

Projet : Machine learning



**Prédiction du type de tumeurs cancéreuses
grâce à une régression logistique
et le machine learning**

[Introduction](#)

[Matériels et Méthodes](#)

[Scikit-learn](#)

[Keras / Tensorflow](#)

[Résultats](#)

[Etude avec scikit-learn](#)

[Condition normal d'entraînement](#)

[Underfitting](#)

[Overfitting](#)

[Régularisation](#)

[Etude avec Keras et Tensorflow](#)

[Discussion / Conclusion](#)

[Références](#)

Introduction

L'apprentissage automatique fondé sur une approche mathématique et statistique (« Apprentissage automatique » 2021), permet de définir des motifs récurrents permettant une prédiction automatique et intelligente (« Machine Learning : Définition, fonctionnement, utilisations » 2020). Il permet, grâce à des données d'études antérieures, de créer des modèles prédictifs pour des études postérieures. On peut l'utiliser notamment en recherche médicale pour déterminer en avance la potentielle pathologie d'un patient qui regroupe des caractéristiques avec d'autres patients dont leur maladie a déjà été déterminée. Grâce à cela, on peut accélérer le traitement en ciblant plus précisément les causes avant toute autre analyse plus invasive. Cette collection de données fait partie de l'ensemble de données PANCAN RNA-Seq (« UCI Machine Learning Repository: gene expression cancer RNA-Seq Data Set » s. d.) pour l'extraction aléatoire d'expressions génétiques ayant différents types de tumeurs. On peut donc dénombrer 5 classes de tumeurs: BRCA, PRAD, LUAD, COAD, KIRC ainsi que 20 531 expressions géniques pour 800 patients atteints d'une des classes tumorales. Dans cette étude, nous allons essayer de créer un modèle permettant la classification des patients en fonction de leur expression génétique. Tout d'abord avec la bibliothèque "scikit-learn" permettant de faire une régression logistique, puis une analyse complémentaire avec un réseau neuronale grâce à la librairie "tensorflow" et "keras"

Matériels et Méthodes

I. Scikit-learn

Librairie open-source développée en Python et Cython pour implémenter rapidement des algorithmes pour le traitement de données en machine learning (« Scikit-learn » 2021). On peut donc faire des algorithmes de régression logistique, de random forest ou de machines à vecteurs de support. Nous avons utilisé un algorithme de régression logistique pour analyser nos données. En effet, nos données étant associées à une classification, la régression logistique est parfaitement adaptée à cette étude.

II. Keras / Tensorflow

La librairie Keras open-source permet au data scientist d'utiliser des outils tels que "Tensorflow" (« TensorFlow » 2021) pour l'analyse de réseau de neurones profonds. (« Keras » 2020) On peut ainsi créer des couches de neurones traitant des informations de prédiction avec un effet "feedback" pour augmenter la précision de leurs résultats. En effet, le fait de pouvoir revenir en arrière dans son traitement avec des analyses postérieures permet d'affiner plus précisément les résultats de sorties.

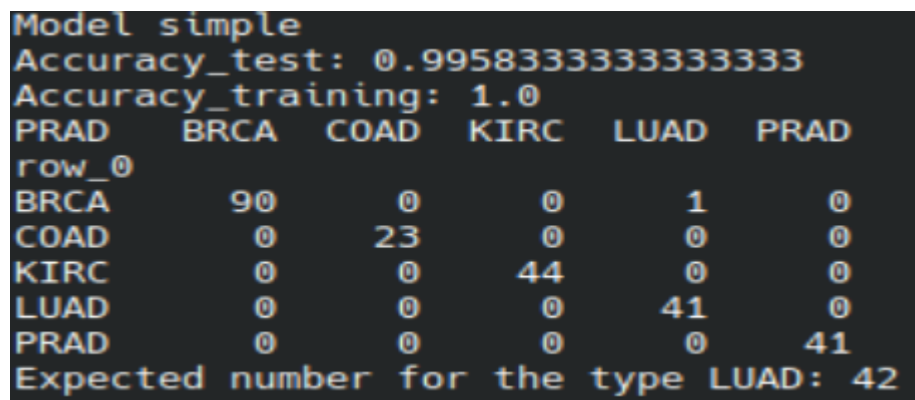
Résultats

Pour la création du modèle simple, nous avons une matrice de résultats d'expression génique de gènes sélectionnés avec soin qui sont mis en corrélation avec des patients atteints de type de cancer bien déterminé. Ces données vont ainsi permettre d'entraîner et de tester le modèle pour le rendre le plus optimal possible.

