L2 MI – Mini Projet

Groupe 01 & 01 bis

Challenge HADACA – Groupe Doctor

Membres : ABICHOU Asma, CINTRA Paul, TRIKI Bilelle, CORNEDE Cédric, AMRANE Lydia, NAIT-LARBITakfarinas.

URL du challenge: **https://codalab.lri.fr/competitions/333**
Le numéro de Soumission de CODE : 8826
URL de la Vidéo YouTube :
https://youtu.be/RcYYz6Wtb8
URL du repository Github: https://github.com/doctor-dl/doctor
URL des diapos de la présentation : https://github.com/doctor-dl/doctor/blob/master/Doctor.pdf?fbclid=IwAR1EHpEqbNLDJRtruqyhgUAUa2dLSL1jJd DOqjNt zjSnFOC5UydtzsACc4Q

---

1. Motivation et contexte :

Nous avons décidé de travailler sur le projet HADACA qui aborde une maladie qui est connue comme la maladie du siècle souvent imputable à notre mode de vie « LE CANCER ».

Le but de ce défi est d'apprendre comment atteindre de bons résultats de classification dans un contexte de grande dimension et déséquilibré.

C'est simple c'est le problème de classification qui nous a attiré car il nous permettra d'en apprendre davantage sur le problème de l'apprentissage automatique dans le contexte d'une application réelle qui est basé sur une tâche de classification et aussi qui nécessite un prétraitement important, y compris une factorisation matricielle.

2. Description du problème et donnée :

C'est un problème de classification multi-classe. Notre but consiste à classer les stades du cancer parmi une population spécifique dans l'une des 10 catégories. Les données sont une matrice de lignes (nombre de patients) * de colonnes

(nombre de caractéristiques par patient). Les caractéristiques correspondent aux informations de méthylation liées aux conditions médicales de chaque patient [5]

L'ensemble de données que nous manipulons est un ensemble de patients qui sont étiquetés sur dix classes : stade i, stade ia, stade ib, stade iia, stade iib, stade iiia, stade iiib, stade iv, non rapporté et Nan [6] ces dix classes représentent les différents stades de cancer diagnostiqués chez les différents patients.

3. Contribution :

Binôme 1 (TAKFARINAS & Lydia): travaillera sur le prétraitement, en suivant le plan de classe PREPROCESSOR.PY dans notre rapport Github.
(testé par MODEL.PY). Cette classe implémentera des méthodes de sélection des fonctionnalités telles que la méthode.
Binôme 2 : Abichou Asma et Cintra Paul , nous avons travaillé sur le classifieur. Notre objectif principal est d'améliorer le taux de classification pour ça on va comparer les algorithmes proposés dans la première partie donc pour classifier nous devons récupérer les données qui viennent d'être traitées.

Le binôme 3 (Triki Billel et Cornede Cédric) visualisation s'occupe principalement de la communication des résultats ainsi que de leurs représentations graphiques pour permettre une vérification ayant pour but de voir si notre classificateur fonctionne correctement, cette vérification peut se faire en utilisant par exemple une matrice de confusion. Nous aurons beau etre sur à 100 % de notre partie théorique un affichage pertinent des résultats permet de confirmer que notre projet va dans le bon sens . Avec le soutien d'une bonne représentation graphique des résultats sous forme de tableaux ou de diagrammes, ce binôme apporte également une aide au binôme preprocessing leur offrant une vision plus claire et précise des données qu'ils manipulent .

Dans ce projet , ce binôme s'est également occupé de la création et la réalisation de la vidéo ayant pour but de présenter notre projet et de fournir une explication claire concise sur son déroulement.

