Development project in Machine Learning – TAF MCE

Louis Faul, Eric Xavier N'guessan, Amine Ouasfi, Nicolas Stucki

05/12/2019

INTRODUCTION

Dans le domaine du *machine learning* et des statistiques, la classification automatique a toujours été un des sujets les plus abordés. L'énoncé du problème est simple: identifier à quelle catégorie d'un ensemble de catégories (sous-populations) appartient une nouvelle observation, en se basant sur un ensemble de données de formation *(training set)* contenant des observations (ou des instances) dont l'appartenance à une catégorie est connue.

Les domaines d'application sont nombreux : Séparation des emails (*spam* et non *spam*), fournir automatiquement le diagnosis d'un patient en fonction de ses symptomes, ... Quelles sont donc les différentes solutions trouvées à ce problème ? Quelles sont leurs limites et comment sont-elles implémentées ?

Dans le cadre de l'UE **Introduction au Machine Learning (IML)** il nous a été donné comme projet d'étudier deux *dataset* de *classification binaire* auxquels nous devions appliquer les différents modèles d'apprentissage automatique pour en tirer les meilleurs conclusions. Ce document sert de rapport à notre étude.

Nous avons structuré notre étude comme suit: en premier lieu, nous allons présenter les différents *datasets* fournis. Ensuite, nous allons présenter notre approche, du prétraitement des données à l'entraînement de nos modèles jusqu'au test. Enfin, nous terminerons par présenter quelques bonnes pratiques au sujet de la programmation.

1 Présentation des données

1. **Banknote Authentication Dataset** [1] :
   Il s'agit de données extraites à partir d'images prises pour l'évaluation d'une procédure d'authentification des billets de banque. Chaque individu est identifié par 4 caractéristiques.

2. **Chronic Kidney Disease Dataset**[2] :
   Ce *dataset* est issu d'une étude de deux mois en Inde, sur des individus souffrant ou non d'une *maladie rénale chronique (chronic kidney disease)*. Ce tableau recense une étude de 400 individus autour de 25 caractéristiques (le nombre de globules rouges, le nombre de globules blancs,...) avant de fournir le diagnostic : Ayant une maladie rénale chronique **('ckd')** ou n'en ayant pas **('notckd')**.

2 Traitement des données

2.1 Prétraitement

Le prétraitement des données que nous avons effectué consiste , dans un premier temps, à remplacer les valeurs nominales (ex.: "Normal", "Anormal") par des valeurs numériques binaires (0 ou 1) car les nombres sont plus faciles à interpréter. Dans un second temps, nous avons traité le cas des valeurs manquantes par individus. On peut décider de les supprimer, mais dans notre cas nous avons décidé de les remplacer par la moyenne des éléments de la colonne.
Nous avions deux *datasets*:

- le premier *(Banknote Authentication Dataset)* ne comportait ni valeurs nominales ni valeurs manquantes et de fait le prétraitement n'était pas nécessaire;
- le second *(Chronic Kidney Disease Dataset)* comportait des valeurs nominales et des éléments manquants par individus. Nous avons donc appliqués un prétraitement sur ces données qui ont permis d'avoir un autre jeu de données traitées (voir figure1 et 2). Nous avons effectué ce traitement à l'aide d'une fonction que nous avons implémenté dans le fichier *preprocessing.py*.

2.2 Reduction des dimensions

Nous avons utilisé l'algorithme principal component analysis (PCA). Cet algorithme nous permet de reduire le nombre de variables, ce qui est très utile pour le second dataset, car il est composé de 25 caracteristiques. La première fonction permet d' afficher

[1] https://archive.ics.uci.edu/ml/datasets/banknote+authentication
[2] https://www.kaggle.com/mansoordaku/ckdisease

le nombre optimal de composants à garder par notre algorithme. Lorsque nous appliquons cette fonction au second dataset, 80 pourcent de la variance est expliqué pour au minimum 11 composants (voir figure annexe). Nous avons donc réduit notre dataframe à 11 caracteristiques. Pour le 1er dataset, nous avons gardé les quatres composantes de depart. La seconde fonction nous permet d' afficher les données selon les deux premieres composantes de notre algorithme. Il se trouve que les données sont preque séparé, en fonction de leur classe (voir figure annexe). Nous pouvons donc anticiper de bons résultats de précision sur nos modèles futures.

