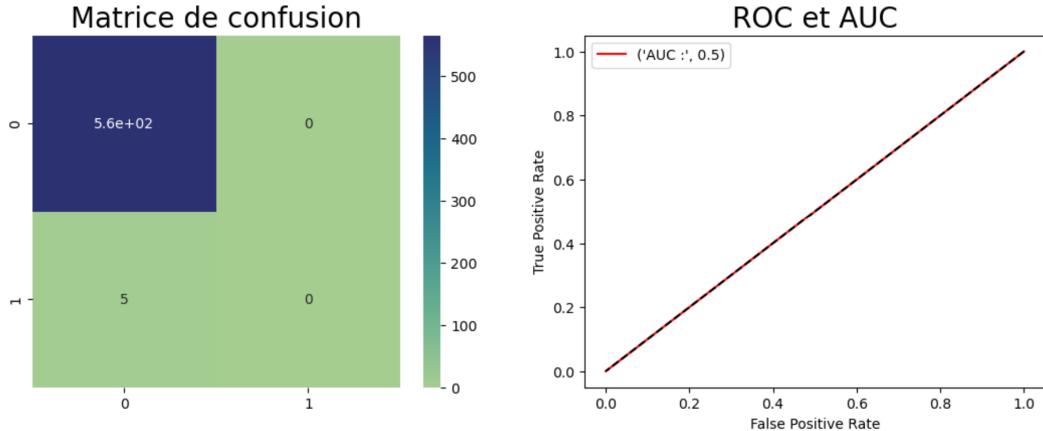
Les corrélations sont assez logiques : le flux pris à t_i est fortement corrélé à celui relevé à t_{i-1} et t_{i+1} , d'où les tâches que l'on observe sur le plot dans la diagonale principalement.

2.4 Premier modèle intuitif

Dans un premier temps, nous avons simplement essayé de construire un modèle très sommaire qui pourrait nous faire comprendre les problèmes que pose ce dataset. Ainsi, nous avons réalisé une random forest et obtenu les résultats suivants :

Accuracy: 0.9912280701754386

A première vue, il semble que le modèle soit très performant. Cependant, lorsque l'on regarde d'autres metrics, on observe ceci:



On remarque en effet que notre classifieur est très proche de l’aléatoire pur, ce que l’on ne pouvait pas remarquer avec seulement l’accuracy. Cela est du au nombre très faible d’une des deux classes par rapport à l’autre. En réalité, notre classifieur se contente de classer toutes les observations en ”non candidates”. Nous pouvons donc ainsi comprendre que le nombre d’éléments dans chaque classe sera un problème à traiter pour l’entraînement de notre modèle.

Ainsi, nous allons donc devoir traiter le dataframe pour le rendre plus utilisable.

3 Traitement des données

3.1 Class balancing

Le premier traitement effectué est donc en lien avec le problème évoqué précédemment : l’une des deux classes est extrêmement majoritaire, ce qui biaise notre modèle lors de l’entraînement. Pour remédier à cela, nous allons utiliser une méthode de class weighting.

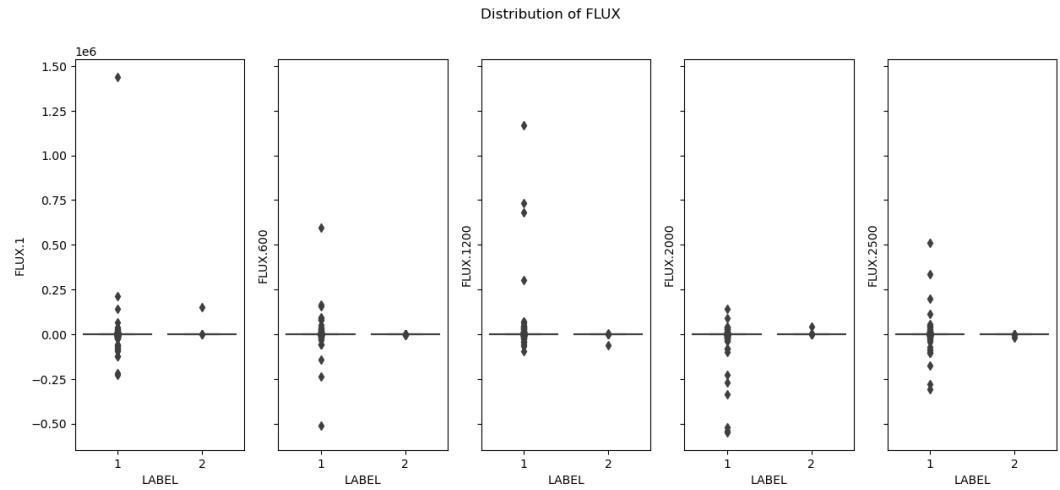
Cette méthode consiste à pénaliser grandement le modèle lorsque celui-ci fait une erreur sur la classe majoritaire. Cette pénalisation est d’autant plus forte que la classe est minoritaire. Cela permet ainsi de faire en sorte que notre modèle prenne en compte cette classe. Nous obtenons ainsi les poids suivants :