4. Description des classes :

1. Preprocessing : (TAKFARINAS & Lydia)

« Le **prétraitement des données »** est une étape importante du processus d'exploration des données. Cette étape décrit tout type de traitement effectué sur des données brutes afin de les préparer pour une autre procédure de traitement par exemple

classification. C'est souvent la phase la plus importante d'un projet d'apprentissage automatique, car il transforme les données brutes en un format qui sera traité plus facilement et plus efficacement par le model d'apprentissage, il existe plusieurs méthodes pour faire le « preprocessing » par exemples :

- Suppression des caractéristiques les moins significatives.
- Réduction de la dimension des données

L'objectif de notre projet est de classer les stades du cancer parmi une population spécifique dans l'une des 10 catégories voir figure « 6 ». Les données sont une matrice de lignes (nombre de patients <5000>) * de colonnes (nombre de caractéristiques par patient <4000>). Cette matrice avec 20000000 cases est d'une dimension énorme pour qu'elle soit rapidement traitée par notre Classifieur. Afin d'accélérer l'apprentissage et de facilité la taches de notre model, on décomposera cette grande matrice en une matrice d'une dimension moins importante, mais tout en gardant la signification des données initiales. Garder la signification des données bruts, c'est sur cet aspect qu'on doit faire attention et pour cela on a fait des tests unitaires afin de vérifier et de contrôler la pertinence et la signification de la matrice de données résultante après cette étape, on en parlera de ces tests à la fin de cette partie.

Pour effecteur cette décomposition on a opté pour la méthode de SVD (SINGULAR Values décomposition), qui est un pure procédé d'algèbre linéaire, c'est un outil important de factorisation des matrices rectangulaire. On a opté pour cette méthode car elle fonctionne parfaitement avec des matrice SPARCE qui veut dire les matrices qui contiennent beaucoup de valeurs nulles (Zéro), et du fait que notre matrice de données brutes est une matrice de densité faible on a choisis la SVD sur la méthode PCA qui est une méthode très similaire mais qui beaucoup plus compatibles avec les matrices denses.

Pour effectuer ce prétraitement on a créé une **CLASSE PREPROCESSOR.PY** dans laquelle on implémente l'algorithme de la décomposition, notre implémentation se base entièrement sur la bibliothèque SCIKIT-LEARN, qui est une bibliothèque très riche en termes de méthodes et de model d'apprentissage automatique. Cette classe à un hyper paramètres qui joue un rôle très important dans le déroulement de cette décomposition qui est **Nb COMPONENTS.** A travers ce paramètres on peut contrôler la dimension de la matrice de données résultantes après la décomposition.

Et afin d'intégrer ce Preprocessing dans notre model, on a créé une classe **MODEL.PY** dans laquelle on a implémenté un pipeline pour jumeler la décomposition et la classification, ce pipeline effectue dans un premier temps la réduction de dimension (SVD) et ensuite il procède à la classification en utilisant la matrice de données obtenue par le prétraitement.

- **TESTS UNITAIRE :**

Dans la classe MODEL.PY on effectuer trois tests pour le préprossecing afin de comparer entre les méthodes et vérifier la pertinence et la signification de la matrice de données résultantes.

1. Le premier test consiste à comparer un model avec et sans préprocessing, après ce test on a obtenue des scores très proches mais avec un temps d'exécution beaucoup mois important pour le model incluant le prétraitement (SVD)
2. Le premier test consiste à comparer la méthode SVD et PCA, qui sont deux méthodes très similaires, mais la seule différence c'est que la méthode SVD fonctionne efficacement sur les matrice qui ne sont pas denses.
3. Dans le troisième test, on fixe un modèle de classification et on fait varier l'hyper paramètre NB_COMPONENTS, et pour chaque variation on enregistre le score du model, ce qui nous a permet d'établir le graphe du score en fonction de la dimension de la matrice d'apprentissage (FIGURE 2), ce graphe montre que pour une dimension petite de la matrice d'apprentissage le score n'est pas vraiment satisfaisant, et cela se justifie par la perte d'information lors de la décomposition, et à partir d'un certain seuil on remarque que le score à une tendance de se stabiliser malgré l'augmentation de NB_COMPONENTS, cela nous permet de déterminer la dimension de la matrice d'apprentissage optimale pour gagner de la performance et surtout de l'efficacité pendant l'exécution.