2.3 Séparation des données

Après cette phase de prétraitement, nous avons séparé les données entre des données d'entrainement, qui serviront à entrainer les modèles implémentés par la suite, et des données de test, qui ont pour but de vérifier la qualité des méthodes utilisés en prédisant la classe d'appartenance .

Nous avons utilisé des coefficients classiques de séparation, en prenant 70 % des données pour l'entrainement et 30% pour le test. En absence de validation set cette séparation des données expose au risque de surapprentissage (*overfitting* car on peut, en entraînant que sur les données d'entraînement continuer à ajuster nos hyperparamètres jusqu'à ce que notre modèle colle parfaitement aussi bien aux données d'entraînement qu'à ceux du test. Nous avons donc implémenté une méthode de validation croisée, afin d'entrainer constammment les modèles sur des jeux de données différents ce qui permet de limiter les risques d'overfitting et permet d'ajuster les méthodes utilisées.

2.4 Implémentation des modèles

2.4.1 Approche générale

Pour tous les modèles que nous avons utilisés nous avons mis en oeuvre les mêmes étapes de développement à savoir :

- Entrainement du modèle avec des paramètres de base, choisis par défaut sur les données, puis prédiction.
- Une phase de validation croisée afin d'évaluer l'apprentissage du modèle. Il s'agissait à chaque fois de 10 validations croisées.
- L'application d'une fonction GridSearchCV qui parcourt l'ensemble des hyperparamètres qu'on lui donne en entrée et retourne les meilleurs hyperparamètres pour le modèle donné.
- Puis on refait une phase d'entraînement et de test avec le modèle ayant ses hyperparamètres optimisés.
- A chaque étape bien sûr on regarde l'écart entre les prédictions effectués par le modèles et les véritables classes d'appartenance des échantillons. Plusieurs indicateurs de résultat peuvent être analyser dont la matrice de confusion et la courbe RoC. De plus nous verrons que ces résultats sont très bons pour la majorité des classifieurs et ce sur les deux bases de données, ainsi nous avons décidé de comparer les temps d'éxécution des différents algorithmes afin de pouvoir les différencier

Nous allons prendre un jeu de données après l'autre et analyser les résultats de nos algorithmes sur ces données

2.4.2 Les modèles implémentés

2.4.2.1 SVM

Le premier modèle que nous allons aborder est le modèle SVM. Il s'agit d'une généralisation des classifieurs linéaires, qui a été rapidement adaptés pour son faible nombre d'hyperparamètres et les bons résultats obtenus. Dans notre cas on utilise un modèle à marge souple, qui est matérialisé par la constante C qui permet de contrôler le compromis entre nombre d'erreurs de classement, et la largeur de la marge.

2.4.2.2 Decision tree

Il s'agit d'un algorithme qui effectue un ensemble de choix sous la forme graphique d'un arbre. C'est une méthode d'apprentissage supervisé non paramétrique utilisée pour la classification et la régression. L'objectif est de créer un modèle qui prédit la valeur d'une variable cible en apprenant des règles de décision simples déduites des caractéristiques des données. Les principaux paramètres que nous utilisons dans cette implémentation sont :

- **Criterion** :Une fonction qui mesure la qualité de la séparation d'un noeud. Nous utilisons par défaut l'indice d'impureté de *gini* qui mesure de la fréquence à laquelle un élément choisi au hasard de l'ensemble serait incorrectement étiqueté s'il était étiqueté au hasard selon la distribution des étiquettes dans le sous-ensemble.
- **max depth** :La profondeur maximale de l'arbre. Par défaut, elle vaut 3.
- **min samples leaf**: Le nombre minimum d'échantillons requis pour fractionner un nœud interne. Par défaut, nous l'avons fixé à 5.

2.4.2.3 Random Forest

L'algorithme Random Forest s'appuie sur le concept de bagging, en effet il effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents. Les hyperparamètres principaux sont le nombre d'arbres la profondeur des arbres et la fonction qui mesure la qualité de séparation.

2.4.2.4 Multilayer perceptron

Il s'agit d'une succession de couches de neurones entièrement connéctées. C'est à dire que les neurones d'une couche sont reliés à la totalité des neurones des couches adjacentes à travers des poids et suivis d'une fonction d'activation non linéaire.