- classe 1 (Non candidats): 0.5037
- classe 2 (candidats): 68.7432

3.2 Outliers

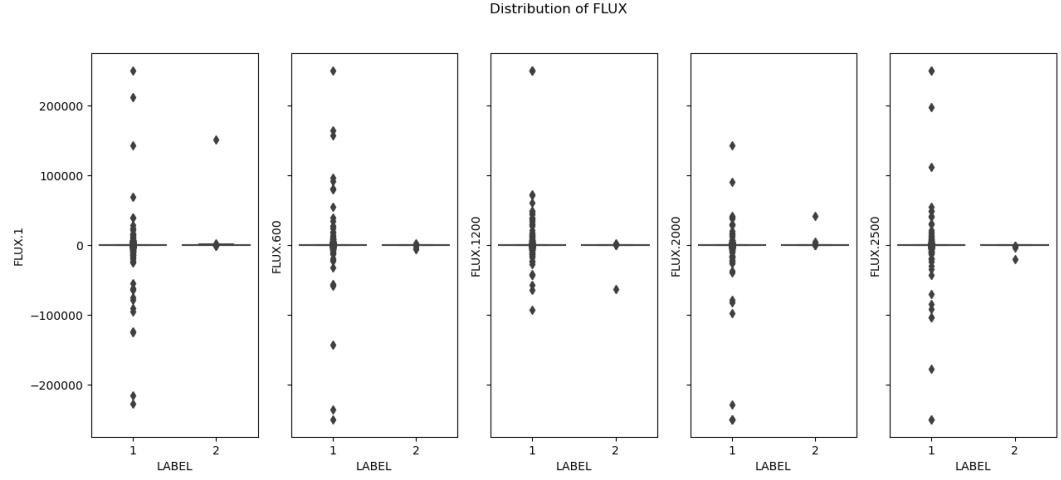
Le traitement des valeurs extrêmes a été une question cruciale. En effet, il fallait d’abord vérifier que ces valeurs extrêmes n’étaient pas une caractéristique de l’un des deux types d’étoile. Les traiter aurait alors fait simplement réduire la performance du modèle.

Pour vérifier cela, nous avons construit un boxplot séparant les deux types d'astres comme suit :



Cela nous a permis de nous rendre compte que ces outliers ne semblaient pas être une caractéristique (pour une classe ou une autre) car il y en a assez peu en comparaison au nombre de données (aucune pour les étoiles à exoplanètes mais cela est sûrement dû au nombre très faible). Nous avons donc choisi de les traiter mais seulement les valeurs vraiment trop hautes (ou trop basses). Typiquement, cela concerne une seule valeur pour le plot 1.

cela nous donne alors les nouveaux boxplots suivants :