2. Classification (Abichou Asma Et Cintra Paul) :

Il s'agit d'un défi d'apprentissage automatique « classification multi-classe » centré sur une tâche de classification qui nécessite un prétraitement de donnée.

Pour simplifier **« La classification multi class »** signifie une tâche de classification comportant plus de deux classes; par exemple, classer un ensemble d'images de fruits qui peuvent être des oranges, des pommes ou des poires. La classification multi class repose sur l'hypothèse que chaque échantillon est attribué à une seule et même étiquette: un fruit peut être une pomme ou une poire mais pas les deux à la fois.
Notre objectif principal est d'améliorer le taux de classification pour ça on va comparer les algorithmes proposés dans la première partie donc pour classifier nous devons récupérer les données qui viennent d'être traitées ! Notre partie constitue un premier pas dans le développement d'un système d'aide au diagnostic médical intelligent qui peut aider les experts dans le domaine d'une détection précoce de cancer.

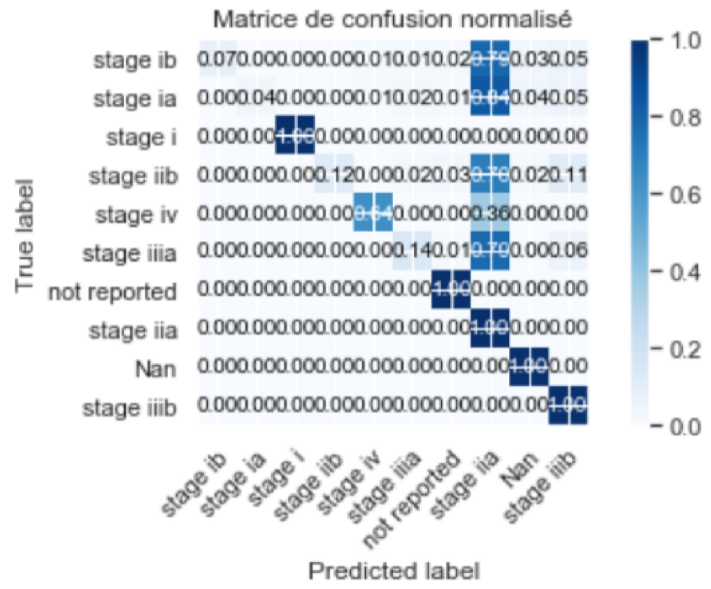
Les réseaux de neurones comme le perceptron (qui est organisé en 3 parties : la couche d'entrée (input layer) , la couche cachée (hidder layer) elle s'agit du cœur de notre perceptron , et la couche de sortie (output layer) cette dernière représente le résultat final de notre réseau , sa prédiction) ; sont une famille d'algorithmes largement utilisées en apprentissage statique , en particulier pour des problèmes d'apprentissage supervisé. Dans une première phase d'apprentissage on a soumis aux modèles des cas

étiquetés (ex malades , sains) et toutes mes variable d'entrée que l'on pense être pertinentes (ex : âge , poids , antécédents…). Notre classifieur se comporte comme « un étudiant en médecine » qui apprendrait à reconnaitre des personne en consultant un nombre suffisant de dossiers . Les règles qu'il établit sont complexes et sans doute non linéaires, ainsi que les variables d'entrée utilisées peuvent être très nombreuses. Dans une deuxièmes phase, dite phase de test, on a fait deux tests , le premier consiste à faire varier le paramètre nb_epoque qui représente le nombre d'itération , et faire tourner le model pour chaque valeur de nb_epoque pour un même preprocessing et un taux d'apprentissage fixe aussi, ce qui va nous permettre d'établir un graphe du score au fonction de nombre d'itération. Ce test peut nous