Nous avons implémenté une version améliorée du Multilayer perceptron. Il s'agit d'une succesion de blocks linéaires suivis de fonctions d'activations non linéaires.

- Chaque block linéaire correspond à une BachNormalisation, un Dropout et une couche complètement connéctée.
- Les fonctions d'activation des différents blocks peuvent être définies en paramètre à l'exception de la couche de sortie qui correspond à une fonction **sigmoid**.

- La largeur de chaque couche, c'est à dire son nombre d'unités, ainsi que pronfondeur du réseau peuvent être définies dans le paramètre *units*. Dans le cas où ce paramètre est fixé à *None*, le modèle correspond à une regression logistique.

3 Chronic kidney disease

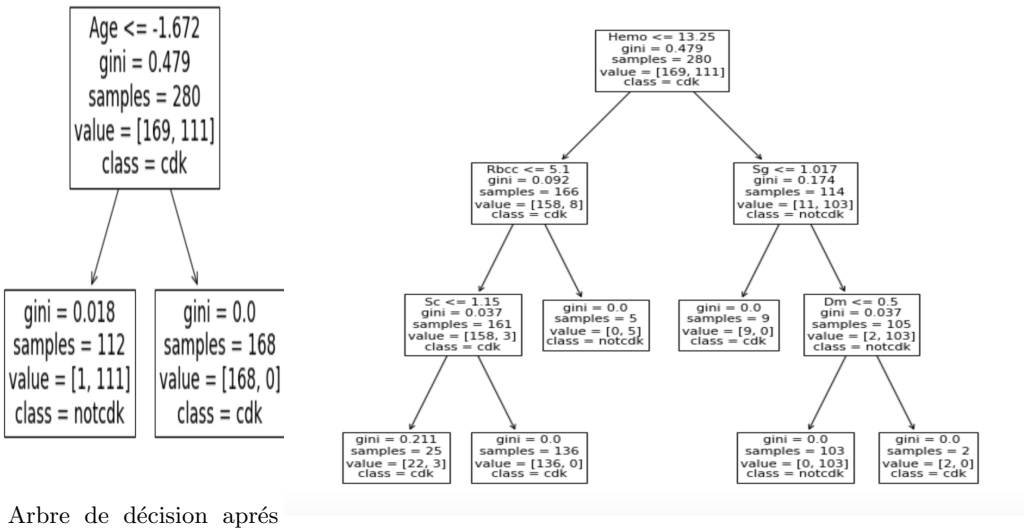
3.1 SVM

Tout d'abord dans le cas des paramètres par défaut, c'est à dire C=1 et un noyau gaussien d'ordre 3,et dans le cas où l'on travaille sur l'ensemble des features, les résultats sont visibles en annexe sur la figure 6.
On remarque que tous les individus sont classés comme positifs, ce qui bien sûr n'est pas satisfaisant. On applique ensuite l'algorithme GridSearchCv. On donne en entrée de cete fonction deux possilités de noyaux, soit un noyau linéaire, soit un noyau gaussien.L'algorithme nous donne que le noyau linéaire est plus performant et les meilleurs paramètres sont $= 1$ et de $C = 0.1$. Avec ses valeurs on obtient les résultats visible sur la figure 7
Les résultats sont très satisfaisants mais le temps d'éxécution de la fonction GridSearchCV est assez long dû au nombre important de combinaisons d'hyperparamètres à tester.

3.2 Decision tree

Avec les paramètres par défaut, nous obtenons une *accuracy* moyenne sur les validation croisées de 0.97. La performance sur le données de test est légèrement plus basse comme le montre la figure 8.

L'optimisation des hyperparamètres permet d'obtenir une *accuracy* moyenne sur les validations croisées un peu plus élévée : 0.9875. Mais le modèle à du mal à généraliser. Sa performance sur les données de test ne change pas. Avec la réduction de dimension par PCA, l'*accuracy* aprés GridSearchCv vaut 1. Le modèle dans ce cas est plus simple : une profondeur maximale et un nombre minimal d'échantillons requis par fraction qui tous deux valent 1.



(a) Arbre de décision aprés la réduction de dimension par PCA

(b) Arbre de décision sur l'ensemble des caractéristiques

Figure 1: Arbres de décisions pour le premier dataset.