I. Etude avec scikit-learn

Nous allons commencer avec la bibliothèque "scikit-learn" et la régression logistique pour l'analyse de nos données. On commence par créer le modèle de régression logistique puisque notre étude est une classification. Nous allons ensuite découper notre échantillon de données avec un ratio 70% des données qui vont entraîner le modèle et 30% qui vont tester la fiabilité du modèle. Grâce à la méthode "fit" nous lançons l'entraînement du modèle sur nos données d'entraînements et sur leurs classifications correspondantes. Nous faisons fonctionner ensuite le modèle sur les données de test puis nous récupérons la classification prédite. Cette classification est comparée à notre classification réelle et nous pouvons ainsi obtenir un tableau de contingence.

A. Condition optimal d'entraînement



```
Model simple
Accuracy_test: 0.9958333333333333
Accuracy_training: 1.0
PRAD    BRCA    COAD    KIRC    LUAD    PRAD
row_0
BRCA      90      0      0      1      0
COAD      0      23      0      0      0
KIRC      0      0      44      0      0
LUAD      0      0      0      41      0
PRAD      0      0      0      0      41
Expected number for the type LUAD: 42
```

	PRAD	BRCA	COAD	KIRC	LUAD	PRAD
row_0						
BRCA		90	0	0	1	0
COAD		0	23	0	0	0
KIRC		0	0	44	0	0
LUAD		0	0	0	41	0
PRAD		0	0	0	0	41

Figure 1 : Tableau de contingence sur la classification du modèle de régression logistique avec l'ensemble des données.

Pour aller plus loin dans notre analyse, on peut récupérer la variation de la précision de notre modèle durant la phase d'apprentissage et de test en faisant varier la quantité de données. Nous sommes donc sur une plage allant de 20 échantillons à 500 échantillons.

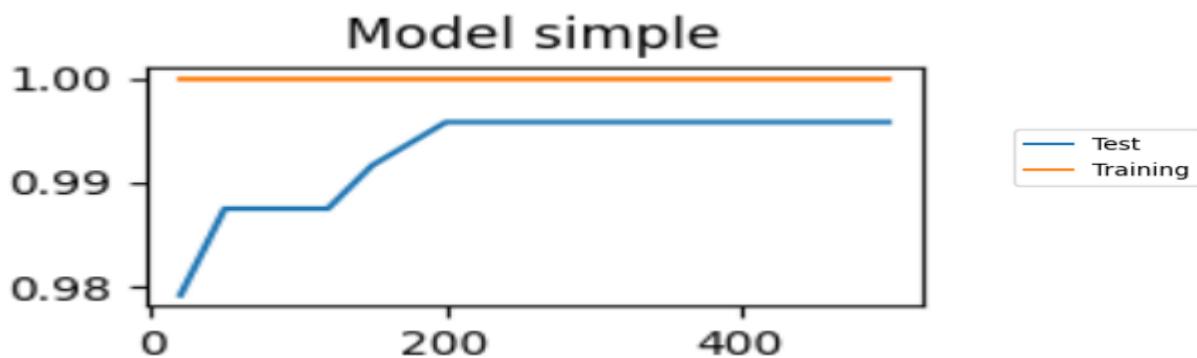


Figure 2 : Variation de la précision du modèle dans des conditions normales

Après avoir étudié la fiabilité du modèle, nous nous sommes concentrés sur les potentielles failles. En effet, un modèle ne peut être précis que s'il dispose de l'ensemble des paramètres lui permettant de comprendre le plus de nuances possible. Par conséquent, on peut être dans des cas de sous-entraînement ou de surentraînement mais aussi de correction d'erreurs extrêmes ou de régularisation trop négligeable.

B. Underfitting

Dans le cas d'un sous-entraînement, plusieurs causes peuvent être déterminées :

- le temps d'entraînement est trop court, pas assez d'itérations
- le nombre de données de classification est trop faible pour une bonne compréhension des motifs spécifiques à un type

Dans notre cas, nous avons pris un nombre de labels trop faible par rapport à la quantité de données. Notre modèle a donc du mal à affiner les spécificités.

```
Underfitting
Accuracy_test: 0.5708333333333333
Accuracy_training: 0.5660714285714286
PRAD  BRCA  COAD  KIRC  LUAD  PRAD
row_0
BRCA   63    4    14    20    14
COAD   3    19     0     3     1
KIRC   14     0    28     7     0
LUAD   8     0     2    10     9
PRAD   2     0     0     2    17
Expected number for the type LUAD: 42
```

