Projet Machine Learning

Axel Rubio, Sofiane Bouibeb et Hedi Sagar

***Abstract*— Les méthodes de classification sont très développées de nos jours et il existe une grande variété d'algorithmes qui permettent de trouver des liens, des patterns entre les différentes features composant les dataset, dans le but ensuite de pouvoir regrouper différents individus comme appartenant à une même classe. Dans ce projet, l'accent est mis sur l'apprentissage supervisé afin de pouvoir exploiter les labels binaires de deux dataset "Chronic Kidney Disease" et "Banknote Authentication Dataset". Après un pré processing minutieux permettant d'extraire toutes irrégularités et incohérence dans les données, et de mettre en forme certains features (encodage), les différents algorithmes testés s'avèrent être très précis, d'un point de vu du recall et de l'accuracy. Dans un second temps nous abordons la feature selection sur le dataset "Chronic Kidney Disease" comportant 25 features par échantillon. Cette analyse montre la présence de 3 features majoritaires permettant à eux seuls d'obtenir la même précision qu'avec l'ensemble des 25. Nous abordons enfin les bonnes pratiques à avoir en tête lors de la programmation pour un projet de la sorte.**

I. INTRODUCTION

L'objectif de ce projet est de résoudre un problème de classification binaire de données à partir d'un dataset brut, dans un environnement collaboratif en mettant en oeuvre de bonnes pratiques de programmation permettant un suivi et une prise en main facile pour tout utilisateur extérieur. Nous avons fait le choix de l'utilisation de GitLab afin de sauvegarder les différentes versions d'avancement de notre code. L'accent a été mis sur la construction d'un script python faisant office de librairie pouvant être appliquée aux deux codes, même si la partie preprocessing comportes quelques spécificités propres à chaque dataset comme nous le verrons par la suite.

Les deux dataset sur lesquels se portent notre étude sont d'une part le "Chronic Kidney Disease"[1], et d'autre part le "Banknote Authentication Dataset"[2]. Nous avons utiliser 5 algorithmes différents afin de mener à bien notre classification : Logistic Rgression, Linear SVC, Random Forest, Gradient Boosting et un Réseau de Neurones utilisant tensorflow.

II. PRESENTATION DES DATASET

A. Chronic Kidney Disease

id	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no	ckd

Fig. 1: Extrait du dataset brute Chronic Kidney Disease

Ce premier dataset une base de données médicale rassemblant 24 caractéristiques médicales de 400 patients (âge, albumine, pression sanguine, taux de sucre, hypertension) ainsi qu'un label pour indiquer s'ils sont ou non diagnostiqués positifs à la maladie rénale chronique ("ckd" si positif, "notckd" sinon). Les valeurs des différentes caractéristiques peuvent être des réels, des entiers ou encore des chaînes de caractères. Ce dataset est disponible au format ".csv" ce qui va rendre facile sont extration via pandas.

B. Banknote Authentication Dataset

	0	1	2	3	4
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Fig. 2: Extrait du dataset brute Banknote Authentication

Ce second dataset est rassemble 4 caractéristiques de 1372 billets de banque qui ont été extraites d'images prises lors de procédures d'authentification des billets (variance, skewness, curtosis et entropy). La cinquième colonne correspond au label de chaque billet. S'il vaut 0, il s'agit d'un vrai billet, et s'il vaut 1 c'est donc un faux. Ici, les features sont tous des float, il n'y a pas de chaine de caractères, ce qui va rendre le preprocessing beaucoup plus simple. Enfin ce dataset est disponible au format ".txt", que nous transformerons au format ".csv" pour pouvoir traiter avec pandas en utilisant le même code python.

III. PREPROCESSING

A. Données Incohérentes

La première étape essentielle afin d'optimiser les chances de bonne classification de notre algorithme est la mise en place d'un algortithme de preprocessing efficace. Il va de soit qu'il doit être adapté à chaque dataset, et repose donc sur des observations en amont des différents types de données. Pour le Chronic Kidney Dataset, nous avons fait particulièrement attention aux chaines de caractères, véritable point sensible lié à des erreurs de typographie plus difficiles à détecter. Pour ce faire, nous avons commencé par étudier chaque feature un par un en se referant à la documentation du dataset et pouvoir comparer les différentes valeurs attendues. En utilisant une méthode très utile de pandas "value_counts()" nous avons pu identifier les valeurs érronées. Par exemple sur la colonne "classification" ci-dessous, on peut identifier une incohérence avec la présence de valeurs ckdt. Nous avons ainsi pu détecter les différentes icohérences dans plusieurs features. ("tno", "tyes" et " yes", repérés dans la classe

```
import pandas as pd
kidney = pd.read_csv(data_kidney)
kidney["classification"].value_counts()
```
✓ 0.0s
```
ckd        248
notckd     150
ckd\t        2
Name: classification, dtype: int64
```

