Rapport laboratoire

Nguyen Hoanh Duc Trung
Alexey Paulot

14/12/2021

Contents

1 Introduction **2**

2 PMC **2**
2.1 Notre implémentation du PMC . . . 3
2.1.1 Heart . . . 3
2.1.2 Bands . . . 4
2.2 Sklearn . . . 5
2.2.1 Heart . . . 5
2.2.2 Bands . . . 6
2.3 Comparaison . . . 7

3 Bagging **7**
3.1 Notre implémentation . . . 7
3.2 Sklearn . . . 8
3.2.1 Heart . . . 8
3.2.2 Bands . . . 8
3.3 Réseaux convolutifs . . . 9
3.4 mono-couche . . . 9
3.5 multi-couche . . . 10

4 Conclusion **11**

1 Introduction

Tout d'abord, nous avons implémenté les modèles d'apprentissage suivante dans le cadre de ce cours:

- PMC
- Bagging
- Réseaux convolutifs

Dans ce rapport, nous n'allons pas expliquer les modèles, mais seulement montrer nos résultats expérimentaux, c'est à dire la description des traitements réalisés sur les données avec l'apprentissage, puis des tableaux ou des graphiques montrant les résultats obtenus par validation croisée. L'analyse des résultats se fera sous forme de discussion.

En terme de données, nous avons utilisées 2 datasheets différentes:

- heart
- bands

2 PMC

Avant d'entraîner notre modèle MLPClassifier, nous avons effectuer quelques traitement sur les données. Tout d'abord, nous avons séparé notre données en 2 tableaux:

- un tableau contenant tous les données des attributs.
- un tableau contenant tous les données des classes.

L'équivalente en code:

```
data_train, class_train = PMC.open_file_dat("path/file.dat")
```

Ensuite, à partir de ces tableaux, nous utilisons la fonction train_test_split de la libraire sklearn afin de générer 2 set de données, un set de données sert à entraîner notre modèle, et un set de données sert à valider notre données. Chaque set contient:

- un tableau contenant tous les données des attributs.
- un tableau contenant tous les données des classes.

L'équivalente en code:

```
data_train, data_test, class_train, class_test =
train_test_split(data_train, class_train, test_size=0.1, random_state=0)
```

Dans notre cas, notre set de données d'entraînement représente 90% des données, et notre set de données de test représente 10% des données.