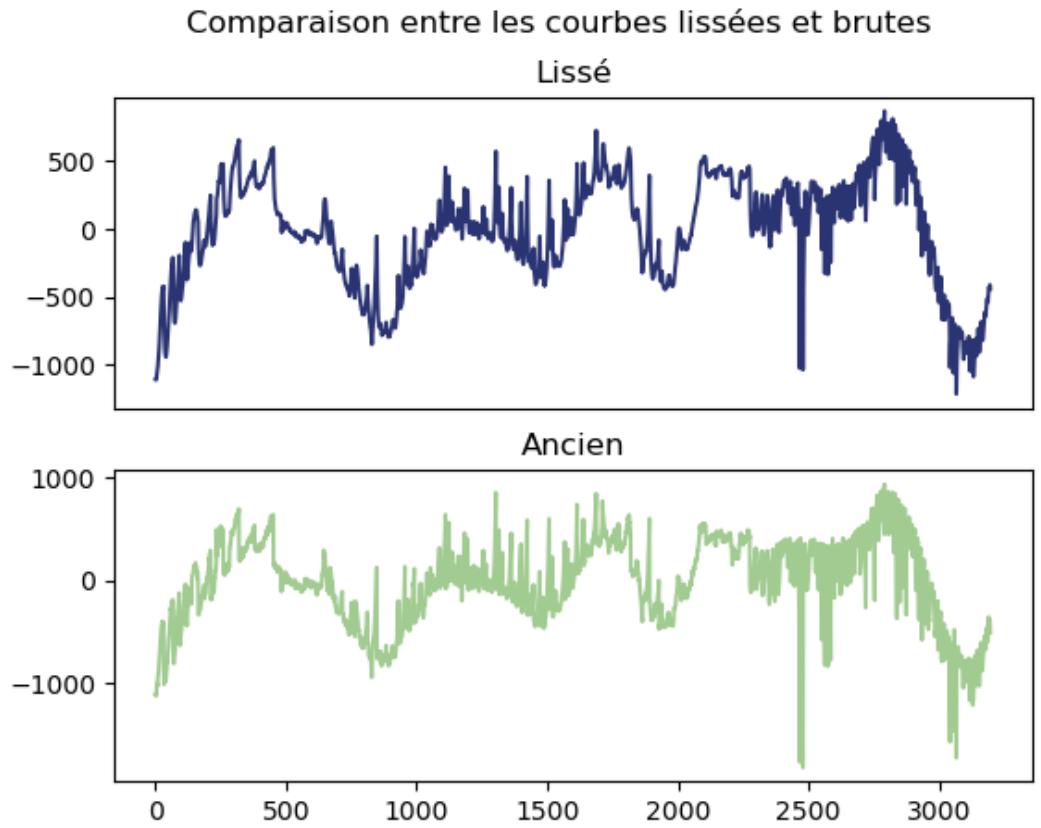


3.3 Smoothing

3.3.1 Lissage Exponentiel

L'idée de base du lissage exponentiel est de calculer une moyenne pondérée des observations passées, les poids diminuant de manière exponentielle au fur et à mesure que les observations vieillissent. Le poids accordé à chaque observation est déterminé par un paramètre de lissage, qui est un nombre compris entre 0 et 1. Plus le paramètre de lissage est proche de 1, plus le poids accordé aux observations récentes est important, et plus la prévision est sensible aux changements à court terme. Inversement, plus le paramètre de lissage est proche de 0, plus les observations plus anciennes sont prises en compte et plus la prévision est stable dans le temps.