Fig. 3: Données Incohérentes

"dm" par exemple). Une fois toutes les erreurs détectées, l'algorithme de preprocessing a été ajusté afin de lescorriger.

B. Données manquantes

Un autre soucis rencontré lors de la phase d'analyse de nos dataset a été la présence de données manquantes "Nan". A noter que ces manques sont uniquement présents sur le dataset Chornic Kidney Disease. La plupart des manques correspondant à des features dont les éléments sont des strings. Notre choix s'est donc porté sur la transformation de tous les features de type chaine de caractère qui ont été corrigés précedemment en groupes représentés par des entiers. Cette opération appelée encodage a été effectuée à l'aide d'un encordeur "LabelEncoder()" de scikit-learn. Ainsi pour des features ou il y aurait 2 possibilités "yes" ou "no", et des valeurs manquantes "nan", l'encoder retournerait 3 classes (0, 1, 2). L'utilisation d'encodeur a deux avantages principaux ici. D'une part, on n'a pas a choisir la valeur de manière arbitraire pour compléter la valeur manquante, ce qui pourrait influencer de manière négative la classification par la suite. Et d'autre part ca nous permet de garder un nombre conséquent de données, plutot que de supprimer toutes les lignes ou il y a des valeurs manquantes, étant donné qu'il y en a beaucoup dans ce dataset.

C. Normalisation

Enfin, pour le reste des features dont les valeurs sont des entiers ou des réels, nous avons tout converti en réel afin de pouvoir généraliser le processus de normalisation des données (moyenne nulle et variance unité). A noter que notre algorithme de preprocessing étant le même pour les deux dataset, seule la partie de normalisation des données sera effectuéé sur le dataset Banknote Authentication, puisqu'il ne contient pas de features dont les éléments sont des chianes de caractères.

IV. Processing

Une fois la phase de preprocessing effectuée, notre travail se poursuit avec la phase d'entrainement de nos algorithmes avant de pouvoir comparer leurs efficacités.

A. Pré découpage des datasets

Dans un premier temps, il est primordial de découper notre dataset en un training set et un test set distinctifs. Cela est nécessaire pour assurer la capacité de généralisation de nos algorithmes. Le split choisi est un split classique 80/20.

B. Choix des Algorithmes

Les cinq modèles que nous avons décidé de comparer sont les suivants : Logistic Regression, Linear SVC, Random Forest, Gradient Boosting, et une architecture simplifiée de Neural Networks (MLP) que nous avons décidé d'implémenter sur tensorflow. Un focus sera fait par la suite sur ce dernier algorithme où il y a beaucoup plus de choses à définir.

V. Validation des modeles

Dans cette section, nous allons commencer par recontextualiser les deux problèmes de classification afin d'utiliser les métriques de classification les plus pertinents. Nous comparerons ensuite ces indicateurs pour évaluer la performance globale de nos algorithmes.

A. Contextualisation et Métriques Retenues

Les deux problèmes de classification qui nous sont présentés sont relativement similaire dans leur objectif sous-jacent. Le dataset "Chronic Disease" a pour objectif d'aider à la détection de patients malades. Outre l'objectif d'avoir le meilleur nombre de bonnes classification, il semble donc particulièrement important de limiter l'erreur liée à une mauvaise classification de patients malades considérés comme sains. Naturellement, nous allons nous intéresser au Recall afin de minimiser le nombre de "False Negative". Il en est de même pour le second dataset avec les billets de banques où le plus important, outre le plus de bonne classification possible, est la minimisation du nombre de faux billets considérés comme vrais, c'est à dire la minimisation d'erreur de type 'False Negative".

En complement, étant donné que nos deux classes cibles sont relativement bien équilibré, nous utiliserons également l'accuracy pour statuer de la bonne classification en générale. En effet, l'utilisation de l'accuracy peut être non pertinent s'il y a un déséquilibre des classes dans le dataset. Ce point essentiel est important à soulever même s'il ne nous concerne pas ici.