2.1 Notre implémentation du PMC

2.1.1 Heart

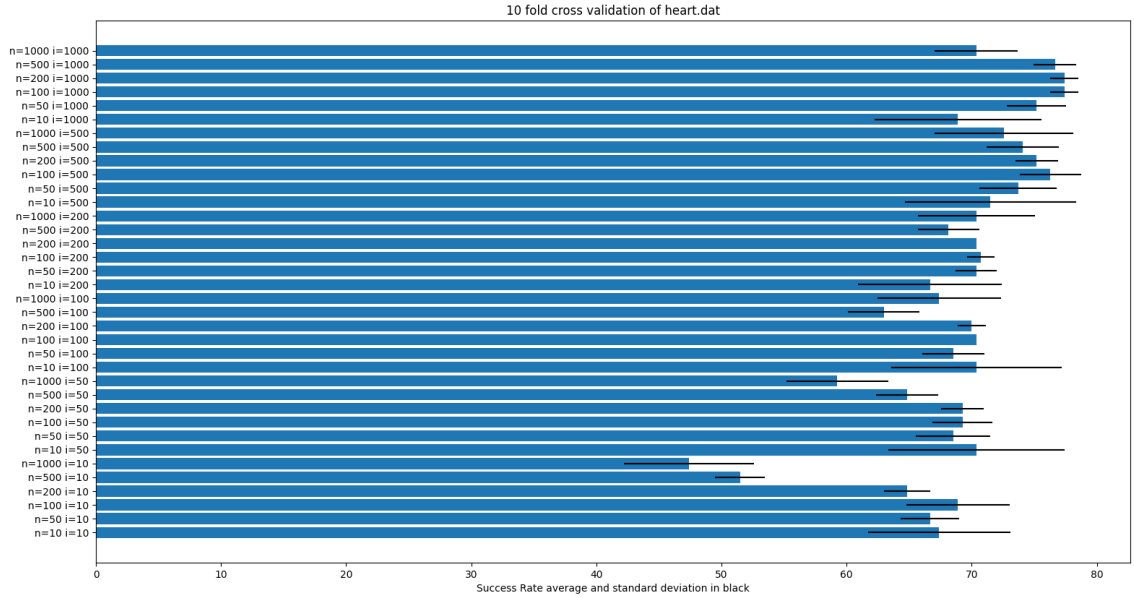

Figure 1: Courbe précision

On voit sur le graphique que avec peu d'itération il y a beaucoup d'écart type. Même chose quand il y a 10 neurones. On voit que le meilleur apprentissage est vers 77% avec 1000 itérations de 100 et 200 neurones et on est en sur-apprentissage sur 500 neurones et après. Malheureusement j'ai pas test plus d'itération car mon PMC est lent. On voit aussi que l'écart type augmente après le sur-apprentissage. On voit donc que en sous-apprentissage et sur-apprentissage l'écart type augmente. On peut donc en conclure que le meilleur est 100 neurones avec 1000 itérations on va donc l'utiliser plus tard dans le bagging.
La classe majoritaire est 55% et avec 77% de taux on est bien au dessus donc c'est un succès.

2.1.2 Bands

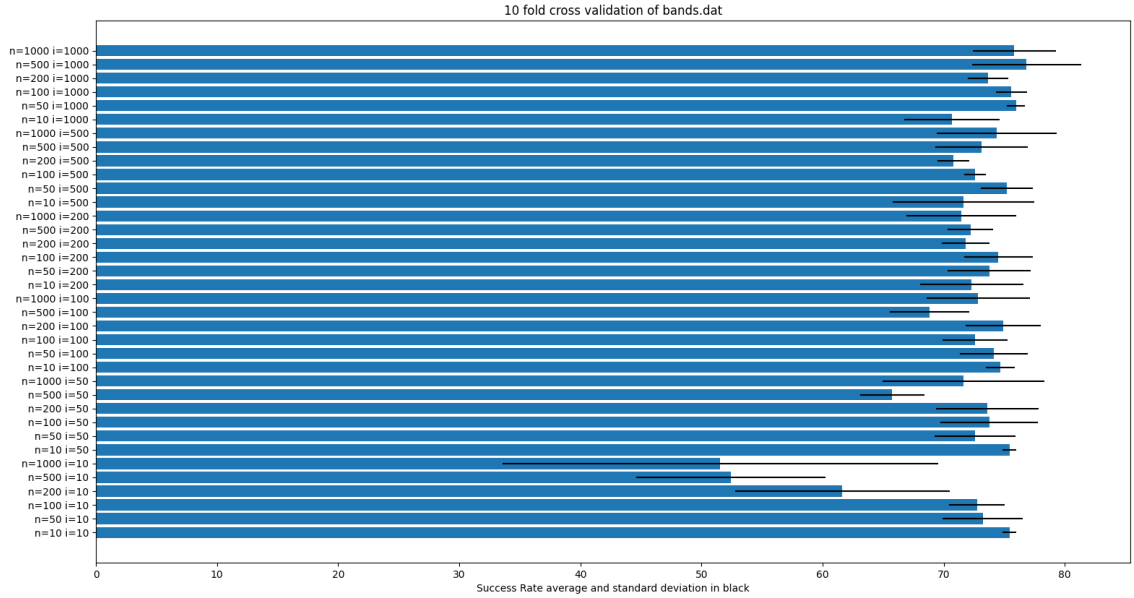

Figure 2: Courbe précision

On voit que le meilleur taux est à 500 neurones et 1000 itérations avec 76%. Par contre par rapport à heart.dat l'écart type est toujours identique et un peu aléatoire. Pour ce dataset peut être avoir plus d'itération aurai été bénéfique mais j'ai pas test avec plus de 1000 itérations malheureusement.