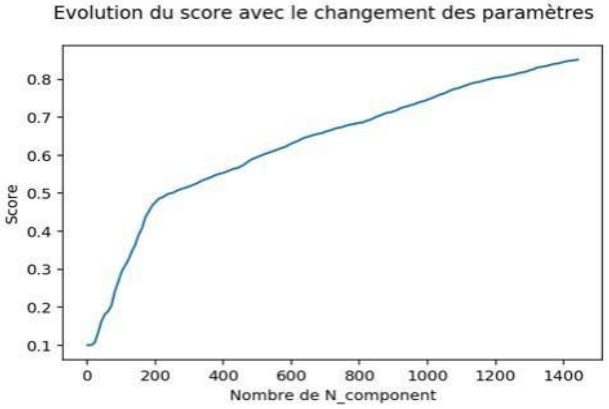
nous permettre de terminer le seuil du sur-apprentissage en utilisant les résultats de CrossValidation, et le deuxième consiste à faire varier le paramètre tda qui représente le taux d'apprentissage, et faire tourner le model pour chaque valeur de tda pour un même preprocessing et un nb_epoque fixe aussi, ce qui va nous permettre d'établir un graphe du score au fonction de taux d'apprentissage. Lorsque les modèles arrivent à reconnaitre les malades avec une performance satisfaisante, on estime qu'ils ont trouvé une règle liant les variables d'entrée et la décision attendue. Même si les réseaux de neurones ne sont qu'un sous ensemble des algorithmes d'apprentissage statistique. Il existe des algorithmes bien connus du monde médical, comme la régression logistique, et les réseaux bayésiens, la performance de notre perceptron est de 85%.

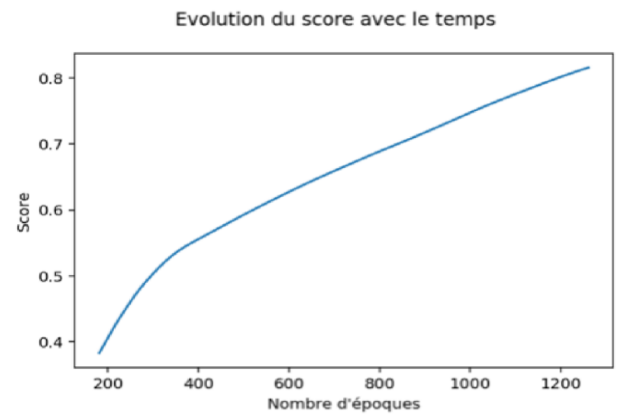
Le rôle des figures est particulièrement importante dans cette partie. En effet, après une évaluation de nos paramètres pour notre perceptron, nous avons besoin à quel point notre perceptron est fonctionnel.

Une figure très utile pour cet objectif :la matrice de confusion



Nous voyons sur notre matrice que notre classifieur particulièrement bien les 8 dernières classes. Cependant, les données provenant des deux premières classes ne sont pas bien classées. Enfin, après évaluation de notre score, nous avons voulu connaître son évolution au cours du temps ainsi qu'en changeant les paramètres.
Le score augmente de 0,5 à 0,95 lorsque le nombre de N_components augmente de 200 à 1400, dans le cas du perceptron de la bibliothèque sklearn. On ne s'attend pas à ce que le score soit égal à 1 car cela signifierait que tous les patients auraient bien été classés, c'est hautement improbable, un perceptron commet des erreurs.
On rencontre d'ailleurs des résultats similaires avec notre second perceptron, qui atteint un score maximal de 0,85. On n'observe pas d'overfitting sur nos courbes, cela serait peut être arrivé avec un entraînement différent.
De plus, le nombre d'époques effectués croît avec le score, jusqu'à atteindre 0,9 comme score pour 1200 époques, comme vu sur la seconde figure.
On peut en conclure que N = 1400 et 1200 époques sont de bons paramètres.