En conclusion, nous utiliseront deux métriques : l'Accuracy et le Recall.

B. Analyse des Résultats

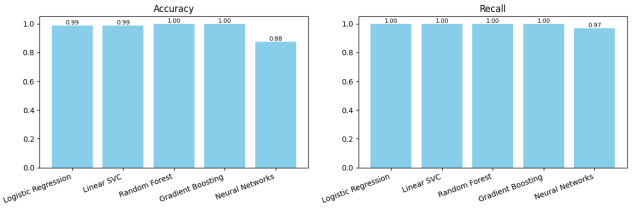

Fig. 4: Accuracy et Recall pour les 5 modèles sur le "Chronic Kidney Disease Dataset"

1) Chronic Kidney Disease Dataset: L'analyse des résultats des prédictions sur ce premier dataset nous montre que le recall est particulièrement élevé sur ce dataset, puisqu'on atteint des scores de 1 pour chacun

des algorithmes, sauf pour le réseau de neurone. Pour essayer de cacatériser la généralisation et les performances de ces algorithmes on s'intéresse également à l'accuracy. Il semblerait que tous les algorithmes soient également très performants avec des accuracy avoisinant les 1 ou 0.99. Gradient Boosting, Logistic Regression et Linear SVC semblent afficher de meilleurs pouventages d'accuracy. Enfin le réseau de neurones est le moins performant, avec "seulement" 88% d'accuracy. Mais cela est dû au choix de priorisation du recall comme métrique pour son entrainement.

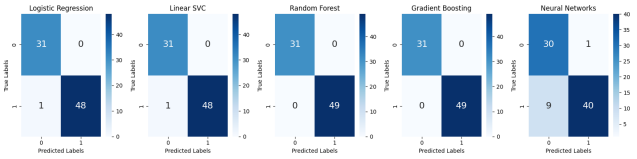

Fig. 5: Matrice de Confusion pour les 5 modèles sur le "Chronic Kidney Disease Dataset"

Il faut faire extrèmement attention ici. En effet, dans le dataset, les patients malades étant classés en premier, l'encodeur leur a donné le label 0. Ainsi dans le code vous pourrez remarquer une annotation de changement de labels à effectuer seulement pour le premier dataset au niveau de la matrice de confusion (labels=[1,0])! Ce point est éssentiel autrement l'analyse serait complètement éronnée., étant donnée que par convention, scikit learn assigne 0 au négatif et 1 au positif. Ainsi ici, on peut voir par exemple sur la matrice pour le Neural Network, qu'on a une erreur de type False Negative. (0 voulant dire que le patient est malade et 1 qu'il est sain). L'erreur False Negative se lit dans le carré en haut à droite!

2) Banknote Authentication dataset : Pour ce deuxième dataset, on peut observer que les recall sont également presque tous à 1, et les accuracy sont très élevées également. On atteint 1 avec le Linear SVC et 0.99 avec Random Forest et Gradient Boosting. Sur ce dataset, même le Réseau de neurones performes en affichant une accuracy de 0.98 et un recall de 1.

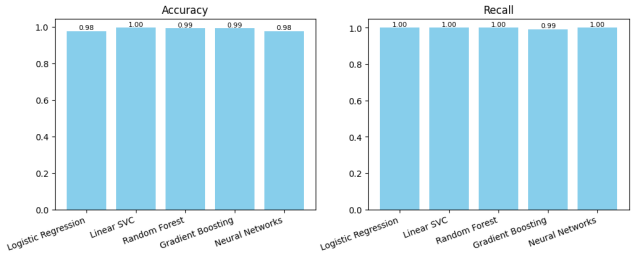

Fig. 6: Accuracy et Métriques pour le Banknote Authentication Dataset

Sur ce deuxième dataset, on peut voir que les recall et accuracy sont extrèmement élevés. Il semble plus facile à prédire que le premier, notamment par le Réseau de Neurones qui atteint 1 de recall et 0.98 d'accuracy. Attention ici à la différence de toute à l'heure, le 0 correspond bien au

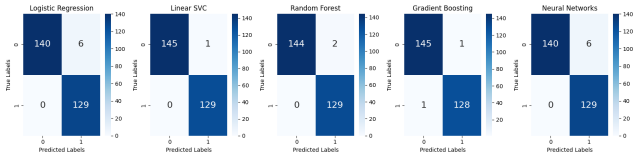

Fig. 7: Matrice de confusion pour les 5 modèles sur le Banknote Authentication Dataset

cas négatif le billet est vrai, pas d'anomalie. L'erreur False Negative se lit bien dans la case en bas à gauche.