Dans la figure 1a, il ne s'agit pas exactement l'age comme la caractéristique détérminente de la classification mais plutôt de la première colonne des données aprés la réduction de dimension.

3.3 Random Forest

On prend Gini comme fonction par défaut sur 10 de longueur maximale 2.Les résultats sur la base de données "Chronic kidney disease" sont déja assez satisfaisants comme indiqué sur la figure 10.

Comme précedemment on va ensuite chercher à optimiser les hyperparamètres du modèle, ce qui donne 11 arbres décisonnels avec 6 comme profondeur maximale, et les résultats sont présents en figure 11.

3.4 Multilayer percepton

Avec les paramètres par défaut nous obtenons une *accuracy* moyenne sur les validation croisées de 1. Il en est de même sur les données de test. Par ailleurs, dans ce cas les paramétres par défaut correspondent aux paramètres optimaux avec un batch size qui vaut 70. Neanmoins, on observe une forte instabilité de la performance sur le test set comme le montre la figure 2. Cette instabilité est encore plus prononcée qu'on applique une PCA sur les données.

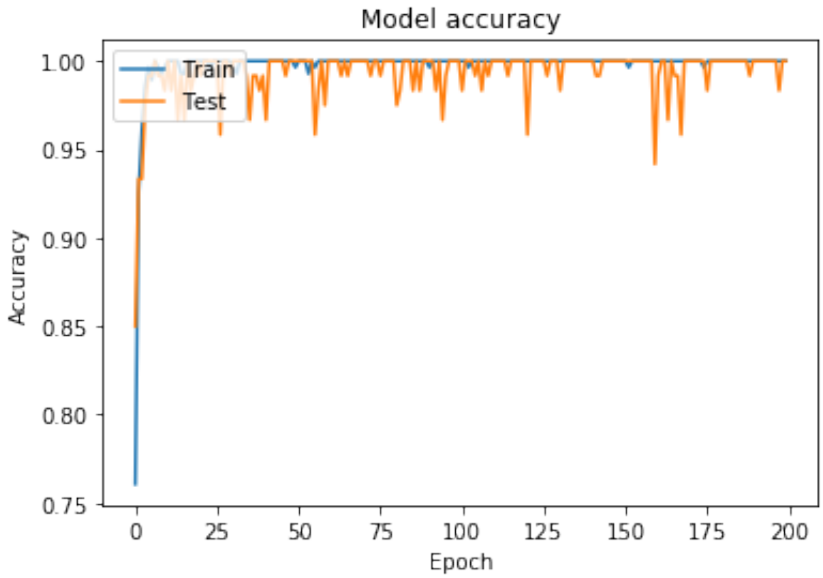


Figure 2: Accuracy du model sur les données d'entrainement et de test en fonction du nombre d'iterations d'entraînement.

4 Data banknote authentication

Nous allons maintenant appliquer l'ensemble de nos algorithmes sur l'autre jeu de données "Data banknote authenticaton". Retrouvent t-on les mêmes algorithmes avec les meilleurs résultats ?

4.1 SVM

On applique à nouveau un SVM avec des paramètres de base sur nos données. Les résultats sont présentés en annexe figure 12.
Les résultats sont déja excellents puisque tous les échantillons sont correctements classifiés. Il n'est ici pas nécessaire d'appliquer une recherche des meilleurs hyperparamètres.

4.2 Decision tree

Avec les paramètres par défaut nous obtenons une *accuracy* moyenne sur les validations croisées de 0.923. La performance sur les données de test est légèrement plus basse (91.75).

L'optimisation des hyperparamètres permet d'obtenir une *accuracy* moyenne sur les validations croisées plus élévée : 0.982. Et la performance sur les données de test est de 99.27. Dans ce cas, on voit que le modèle se géneralise bien aux données non observées.

4.3 Random Forest

L'algorithme de Random Forest appliqué avec des paramètres standard donne lui les résultats de la figure 13.
La précision et le f1-score sont tous deux autour de 93 % . Essayons d'optimiser les hyper-paramètres de ce modèle afin d'améliorer nos résultats, cela donne les résultats de la figure 14.