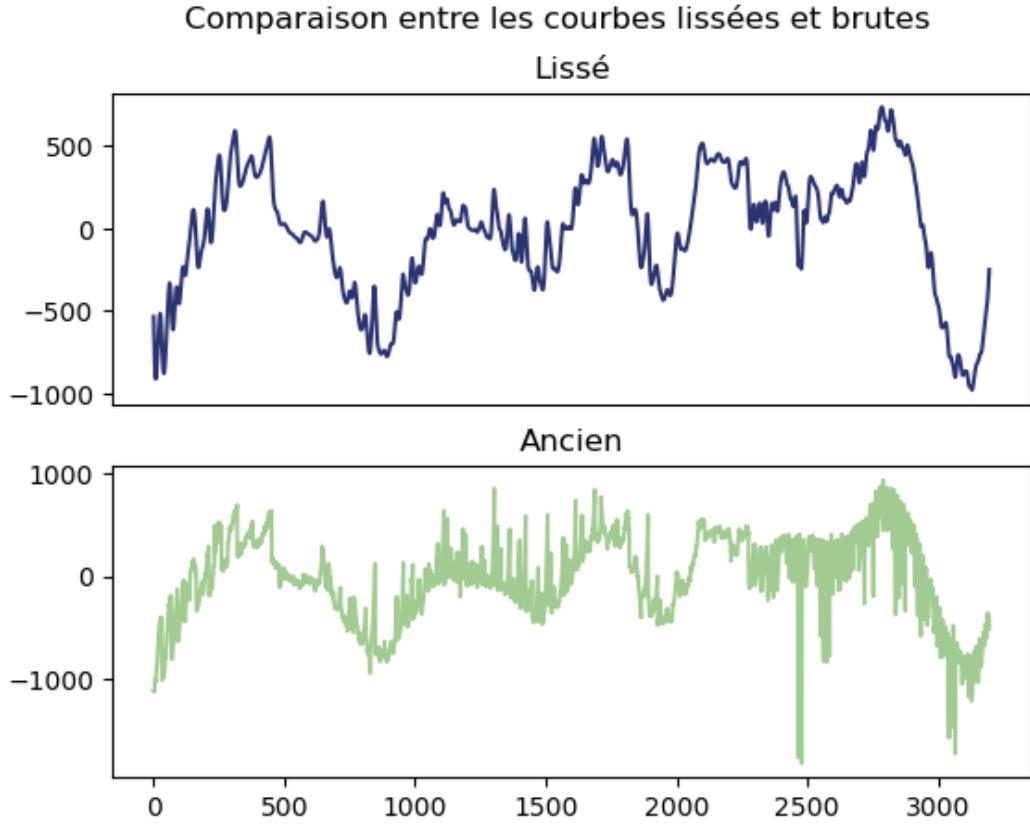
Nous appliquons donc cette méthode à notre dataframe et comparons les courbes pour observer la modification :



Cette méthode de lissage permet visiblement de réduire les brusques variations dans les valeurs et semble surtout aplatisir la courbe. Cependant, on remarque que le bruit est toujours assez visible et pourra être dérangeant par la suite. Une autre méthode de lissage pourrait donc être plus adaptée.

3.3.2 Lissage Gaussien

Cette méthode de lissage est très largement utilisée pour la suppression du bruit dans une série.



En observant la courbe transformée, on peut remarquer que le bruit est beaucoup moins présent dans notre courbe, ce qui pourrait avoir un meilleur impact que le lissage exponentiel. Ainsi, nous avons fait le choix d'utiliser cette méthode pour lisser les séries de notre dataframe.

3.4 Normalisation des données

Après quelques recherches, nous avons appris qu'étant donné que le lissage gaussien avait pour but de réduire le bruit, il est donc préférable de le faire en premier et en suite de normaliser les données. C'est donc ce que nous avons choisi de faire.

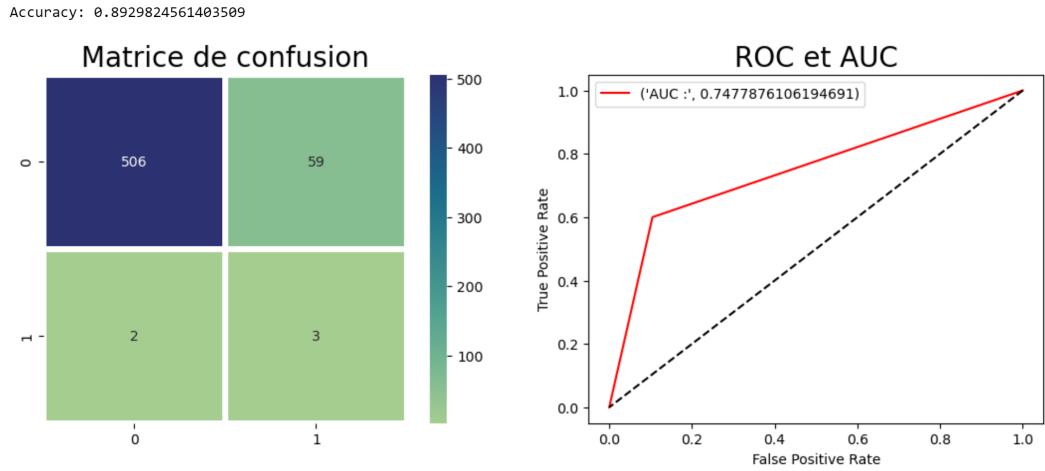
4 Modélisation

4.1 Régression logistique

Le premier modèle testé dans cette partie à été la régression logistique. En effet, notre problème étant une classification binaire, il nous a semblé logique