Analyse des données:

Tout d'abord , nous avons voulu commencer par avoir un simple aperçu des données avec des graphiques simples :

```
Number of examples = 4000
Number of features = 5000
          Class
0      stage ib
1      stage ia
2       stage i
3     stage iib
4      stage iv
5    stage iiia
6  not reported
7     stage iia
8           Nan
9    stage iiib
Number of classes = 10
```

	methyl_0	methyl_1	methyl_2	methyl_3	methyl_4	methyl_5	methyl_6	methyl_7	methyl_8	methyl_9	...	methyl_4991	methyl_4992	methyl_4993	methyl_4
0	0.702318	0.724066	0.602563	0.696525	0.205544	0.434269	0.714714	0.137421	0.616975	0.104775	...	0.070442	0.569205	0.134207	0.550
1	0.062884	0.691114	0.594138	0.732033	0.148307	0.467048	0.651207	0.059064	0.692874	0.034401	...	0.011138	0.640959	0.199193	0.485
2	0.169687	0.584910	0.892193	0.776171	0.159332	0.222362	0.791159	0.223618	0.807791	0.198727	...	0.107689	0.593981	0.148779	0.438
3	0.141969	0.627048	0.799960	0.591641	0.136220	0.482781	0.682585	0.123101	0.415066	0.111355	...	0.146476	0.458663	0.093836	0.480
4	0.244063	0.462871	0.797939	0.574462	0.150395	0.227411	0.612413	0.157773	0.497738	0.175969	...	0.078215	0.463431	0.144891	0.302

Figure 1:Visualisation claire de tous les paramètres de nos données

Cette première figure ,simple mais efficace , nous permet de tout de suite comprendre à quoi s'attendre avec nos données c'est-à-dire le nombre de données, les composantes et leur nombre, le nombre de classes et qu'elles sont-elles.