2.2 Sklearn

2.2.1 Heart

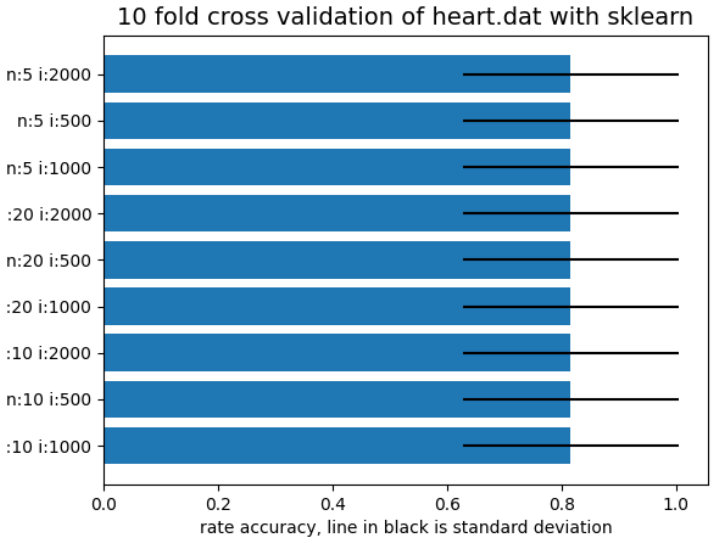

Figure 3: Courbe précision

Dans ce graphique ci-dessus, on peut voir la moyenne de 10 fois validation croisée pour chaque modèle entraîné avec N neurones et I itérations. Pour chaque modèle entraîné, on peut aussi voir l'écart-type qui est représenté par le trait noir sur la bar. De ce qu'on peut en déduire de ce graphique, c'est que notre modèle ,entraîné sur les données Heart.dat, a un taux de précision de 72% environ. Cela se confirme par le graphique ci-dessous.

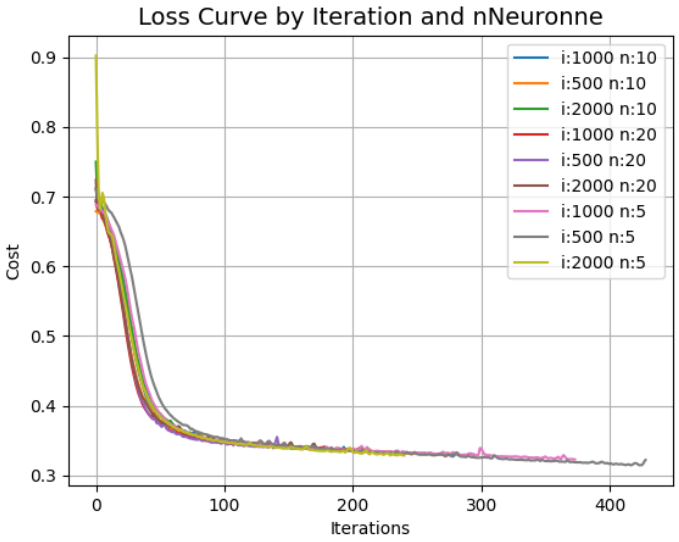

Figure 4: Courbe précision

Sur ce graphique ci-dessus, on peut voir la courbe de taux d'erreur de chaque modèle entraîné avec N neurones et I itérations sur les données Heart.dat. Ce graphique confirme notre observation faite précédemment, c'est à dire qu'à la fin de d'itérations de chaque modèle mesurée, le taux d'erreur de 30% environ.

2.2.2 Bands

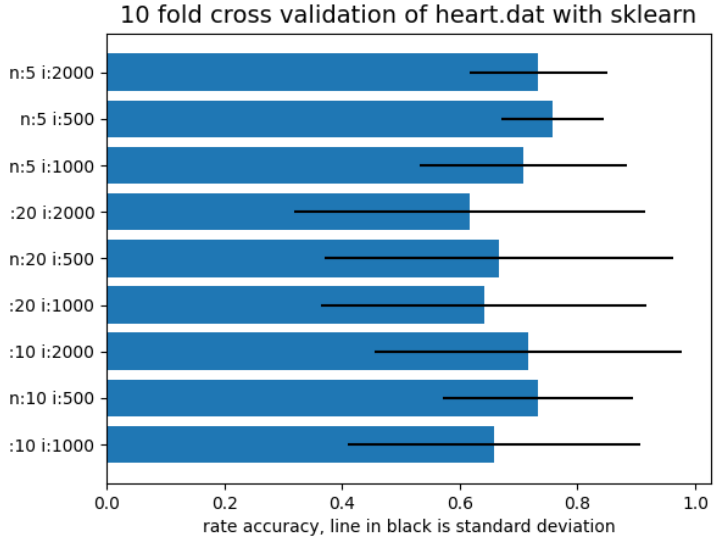

Figure 5: Courbe précision

De la même manière que nous avons analysée le graphique pour les données Heart.dat, on peut en déduire que notre modèle, entraîné sur les données Bands.dat, a un taux de précision très varié selon le modèle entraîné avec différente N neurones et I itérations.