Figure 3 : Tableau de contingence pour un sous-entraînement

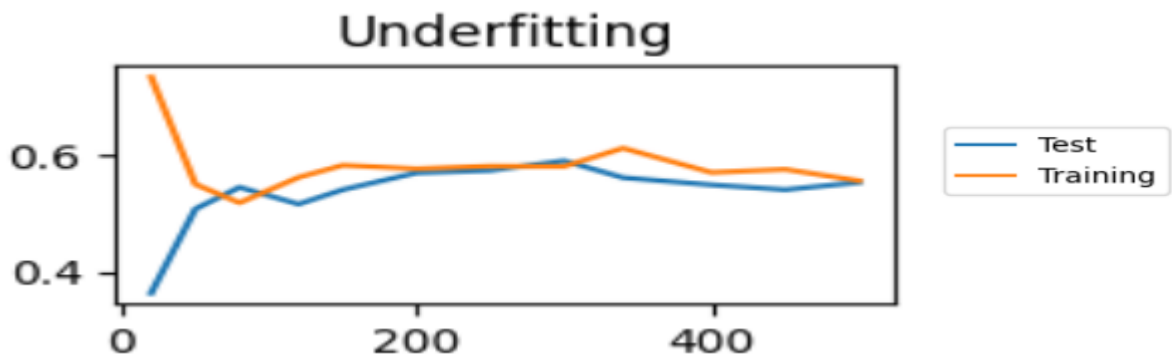


Figure 4 : Variation de la précision sous une condition d'underfitting

C. Overfitting

Dans le cas où notre modèle s'est surentraîné, on n'observe pas de variation de la précision au fur et à mesure de l'apprentissage car dès la première plage de données notre modèle est parfait. Il connaît déjà toutes les données et leur classification avant même de s'entraîner. Ce problème est souvent dû à une quantité de données d'expression génétique trop faible ou un temps d'entraînement trop long par rapport à l'analyse à effectuer.

```
Overfitting
Accuracy_test: 1.0
Accuracy_training: 1.0
PRAD    BRCA    COAD    KIRC    LUAD    PRAD
row_0
BRCA      2      0      0      0      0
COAD      0      1      0      0      0
KIRC      0      0      1      0      0
LUAD      0      0      0      1      0
PRAD      0      0      0      0      1
Expected number for the type LUAD: 1
```

Figure 5: Tableau de contingence condition de surentraînement

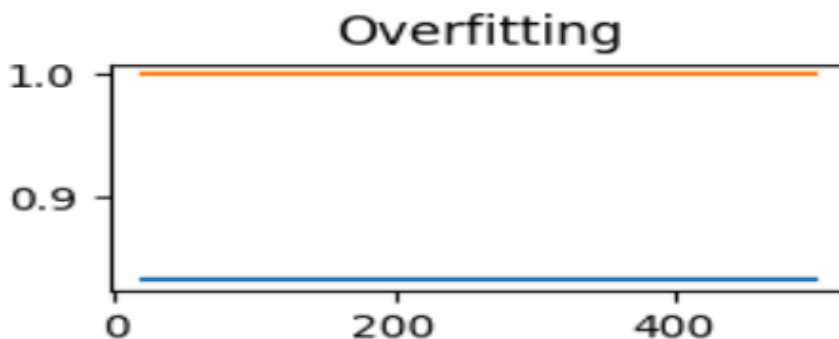


Figure 6 : Variation de la précision dans le cas d'un overfitting sur nos données

D. Régularisation

Pour améliorer notre modèle et compenser certaines erreurs du modèle à trop suivre les données, on peut faire une régularisation mais il faut faire attention de ne pas trop aller dans les extrêmes de la régularisation. Soit une correction trop forte empêchera le modèle d'apprendre correctement les caractéristiques de classification, soit une trop faible rendra le modèle trop fidèle aux données et donc trop complexe pour être réutilisable sur d'autres données.

```
High regulation
Accuracy_test: 0.375
Accuracy_training: 0.375
PRAD  BRCA  COAD  KIRC  LUAD  PRAD
row_0
BRCA   90   23   44   42   41
Expected number for the type LUAD: 42
```

Figure 7: tableau contingence avec une forte régularisation $1e-20$

```
Low regulation
Accuracy_test: 0.9958333333333333
Accuracy_training: 1.0
PRAD  BRCA  COAD  KIRC  LUAD  PRAD
row_0
BRCA   90    0    0    1    0
COAD   0   23    0    0    0
KIRC   0    0   44    0    0
LUAD   0    0    0   41    0
PRAD   0    0    0    0   41
Expected number for the type LUAD: 42
```