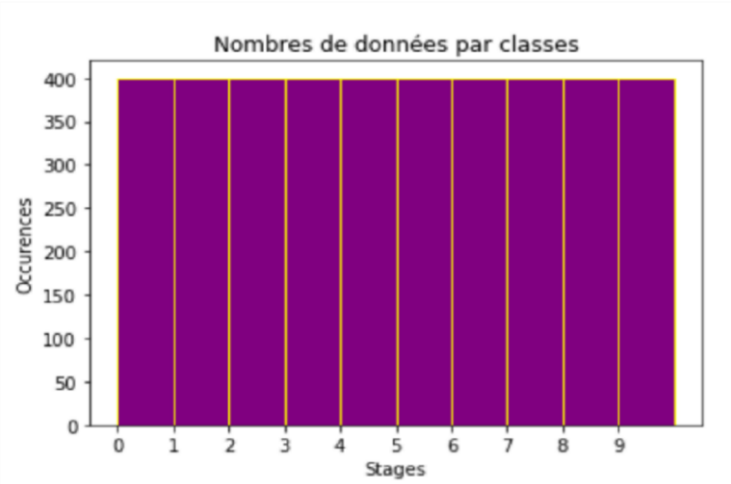


Figure 2:Le nombre de données par stage du cancer

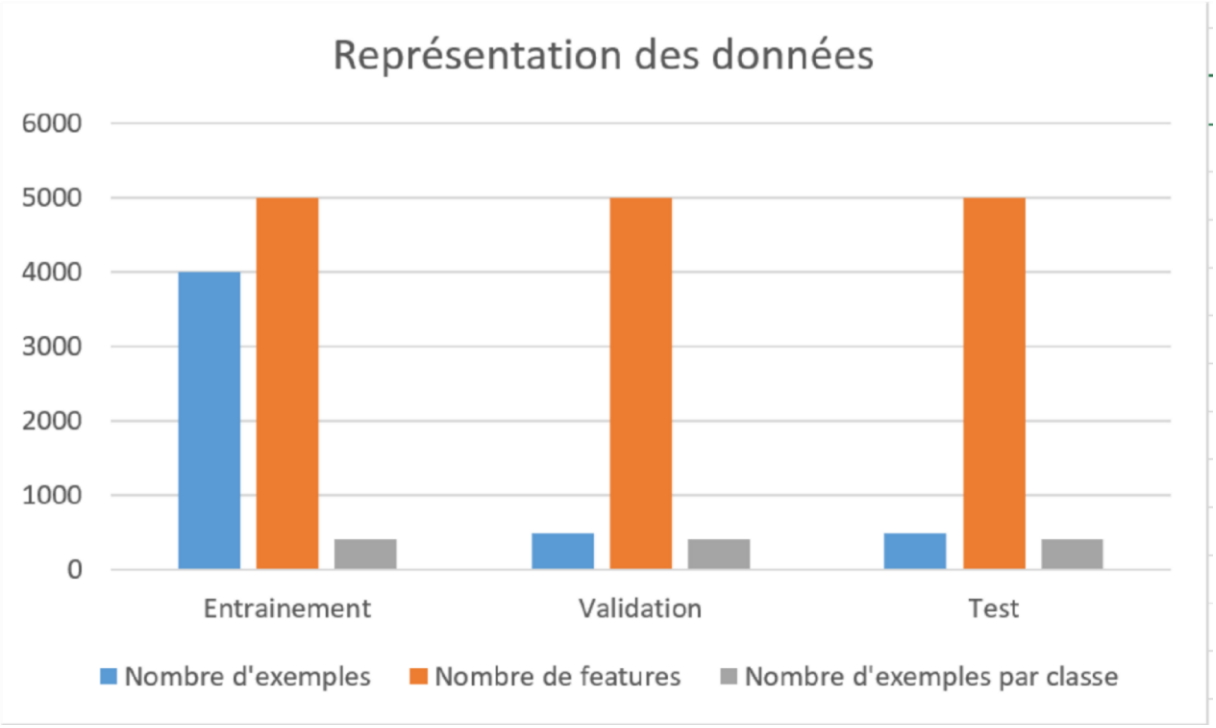


Figure 3(Tableau Bonus):Visualisation simplifiée de nos données

Avec cette figure, nous voyons que le nombre de données pour chaque classe est le même ,les classes ont donc une importance similaire ,il n'a paru pas nécessaire lors du traitement des données de plus représenter une classe qu'une autre (parce que un stade du cancer aurait été plus présent dans la population par exemple).

Il était alors nécessaire d'avoir un affichage plus concret de nos données mais un problème se posait :le nombre de composantes à manipuler est de 5000 . L'analyse en composantes principales (PCA) était la méthode la plus adéquate pour notre cas. Nous avons alors réduit l'analyse à 4 composantes.

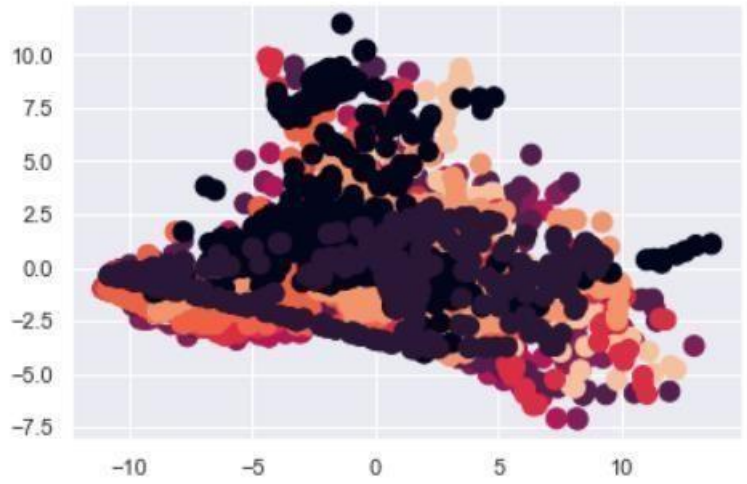


Figure 4:Affichage des données avec leur classes associées à l'aide de la méthode PCA