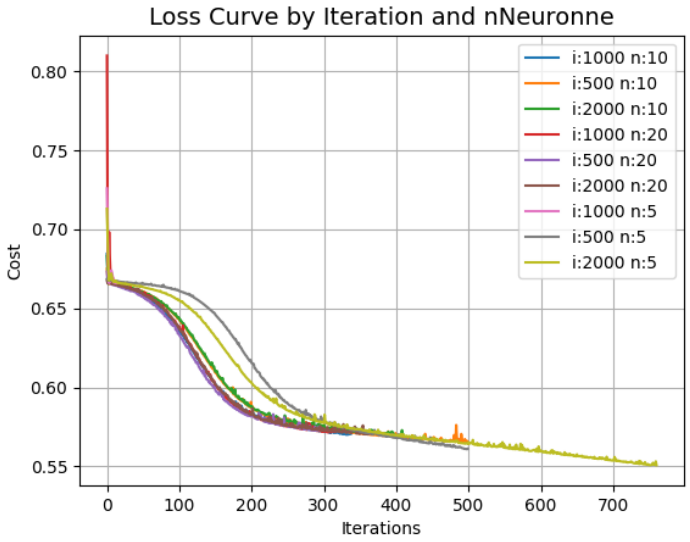

Figure 6: Courbe précision

De la même manière que nous avons analyser la courbe de taux d'erreur de Heart.dat. Ce graphique nous confirme notre observation, mais certain différence reste notable, certains courbes progressent plus vite que les autres.

2.3 Comparaison

On voit que sklearn a des meilleurs taux avec très peu de neurones mais a un écart type plus grands.

3 Bagging

3.1 Notre implémentation

Pour le heart.dat j'ai fait avec 100 neurones et 1000 itérations et en prenant 50% de la data par PMC j'ai eu 85.78 de moyenne et 1.678 d'écart type. J'ai test aussi avec plus de donne mais ça donne par énormément en plus de taux de succès. Pour bands.dat vu que le taux optimal est obtenu avec 500 neurones et 1000 itérations j'ai pas tester car bagging de 25 PMC de telle taille me prendrai 10h alors une validation croisée encore plus.

3.2 Sklearn

3.2.1 Heart

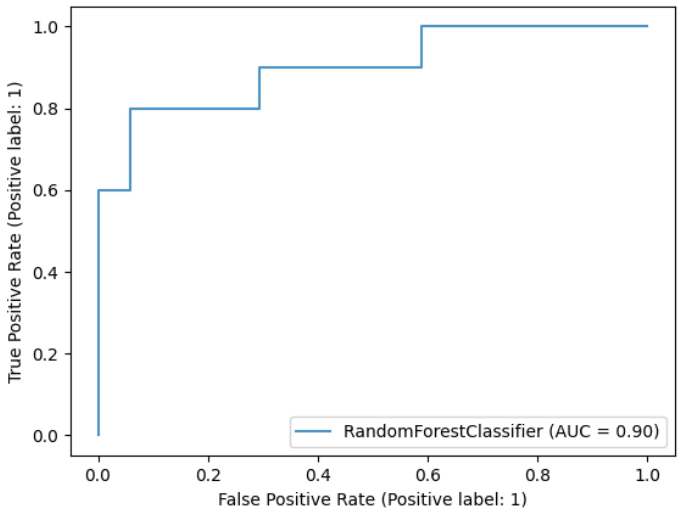

Figure 7: Courbe random Forest

Sur ce graphique ci-dessus, on peut voir la courbe de taux de classification d'un modèle entraîné sur les données Heart.dat, on peut voir que cette courbe se rapproche vers un taux de classification positive au fur à mesure des itérations.