4.4 Multilayer perceptron

Avec les paramètres par défaut, l'*accuracy* moyenne sur les validations croisées est de 1. Il en est de même sur les données de test. L'évolution de l'accuracy en fonction des iterations est plus stable cette fois sur le validation set et converge rapidement (25 iterations) comme le montre la figure 3.

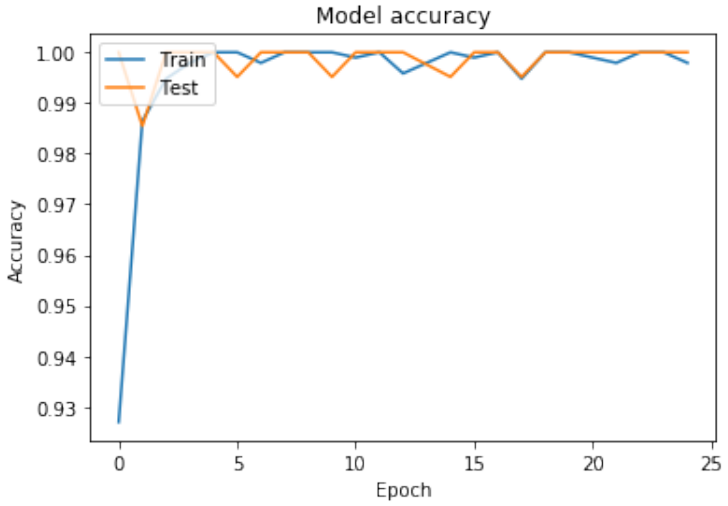


Figure 3: Accuracy du model sur les données d'entrainement et de test.

5 Conclusion

Après l'optimisation des hyperparamètres, on obtient souvent une justesse (*accuracy*) entre 0,98 et 1. S'il fallait choisir un modèle parmis les modèles les plus performants sur chaque jeu de données nous pouvons faire appel à un principe de parcimonie (principe du Rasoir d'Ockham [3]) et choisir ainsi les modèles les plus simples parmis les plus performants. Nous obtenons dans ce cas les modèles suivants :

- Pour les données chronic kidney disease:

[3]principe de raisonnement dit de parcimonie , ou d'économie , antérieur au Franciscain Guillaume d'Occam mais énoncé par lui au 14ème siècle. En substance : Pluralitas non est ponenda sine necessitate.

1. Une normalisation des données d'entrées.
2. Une réduction de dimension par un Principal Component Analysis à 11 dimensions aprés normalisation.
3. Une classification par Arbre de decision avec comme hyperparamètres : (criterion : gini, max depth : 1, min samples leaf : 1).

- Pour les données banknote authentication : Un SVM avec les paramètres par défaut c'est à dire avec un noyau gaussien de degree 3.

6 Bonnes pratiques de programmation

Tout au long du projet, nous avons eu à travailler en collaboration. Nous allons donc donner ci-dessous ce qui semble à notre avis les bonnes pratiques de programmation à avoir lors de la réalisation d'un projet informatique :

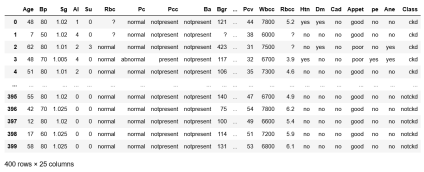
- **Structurer le travail et bien le répartir entre les membres**
- **Commenter chaque partie du code** afin que tous les membres puissent comprendre ce que chacun a effectué. Du fait du travail en équipe et pour assurer un gain de temps considérable, il est nécessaire que tout ceux qui ont accès au code pour y travailler ou le corriger puisse comprendre ce que chaque fonction et/ou classe effectue comme travail pour pouvoir les utiliser ou même les améliorer. Pour cela, chaque membre doit commenter la partie du code qu'il a rédigé pour améliorer la compréhension globale du programme.
- **Ajouter chaque fois des messages à ces commits** pour un suivi du projet plus simple.
- **Garder toujours une version antérieure stable** du programme pour s'assurer que le projet continue meme en cas de problèmes en local.
- **Lorsqu'au cours du travail vous êtes amenés à tester des éléments qui pourrait ne pas fonctionner, effectuez le sur une *branche Git*** afin d'éviter de créer des erreurs sur le projet principal.