Figure 8 : Tableau de contingence avec une faible régularisation $1e20$

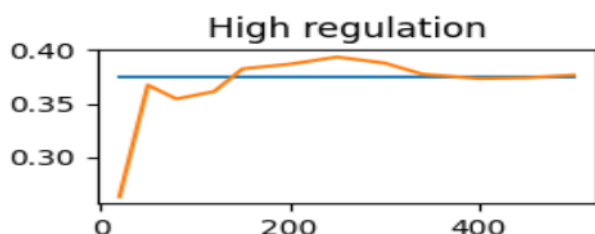


Figure 9: Variation accuracy en cas de forte régularisation

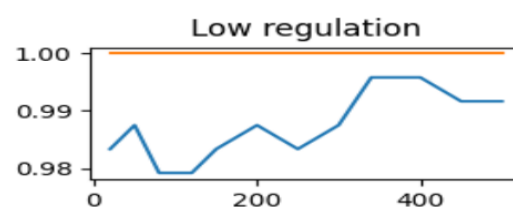


Figure 10: Variation accuracy en cas faible régularisation

II. Etude avec Keras et Tensorflow

Une autre approche, pour la classification des tumeurs selon les profils d'expression génétiques, est une analyse grâce à un réseau de neurones artificiels. C'est la base de l'intelligence artificielle. Pour cela, on commence par encoder nos 5 classes grâce à des entiers et on fait une transformation en catégorie pour l'analyse neuronale. On crée grâce au module "Keras" des couches de neurones successives qui vont analyser nos données et les ranger en fonction de leur appartenance à un type tumorale. On sépare les données comme avec la méthode de régression logistique et on les transmet à notre réseau de neurones en lui indiquant que la répartition des données est uniforme pour éviter les biais. On filtre notre modèle et on fait ensuite les étapes de comparaison des données prédites avec les données réelles pour obtenir un tableau de contingence.

col_0	0	1	2	3	4
row_0					
0	90	0	0	0	0
1	0	41	0	0	0
2	1	0	41	0	0
3	0	0	2	42	0
4	0	0	1	0	22
0.9833333333333333					

Figure 11 : tableau de contingence avec un modèle de réseau de neurones codifiés (BRCA = 0, PRAD=1, LUAD=2, KIRC=3, COAD=4)

Discussion / Conclusion

Dans les conditions optimales du traitement par le modèle ([figure 1](#)) nous constatons tout de même un faible pourcentage d'erreurs avec une prédiction en BRCA à la place de LUAD. Ce pourcentage doit être considéré lors d'une étude réelle pour limiter et prédire les risques d'erreurs éventuels. On remarque dans la [figure 2](#) que notre précision augmente au fur à mesure des entraînements sur des sets de données plus importants pour se rapprocher des précisions optimales de test . On peut aussi voir de nombreux problèmes de prédictions dans la [figure 3](#) à cause de l'underfitting et la [figure 4](#) nous montre que la précision n'arrive pas à atteindre des valeurs optimales malgré le nombre d'itérations et la quantité de données. Même problème mais inversé dans la [figure 5](#) et [6](#) à cause du surentraînement. Ces failles rendent caduque le modèle. Il ne s'agit pourtant que d'une variation de la quantité de données classifiées. Dans des conditions réelles, l'accès à un nombre important d'échantillons est complexe par manque de moyen, ce qui exprime une difficulté certaine à exploiter ce genre d'approche. Dans la [figure 11](#), on peut visualiser un tableau de contingence correspondant à l'approche avec les réseau neuronal dans des conditions optimales, il est donc comparable à la [figure 1](#). Cependant on constate que le taux d'erreur pour la classification est plus important, ce qui montre que le modèle doit être plus optimisé.

Notre étude a ainsi permis de montrer brièvement comment créer des modèles permettant de prédire des pathologies. Cette analyse est indispensable pour l'avenir, grâce au nombre de plus en plus important de données recueillies sur des patients, afin de prédire les pathologies de nouveaux patients. Mise en corrélation avec une médecine génomique personnalisée cela augmenterait considérablement la vitesse de prise en charge et le ciblage des éléments associés à une maladie, ce qui entraînerait un meilleur taux de guérison. C'est une avancée majeure qui place l'informatique, en particulier l'intelligence artificielle, au centre de la gestion des pathologies et à leur potentielle résolution. Cependant, nous avons encore de nombreux axes à améliorer, au vu des failles mises en évidence, afin de minimiser les risques pour les patients et d'établir de meilleurs diagnostics.

Accès aux données pour l'analyse : <https://github.com/TigiGln/Stat3.git>

Références

- « Apprentissage automatique ». 2021. In *Wikipédia*.
https://fr.wikipedia.org/w/index.php?title=Apprentissage_automatique&oldid=188876947.
- « Keras ». 2020. In *Wikipédia*. <https://fr.wikipedia.org/w/index.php?title=Keras&oldid=173323158>.
- « Machine Learning : Définition, fonctionnement, utilisations ». 2020. *Formation Data Science | DataScientest.com* (blog). 18 novembre 2020.
<https://datascientest.com/machine-learning-tout-savoir>.
- « Scikit-learn ». 2021. In *Wikipédia*.
<https://fr.wikipedia.org/w/index.php?title=Scikit-learn&oldid=181501393>.
- « TensorFlow ». 2021. In *Wikipédia*.
<https://fr.wikipedia.org/w/index.php?title=TensorFlow&oldid=186919162>.
- « UCI Machine Learning Repository: gene expression cancer RNA-Seq Data Set ». s. d.
Consulté le 9 janvier 2022.
<https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq>.