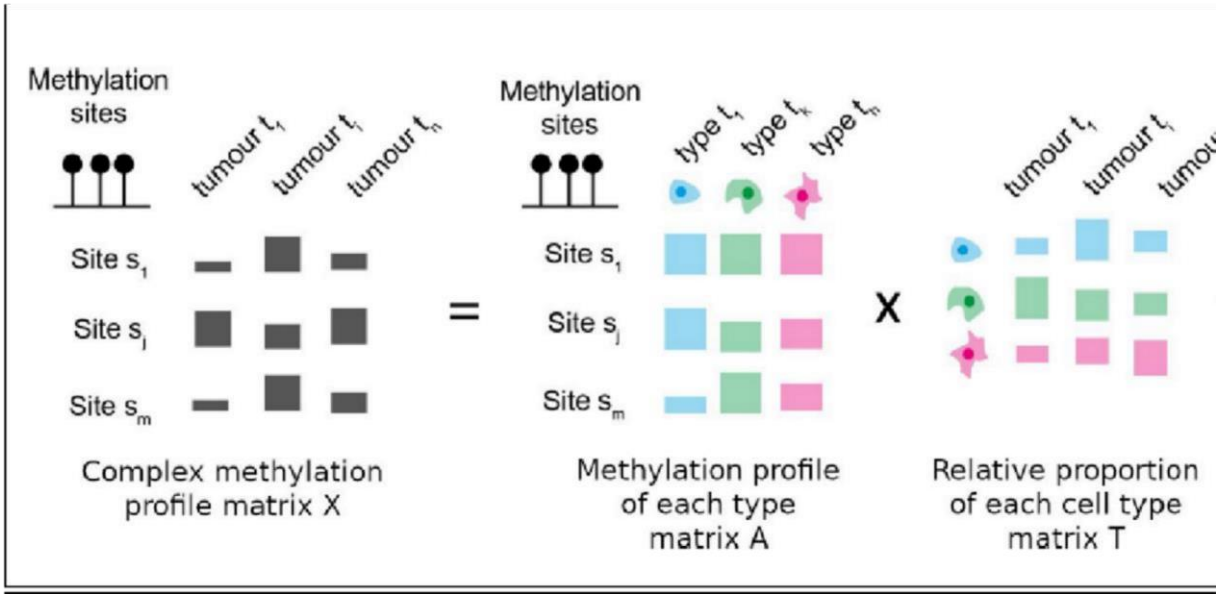


Figure 5 : les decomposion de la matrice complexe en un produit de deux matrice plus petites