C. Conclusion

Nous voyons que nos algorithmes nous permettent d'atteindre des niveaux de précisions très important, ce qui est très satisfaisant pour notre tâche et souligne l'efficacité du travail de nettoyage et le mise en forme des données effectué lors du pre processing. De plus, ces dataset semble relativement facile d'apprentissage pour nos algorithms, au vu de la précision des résultats.

VI. FEATURE SELECTION

Dans cette dernière partie, nous nous proposons d'étudier plus en profondeur les différents features présent dans nos datasets, et plus spécialmeent dans le "Chronic Kidney Disease" dataset qui possède en possède 24. L'objectif de cette partie va être d'observer l'influence et détecter les dépendances entre certains features afin de voir s'il est possible d'obtenir des résultats aussi bon en terme d'accuracy et de recall, mais avec beaucoup moins de features. En effet la dimensionalité des données est un paramètre très important dans tout projet de machine learning, et qui doit être optimisé du mieux possible pour éviter des risques sur de plus gros datasets.(Curse of dimentionality). On peut citer par exemple des problèmes d'overfitting, ou de complexité computationelle. Nous ne traiterons pas le Banknote dataset puisqu'il ne contient "que" 4 features, mais nous nous concentrerons sur les 24 features de l'autre dataset.

A. Analyse de Correlation entre features

L'idée est de construire une matrice de corrélation nous permettant de voir la dépendances entre features. Ici, on porte un intérêt tout particulier à considérer la class (malade ou non) dans la matrice. En effet, si certains features ont une très forte corrélation très forte avec la classe finale, alors ca veut dire qu'en se restreignant à eux seul, on peut espérer obtenir une très bonne accuracy, ou recall.

L'analyse de cette matrice fait ressortir directement deux features principales, la 15ème (indice 14) et la 3ème (indice 2) correspondant respectivement au taux d'hémoglobine et à la Specific Gravity .

B. Recursive Feature Elimination (RFE

Dans un deuxième temps, nous avons voulu tester une autre méthode de sélection de features par importance. A noter que cette méthode n'est pas applicable sur le Linear SVC ni sur le modèle tensorflow.

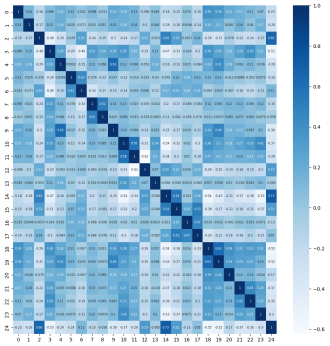

Fig. 8: Matrices de corrélation entre les 25 features comprenant la classe finale en 25 ème

```
[[13.  9.  1.  1. 10.  2. 21. 18. 12.  5. 14.  4.  8. 16.  1. 20. 22. 19.
   3.  7. 15.  6. 11. 17.]
 [11. 15.  1.  2. 14.  3. 13. 20. 21.  9. 10.  1.  6. 17.  1.  4. 12.  7.
   5.  8. 22. 18. 16. 19.]
 [ 3. 10.  1.  1. 15.  6. 20.  8. 12.  9.  5.  2.  4. 18.  1. 11.  7. 14.
  19. 17. 21. 13. 16. 22.]]
```

Fig. 9: Classement des features par importance décroissantes pour les 3 alorithmes suivant : Logistic Regression, Random Forest et Gradient Boosting

Les résultats confirment que les features 15 et 3 sont primordiales et souligne également l'importance des features 4 et 12.

C. Select KBest

Enfin, nous croisons les résultats de la corrélation et de RFE avec Select KBest qui est une méthode utilisée pour donner l'importance relative de différentes caractéristiques dans un ensemble de données sur la base de tests statistiques.

```
Index of Relevant feature in descending order of importance [14  2 18  3 19  9 10 21 22 12 15 23  4  1 11  5  0 20  7 16  6 13  8 17]
```

Fig. 10: List des features classé par ordre décroissant d'importance

Nous pouvons également constater que les caractéristiques des indices 3 et 15 sont les plus importantes, suivies par 18, 3 et 19 en utilisant KBest. Ce qui conforte notre première analyse de correlation.