CONCLUSION GENERALE

Le projet de l'UE *Introduction au Machine Learning* nous a permis de découvrir certains algorithmes utilisées couramment en apprentissage automatique à des fins d'entrainement et de prédiction. En outre, il nous a aussi permis d'apprendre le travail collaboratif en informatique, effectué de nos jours à l'aide des logiciels de *versioning* (**Git** dans notre cas). Cette connaissance nous sera assurément très utile pendant notre parcours d'ingénieur.

Appendices

Les données



	Age	Bp	Sg	Al	Su	Rbc	Pc	Pcc	Ba	Bgr	...	Pcv	Wbcc	Rbcc	Htn	Dm	Cad	Appet	pe	Ane	Class
0	48	80	1.02	1	0	?	normal	notpresent	notpresent	121	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7	50	1.02	4	0	?	normal	notpresent	notpresent	?	...	38	6000	?	no	no	no	good	no	no	ckd
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	...	31	7500	?	no	yes	no	poor	no	yes	ckd
3	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	...	35	7300	4.6	no	no	no	good	no	no	ckd
...																					
395	55	80	1.02	0	0	normal	normal	notpresent	notpresent	140	...	47	6700	4.9	no	no	no	good	no	no	notckd
396	42	70	1.025	0	0	normal	normal	notpresent	notpresent	75	...	54	7800	6.2	no	no	no	good	no	no	notckd
397	12	80	1.02	0	0	normal	normal	notpresent	notpresent	100	...	49	6600	5.4	no	no	no	good	no	no	notckd
398	17	60	1.025	0	0	normal	normal	notpresent	notpresent	114	...	51	7200	5.9	no	no	no	good	no	no	notckd
399	58	80	1.025	0	0	normal	normal	notpresent	notpresent	131	...	53	6800	6.1	no	no	no	good	no	no	notckd

400 rows × 25 columns

Figure 4: Données non traitées (jeu de données ckd)

	Age	Bp	Sg	Al	Su	Rbc	Pc	Pcc	Ba	Bgr	...	Pcv	Wbcc	Rbcc	Htn	Dm	Cad	Appet	pe	Ane	Class
0	48.0	80.0	1.020	1.0	0.0	0.81	1.0	0.0	0.0	121.00	...	44.0	7800.0	5.20	1.0	1	0.0	0.0	0.0	0.0	0
1	7.0	50.0	1.020	4.0	0.0	0.81	1.0	0.0	0.0	148.04	...	38.0	6000.0	4.71	0.0	0	0.0	0.0	0.0	0.0	0
2	62.0	80.0	1.010	2.0	3.0	1.00	1.0	0.0	0.0	423.00	...	31.0	7500.0	4.71	0.0	1	0.0	1.0	0.0	1.0	0
3	48.0	70.0	1.005	4.0	0.0	1.00	0.0	1.0	0.0	117.00	...	32.0	6700.0	3.90	1.0	0	0.0	1.0	1.0	1.0	0
4	51.0	80.0	1.010	2.0	0.0	1.00	1.0	0.0	0.0	106.00	...	35.0	7300.0	4.60	0.0	0	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	1.00	1.0	0.0	0.0	140.00	...	47.0	6700.0	4.90	0.0	0	0.0	0.0	0.0	0.0	1
396	42.0	70.0	1.025	0.0	0.0	1.00	1.0	0.0	0.0	75.00	...	54.0	7800.0	6.20	0.0	0	0.0	0.0	0.0	0.0	1
397	12.0	80.0	1.020	0.0	0.0	1.00	1.0	0.0	0.0	100.00	...	49.0	6600.0	5.40	0.0	0	0.0	0.0	0.0	0.0	1
398	17.0	60.0	1.025	0.0	0.0	1.00	1.0	0.0	0.0	114.00	...	51.0	7200.0	5.90	0.0	0	0.0	0.0	0.0	0.0	1
399	58.0	80.0	1.025	0.0	0.0	1.00	1.0	0.0	0.0	131.00	...	53.0	6800.0	6.10	0.0	0	0.0	0.0	0.0	0.0	1

400 rows × 25 columns

Figure 5: Données traitées (jeu de données ckd)