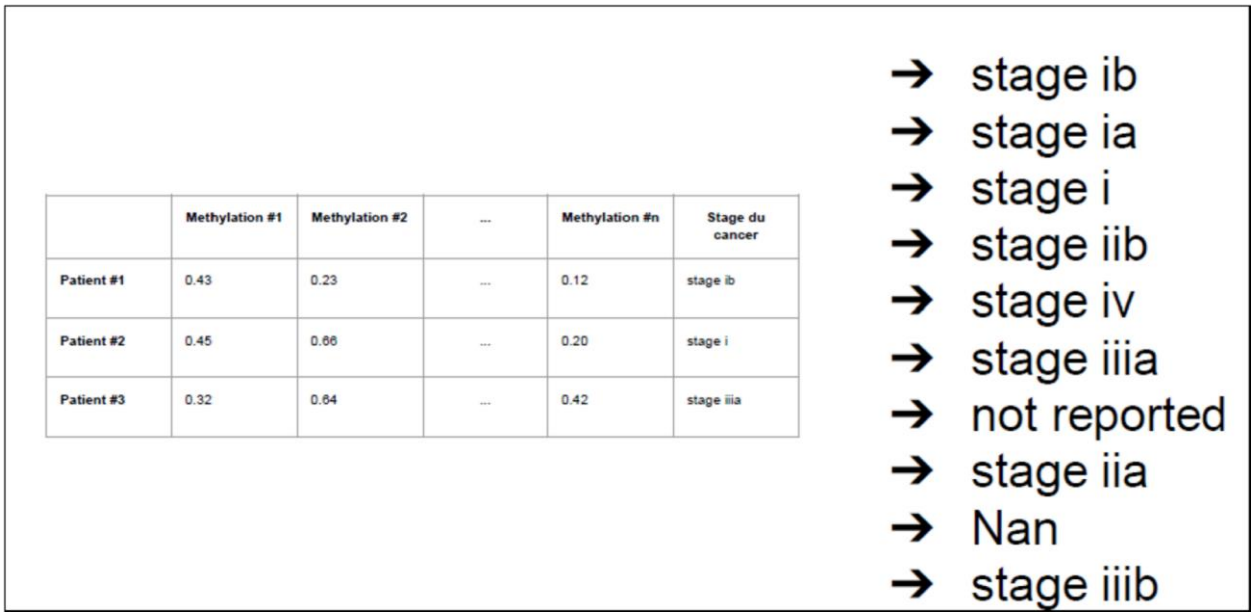


Figure 6 : classification des stades du cancer parmi une population spécifique dans l'une des 10 catégories

4. Conclusion :

Pour conclure, rappelons-nous que l'ambition que nous avions avec ce projet était ,à l'aide de l'intelligence artificielle et de l'informatique, de montrer qu'une guérison du cancer par une autre voie qui pourrait paraître peu conventionnelle était possible. Nous avons pu voir comment à l'aide d'un « simple fichier de programmation », il était possible d'aider les médecins dans leur quotidien lors de l'analyse des radios des tumeurs. Par exemple, nous pouvons prendre l'exemple des chercheurs qui ont mis leur IA à l'épreuve contre une équipe de quinze médecins experts lors d'un test de diagnostics de cancers ;les IA avaient alors gagné. Plus que des connaissances accumulés, ce projet nous a permis de travailler réellement sous forme de groupes de travail autour d'un « gros » projet avec une deadline donnée, ce que nous avons rarement fait dans notre scolarité .

En effet, le travail de groupe implique plus de facteurs que ce qu'il n'y parait :coordination des emplois du temps, communication des résultats entre les membres, explication de chacune de nos parties aux autres …

Pour les étudiants qui s'intéresseront à ce projet, ce projet vous montre bien que la branche de l'intelligence artificielle est une voie qui ouvre des possibilités dans plein de domaines d'application différents (biologie, informatique, sociologie) tout en restant intéressant.

Page BONUS :

En apprentissage machine, nous ne pouvions pas adapter le modèle aux données d'apprentissage et ne pouvons pas dire que le modèle fonctionnera avec précision pour les données réelles. Pour cela, il faut s'assurer que le modèle a obtenu les modèles corrects à partir des données et qu'il ne génère pas trop de bruit. Pour cela, nous utilisons la technique de validation croisée.

La **validation croisée** est une technique dans laquelle nous formons notre modèle à l'aide du sous-ensemble de l'ensemble de données, puis nous évaluons à l'aide du sous-ensemble complémentaire de l'ensemble de données.

Supposons qu'on possédeé 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 : on parle parfois de **surapprentissage.**

L'Overfitting:Ce phénomène de « surapprentissage » dégrade la performance des algorithmes de machine learning)

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

Bibliographie :

[1] : https://seaborn.pydata.org/generated/seaborn.pairplot.html

[2] : https://pypi.org/project/dataIO/

[3] : https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[4] : https://matplotlib.org/

[5] https://scikit-learn.org/stable/?fbclid=IwAR1FJA5H41muIe3RmGbcXcF8GLalw47qLpkSWacASkFb7jF244VekW
IBiA

[6] https://fr.wikipedia.org/wiki/D%C3%A9composition_en_valeurs_singuli%C3%A8res?fbc l id=IwAR3DLFMio6gjmJu1CxVZgHIMfGWAQvBtjKe3a07zoqNgRERhlunY8-ONKGw

[7] https://www.geeksforgeeks.org/python-pandas-dataframe-series-head-method/

[8] https://fr.wikipedia.org/wiki/D%C3%A9composition_en_valeurs_singuli%C3%A8res_&

[9] https://scikit-learn.org/stable/modules/decomposition.html#decompositions

[10] https://fr.wikipedia.org/wiki/R%C3%A9duction_de_la_dimensionnalit%C3%A9

[11] https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels

[12] https://slideplayer.fr/slide/2463951/

[13] https://scikit-learn.org/stable/modules/decomposition.html#decompositions