3.2.2 Bands

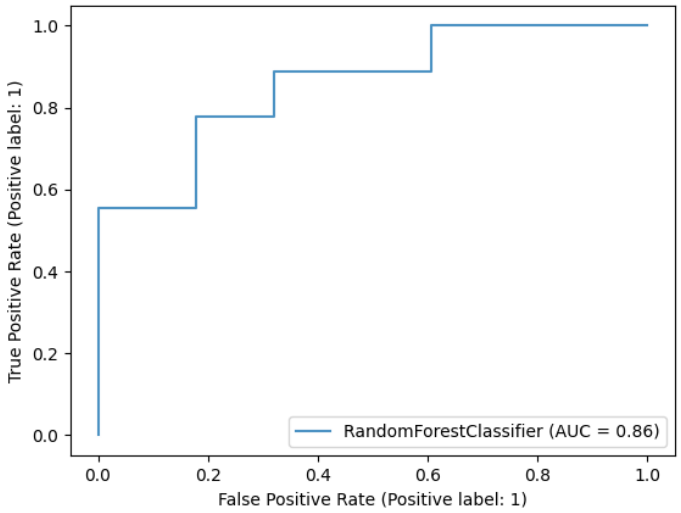

Figure 8: Courbe random Forest

Même observation faite pour le modèle entraîné sur les données Heart.dat, à la différence que cette courbe se rapproche plus vite du taux maximal de classification positive.

3.3 Réseaux convolutifs

Dans cette partie, nous avons utilisés 2 type de réseaux convolutifs:

- mono-couche
- multi-couche

Plus précisement, le réseaux dit mono-couche contient juste une seul couche convolutionnelle et de max-pooling avec 1 couche denses. Pour le réseaux dit multi-couche, cela contient deux couche convolutionnelle et de max-pooling, avec 1 couche denses.

Tous les graphics ci-dessous montrent la performance de nos modèles à 2 types de réseaux convolutifs, notamment le taux de précision et d'erreur par le nombre d'entraînement faite de notre modèle.