Les résultats

```
Confusion Matrix:
 [[81  0]
 [39  0]]
Accuracy :  67.5
Report :
              precision    recall  f1-score   support

           0       0.68      1.00      0.81        81
           1       0.00      0.00      0.00        39

    accuracy                           0.68       120
   macro avg       0.34      0.50      0.40       120
weighted avg       0.46      0.68      0.54       120
```

Figure 6: Résultats obtenus pour un SVM avec des paramètres par défaut sur les données chronic kidney disease

```
les meilleurs paramètres sont  {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
les meilleur score est  0.8107142857142857
Confusion Matrix:
 [[73  8]
 [10 29]]
Accuracy :  85.0
Report :
              precision    recall  f1-score   support

           0       0.88      0.90      0.89        81
           1       0.78      0.74      0.76        39

    accuracy                           0.85       120
   macro avg       0.83      0.82      0.83       120
weighted avg       0.85      0.85      0.85       120
```

Figure 7: Résultats obtenus pour un SVM avec un noyau rbf et des paramètres optimisés sur les données chronic kidney disease

```
Confusion Matrix:
 [[79  2]
 [ 2 37]]
Accuracy :  96.66666666666667
Report :
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        81
           1       0.95      0.95      0.95        39

   micro avg       0.97      0.97      0.97       120
   macro avg       0.96      0.96      0.96       120
weighted avg       0.97      0.97      0.97       120
```

Figure 8: Résultat obtenus pour Decision tree avec les paramètres par défaut sur les données chronic kidney disease

```
les meilleurs paramètres sont  {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 1}
les meilleur score est  0.9857142857142858
Confusion Matrix:
 [[79  2]
 [ 2 37]]
Accuracy :  96.66666666666667
Report :
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        81
           1       0.95      0.95      0.95        39

   micro avg       0.97      0.97      0.97       120
   macro avg       0.96      0.96      0.96       120
weighted avg       0.97      0.97      0.97       120
```

Figure 9: Résultats obtenus pour Decision tree avec les paramètres optimisés sur les données chronic kidney disease

```
Confusion Matrix:
 [[78  3]
 [ 1 38]]
Accuracy :  96.66666666666667
Report :
              precision    recall  f1-score   support

           0       0.99      0.96      0.97        81
           1       0.93      0.97      0.95        39

    accuracy                           0.97       120
   macro avg       0.96      0.97      0.96       120
weighted avg       0.97      0.97      0.97       120
```

Figure 10: Résultats obtenus pour un Random Forest avec des paramètres par défaut sur les données chronic kidney disease

```
les meilleurs paramètres sont  {'max_depth': 6, 'n_estimators': 11, 'random_state': 42}
les meilleur score est  0.9928571428571429
Confusion Matrix:
 [[81  0]
 [ 0 39]]
Accuracy :  100.0
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        81
           1       1.00      1.00      1.00        39

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

Figure 11: Résultats obtenus pour un Random Forest avec des paramètres optimisés sur les données chronic kidney disease

```
Confusion Matrix:
 [[239   0]
 [  0 173]]
Accuracy :  100.0
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       239
           1       1.00      1.00      1.00       173

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412
```

Figure 12: Résultats obtenus pour un SVM avec des paramètres par défaut sur les données banknote authentication

```
Confusion Matrix:
 [[81  0]
 [39  0]]
Accuracy :  67.5
Report :
              precision    recall  f1-score   support

           0       0.68      1.00      0.81        81
           1       0.00      0.00      0.00        39

    accuracy                           0.68       120
   macro avg       0.34      0.50      0.40       120
weighted avg       0.46      0.68      0.54       120
```

Figure 13: Résultats obtenus pour un Random Forest avec des paramètres par défaut sur les données banknote

```
Confusion Matrix:
 [[81  0]
 [39  0]]
Accuracy :  67.5
Report :
              precision    recall  f1-score   support

           0       0.68      1.00      0.81        81
           1       0.00      0.00      0.00        39

    accuracy                           0.68       120
   macro avg       0.34      0.50      0.40       120
weighted avg       0.46      0.68      0.54       120
```

Figure 14: Résultats obtenus pour un Random Forest avec des paramètres optimisés sur les données banknote