de tenter cette approche. Nous utilisons les classes équilibrées comme montré précédemment, et un `n_iter` à 1000.

En appliquant cet algorithme, nous obtenons les résultats suivants :

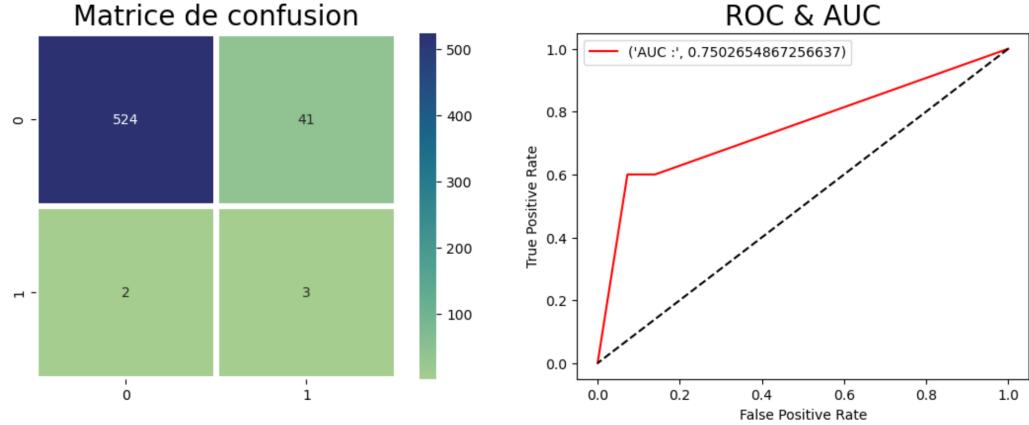


Ces résultats sont beaucoup plus satisfaisants qu'au début. Nous avons 60% des candidats qui sont détectés et assez peu d'erreurs pour les non-candidats.

4.2 KNN

Un algorithme de type KNN peut être assez efficace sur des datasets avec beaucoup de variables. L'un de ses problèmes cependant est si la donnée est inégale dans le nombre de classes, mais ce problème a déjà été réglé. Ainsi, nous avons appliqué notre algorithme KNN et obtenus des résultats très similaires à la régression logistique :

Accuracy 0.9245614035087719

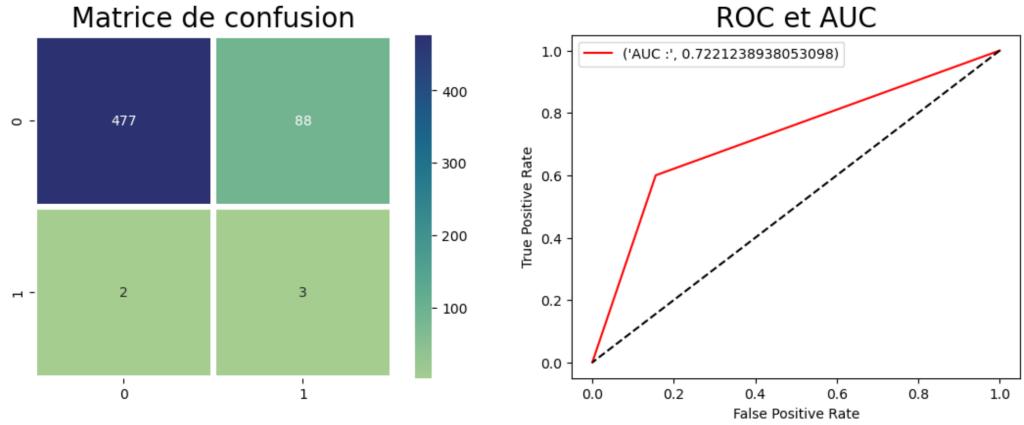


Comme évoqué précédemment, les résultats sont très semblables à la régression logistique. Cependant, on remarque que moins d'erreurs sont commises pour les étoiles non-candidates, ce qui peut faire gagner un temps considérable aux équipes de recherche.