3.4 mono-couche

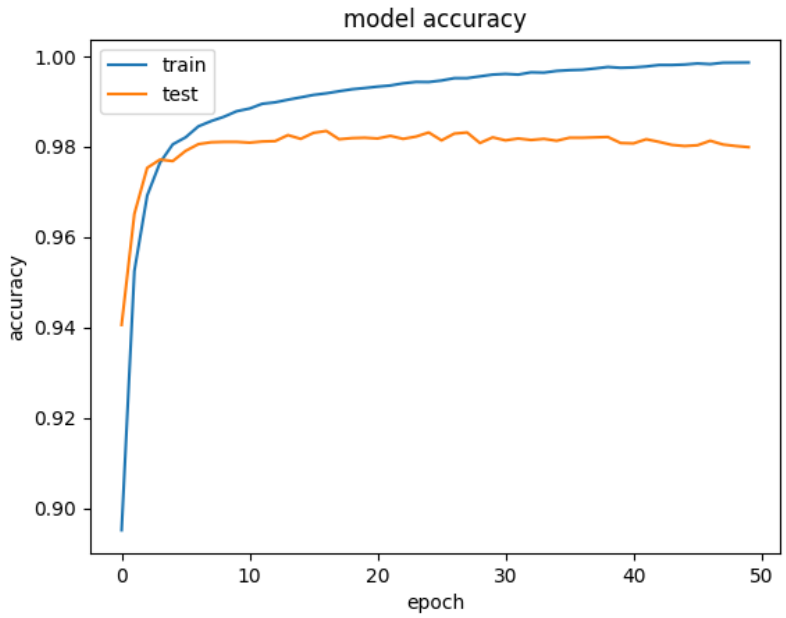

Figure 9: Courbe de taux de précision du réseaux convolutifs mono-couche

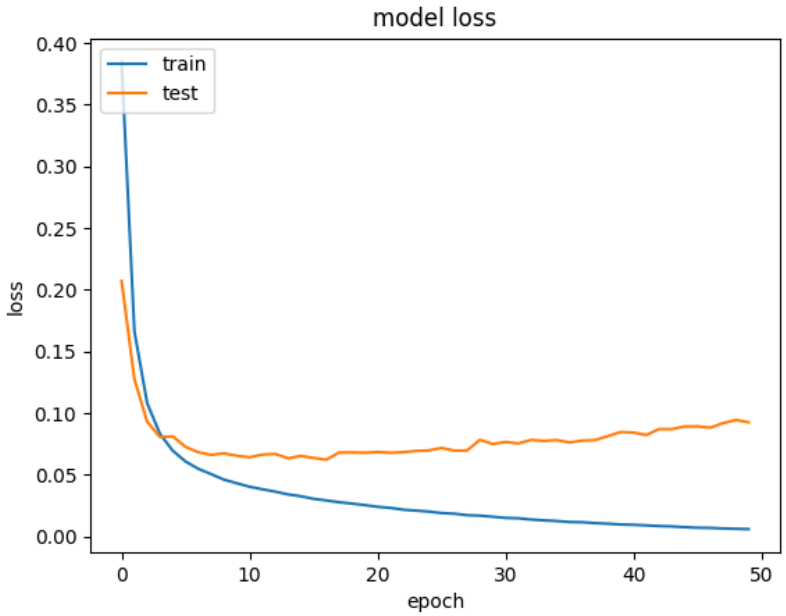

Figure 10: Courbe de taux de perte du réseaux convolutifs mono-couche

3.5 multi-couche

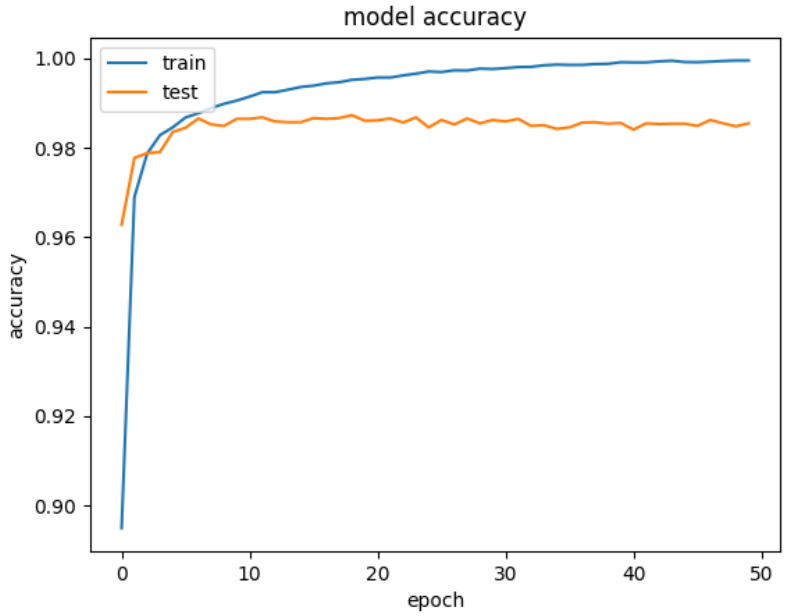

Figure 11: Courbe de taux de précision du réseaux convolutifs multi-couche

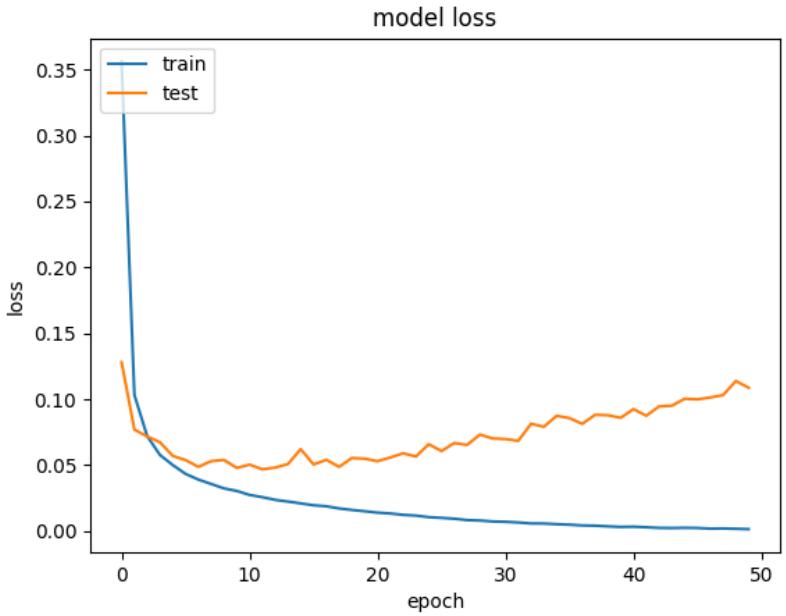

Figure 12: Courbe de taux de perte du réseaux convolutifs multi-couche

4 Conclusion

Grâce à ce projet, nous avons appris beaucoup de choses, notamment le PMC, le Bagging, les librairies sklearn, et Keras.