```
Confusion Matrix:
 [[219  20]
 [ 14 159]]
Accuracy :  91.74757281553399
Report :
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       239
           1       0.89      0.92      0.90       173

   micro avg       0.92      0.92      0.92       412
   macro avg       0.91      0.92      0.92       412
weighted avg       0.92      0.92      0.92       412
```

Figure 15: Résultats obtenus pour Decision tree avec des paramètres par défaut sur les données banknote authentication

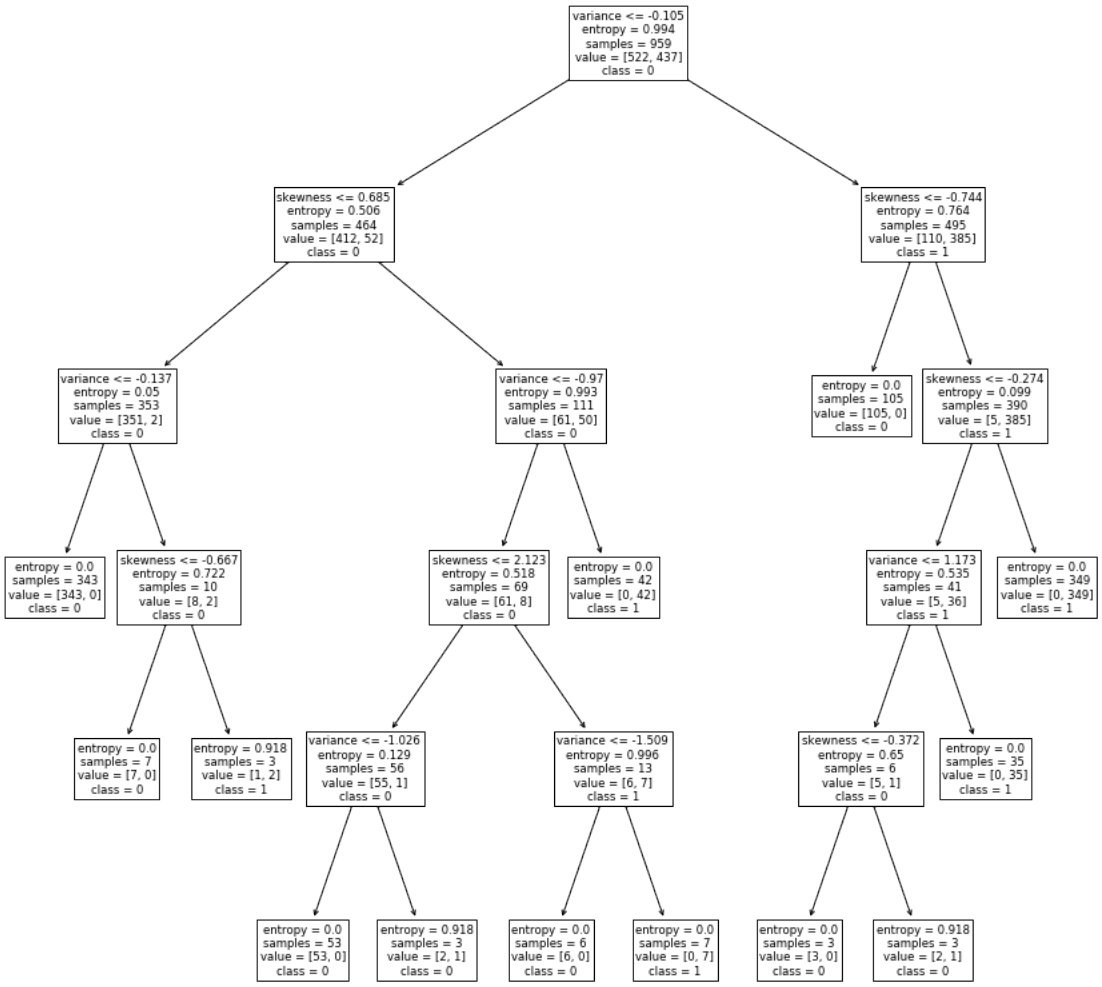


Figure 16: **Arbre de décision sur les données banknote authentication aprés une réduction à deux dimensions**: Ici, les hyperparamètres utilisés sont ceux obtenus aprés optimisation. Dans ce cas, la mesure de la qualité du fractionnement est effectué par l'entropie, la profondeur maximale vaut 5 et le nombre minimale d'echantillons par feuille vaut 3.