4.3 Decision tree

Notre première approche pour ce modèle a été un modèle basé sur la random forest. Cependant, nous avons constaté qu'augmenter le nombre d'arbres faisait baisser les performances du modèle. Ainsi, nous avons choisi de construire simplement un arbre de décision de profondeur maximale 6. Les résultats sont les suivants :

Accuracy: 0.8421052631578947



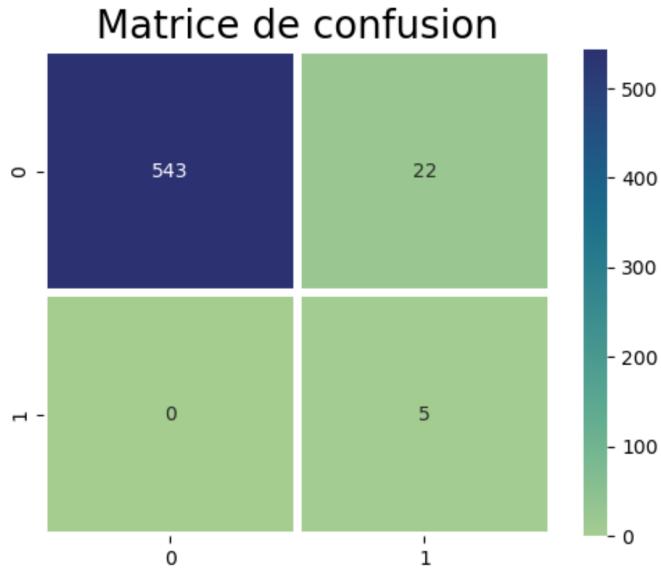
4.4 Approche Deep Learning

Comme nous avons pu le voir avec les modèles précédents, une détection robuste des exoplanètes par une approche Machine Learning ne semble pas encore possible (bien que l'on puisse encore tester d'autres modèles comme des LightGBM ou xGBoost Regressor). En effet, nous ne sommes pas capables d'obtenir un modèle nous permettant d'affirmer que si nous prédisons qu'il y a une exoplanète, alors c'est qu'il s'agit effectivement d'une exoplanète.

Nous allons alors challenger nos modèles de Machine learning, qui ne sont pas encore assez satisfaisants pour espérer les utiliser dans un cas pratique, par une approche Deep Learning avec un réseau de neurones récurrent : le LSTM.

Le LSTM est un type de réseau neuronal récurrent qui excelle dans la modélisation des dépendances à long terme dans des données séquentielles. Sa capacité à conserver une certaine mémoire sur de longues séquences, à traiter des séries de longueur variable et à gérer de manière robuste des données bruitées ou incomplètes en fait un modèle particulièrement adapté à notre étude.

Nous avons donc construit un réseau LSTM avec une fonction de loss personnalisée : En effet, cela est nécessaire dû au grand déséquilibre entre les classes. Nous utilisons ainsi le poids de chaque classe calculé comme précédemment dans notre fonction de perte. Ainsi, en appliquant le modèle, nous obtenons les résultats suivants :



5 Résultats et conclusion

Les modèles de machine learning que nous avons testés fournissent des résultats très similaires. En effet, les prédictions sont à peu près les mêmes. Le problème de ces modèles est qu'il ne détecte pas toutes les étoiles candidates. En effet, il serait préférable d'avoir un modèle qui, pour la même précision, soit juste dans sa détection de toutes les candidates réelles, afin d'être sûr de n'en manquer aucune lors des recherches.

C'est en cela que le modèle de Deep Learning est bien plus performant. En effet, celui-ci n'est pas non plus parfait mais ne manque aucun astre avec une exoplanète confirmée.

Ainsi, il est assez naturel de conclure que le modèle LSTM est le plus efficace pour notre problème