D. Résultats

Après avoir testé avec plusieurs valeurs de K, c'est en gardant K=3 features que nous obtenons les meilleurs résultats suivants :

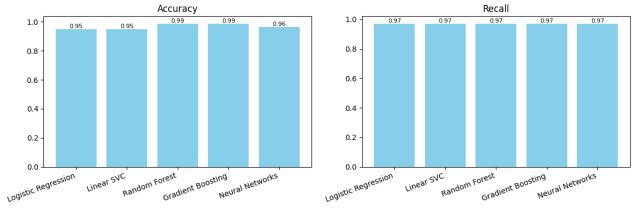

Fig. 11: Metriques sur le dataset ne conservant que les 3 features principales

On observe qu'on obtient des résultats presque aussi satisfaisant en termes de recall avec 0.97 pour tous les

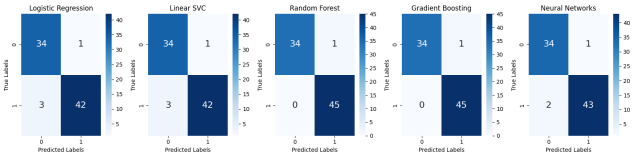

Fig. 12: Matrice de confusion pour les 5 modèles ne conservant que les 3 features principales

algorithmes en utilisant seulement 3 features. De plus il semblerait que cela permette au Réseau de neurones une meilleur généralisation avec une accuracy de 0.96 contre 0.88 auparavant.

VII. BONNES PRATIQUES DE PROGRAMMATION

La production d'un code clair, bien construit et bien annoté est primordial afin d'assurer une prise en main facile pour un utilisateur extérieur. Ainsi nous avons mis l'accent sur la structuration du code pour bien pouvoir identifier les différentes parties (preprocessing, entrainement, feature selection), et sur la cohérence de la séparation entre la partie librairie et la partie test. Le fichier workflow.py fait office de librairie comportant toutes nos fonctions appelées dans le notebook de test. Nous avons également pris soin d'expliciter les différentes variables et fonction intermédiaire afin de faciliter leur utilisation et leur compréhension pour autrui. Il est également important d'assurer la robustesse de notre code et de s'assurer de la fiabilité des résultats produits. Ainsi nous avons utilisé des tests réguliers dans plusieurs fonctions sous la forme d'assertion python, ou d'exception pour s'assurer de la coherence des différents éléments. Par exemple à la fin de notre preprocessing, nous avons mis un test assurant l'abscence totale de NaN values. Ou encore pour la partie preprocessing, lors de l'ouverture des csv, nous avons mis en place une exception si le fichier ne possède pas de colonne "id".

VIII. COMPLEMENT TENSORFLOW

L'architecture retenu est celle d'un MLP. Nous avons fait le choix de l'architecture suivante.

```python
def build_model(x_train) :
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Dense(32, input_shape=(x_train.shape[1],), activation='relu'))
    model.add(tf.keras.layers.Dense(16, activation='relu'))
    model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

    return model
```

Fig. 13: Fonction de construction du modèle

A noter que pour un problème relativement simple comme celui ci, il vaut mieux limiter le nombre de neurones par layer afin d'éviter l'overfitting. Nous avons ensuite fait le choix de la BinaryCrossEntropy comme fonction de coût justifiée par le caractère binaire de notre problème. Enfin, lors de la compilation, il est également important de choisir la métrique que nous souhaitons utiliser. Nous avons pris le Recall, expliquant ainsi un meilleur recall que accuracy notamment sur le premier dataset comme mentionné auparavant.

IX. Conclusions

En conclusion, malgré des dataset relativement simples à apprendre, ce projet nous a permis de comparer différents algorithmes de classification, et surtout de mener un travail de selection de features qui est indispensable pour de plus larges dataset. Enfin, il nous a permis de souligner l'importance de comprendre les choix de modélisation et les répercution sur les différentes métriques, comme on a pu le voir avec le premier dataset où les cas positifs au sens "patient malade" sont labelisés par un 0, ce qui inverse donc le sens de la matrice de confusion si on n'y porte pas attention.

References

[1] L. J. Rubini, "Chronic kidney disease," 2015.
[2] H. Doerksen, "Banknote authetication dataset," 2012.