M2 Data Science

Machine learning de fall data project

Par
LIAO Puwei
ZHANG Shurong

Tutrice Mathilde Mougeot

Novembre 2021

Table des matières

1 Introduction

Dans ce projet, nous avons un dataframe "falldataproject.csv" qui contient un échantillon de la caractérisation sur le pas des certaines personnes. De plus, dans notre dataframe, avec 2821 individus, il y a 87 variables et 1 variable binaire. Les variables correspondent à plusieurs indicateurs calculés soit sur les signaux bruts, soit sur le signal dérivé, soit sur l'énergie des signaux. Et aussi, toutes les variables sont centrées.

Notre but de ce projet est de faire les modèles de classification d'après le training data que nous avons choisi et de faire des test pour ces modèles puis choisir une modèle qui nous semble le plus convenable pour notre classifier final.

Pour réaliser ce but, nous écrivons d'abord une fonction de vérification qui nous affiche directement les critères que nous voulons :

— Accuracy
— Precision
— Recall
— F1-score
— AUC

A l'aide de cette fonction, nous pouvons facilement voir les performances de notre 10 modèles utilisés : Méthode de naive bayesien, LinearDiscriminantAnalysis , QuadraticDiscriminantAnalysis, LogisticRegression, Décision tree, Bagging, Random Forest, KNN, Adaboost et Gradient Boosting.

Pour comparer toutes ces modèles, il faut d'abord estimer les hyperparamètres (Paramètres optimaux qui produit les meilleures performances) pour les modèles KNN, Bagging, Décision Tree et Random Forest.

On rappelle que le choix des paramètres pour l'entrînement des modèles est très important, si on n'a pas bien choisi les paramètres, on risque d'obtenir un modèle qui n'est que très peu utile à la résolution des problèmes et qui sera difficilement généralisable sur des nouvelles données.

2 Optimisation des hyperparamètres

Pour faire un choix de paramètres que nous voulons, le critère que nous avons décidé de regarder est le Roc Auc Score puisque notre jeu de données n'est pas équilibré : Très rare de classification 1 et beaucoup de classification 0, dans ce cas ce sera inutile de regarder la critère accuracy. Donc dans la suite, on regarde la performance sur le Roc Aoc Score pour le choix de hyperparamètres pour les modèles KNN, Bagging, Décision tree et Random Forest.

2.1 Hyper paramètres pour le modèle Arbre de décision

Pour aboutir cette recherche sur les hyperparamètres, nous programmons un algorithme qui s'appelle treeopti() qui nous renvoie les paramètres optimaux en sortie :

— max_depth
— $min\_samples\_split$
— $min\_samples\_leaf$

On rappelle que *max_depth* est la longueur maximum des arbres, *min_samples_split* est le nombre minimal d'exemple que le modèle doit avoir pour pouvoir faire une nouvelle séparation, plus paramètre sera grande plus l'arbre sera simple et plus généralisable. $min\_samples\_leaf$ est le minimum d'exemple requis pour créer une feuille.

Le principal de l'algorithme est de réaliser un K-fold avec $n = 100$ (qui est le nombre de découpage), et testsize = 0.25 (qui est la proportion de la data test), à l'étape suivante on parcourt de 3 à 10 pour le paramètre *max_depth*, de 2 à 10 pour *min_samples_split* , de 2 à 10 pour $min\_samples\_leaf$ en gardant la groupe de hyperparamètres qui a une accuracy mieux.

A la fin de cet algorithme, le résultat que nous avons obtenu est la suivante :

1. $max\_depth = 3$
2. $min\_samples\_split = 4$
3. $min\_samples\_leaf = 4$

Ceux qui sont les hyperparamètres optimaux pour l'arbre de décision.

Cependant, on remarque que le temps d'exécution de cet algorithme est très long puisqu'on réalise un K-fold sur un très grand Dataset. C'est la disavantage de la méthode K-fold.

2.2 Nombre d'estimateurs pour le modèle Bagging

On rapplle que les paramètres du modèle Bagging sont *base_estimator* qui est un estimateur de base (qui est l'arbre de décision ici), *n_estimators* est le nombre d'estimateur qu'on va avoir dans notre foule (qu'on va chercher à l'optimiser).

De la même façon que l'arbre de décision, nous procédons la même procédure en gardant les hyper paramètres obtenus en arbre de décision, en parcourant 10 à 20 l'estimation optimale de n que l'on a obtenu est :

— $n = 19$

2.3 Meilleur k pour le modèle KNN

On rappelle que le hyperparamètre du modèle KNN est le nombre de voisins. Pour l'optimiser, on fait la même démarche d'analyse pour chercher le meilleur k pour la méthode KNN,on parcourt de 1 à 10 avec 50 découpage, en regardant le critère Auc Roc Score, le meilleur k que nous avons réussi à trouver est :

— $k = 9$

2.4 *max_depth* pour le modèle Random Forest

On rappelle que *max_depth* est la longueur maximum des arbres. Pour trouver le meilleur *max_depth*, on a testé les différentes valeurs de *max_depth* et on compare leur performances en AUC et RECALL. Et on constate que pour $max\_depth \geq 4$, on a les meilleures performances et donc on va choisir $max\_depth = 4$ puisqu'on essaie toujours d'utilise des arbres peu profond dans le modèle Random Forest.

3 Comparaisons de performances

A ce moment, nous avons toutes les critères de 10 modèles de classifier, et avec toutes les modèles que nous avons modélisés, on est capable de programmer une matrice de validation croisée appliquant sur toutes les 10 modèles avec une partie de training data et une partie de test data. Grâce à ça, on obtiendra une matrice de taille $n * 20$, avec n le nombre de découpage, nous avons pris $n = 50$ ici.

3.1 AUC_ROC_SCORE

Le premier score que nous nous intéressont est le AUC_ROC_SCORE, et voici dessous la figure du boxplot sur AUC_ROC_SCORE. D'après cette figure, on peut constater que le classifier qui nous satisfait le plus est le classifier Random Forest puisque les médianes de RF_train et RF_test sont plus proches que les autres classifiers. Et le classifier Naive Bayesien est le plus mauvais classifier.

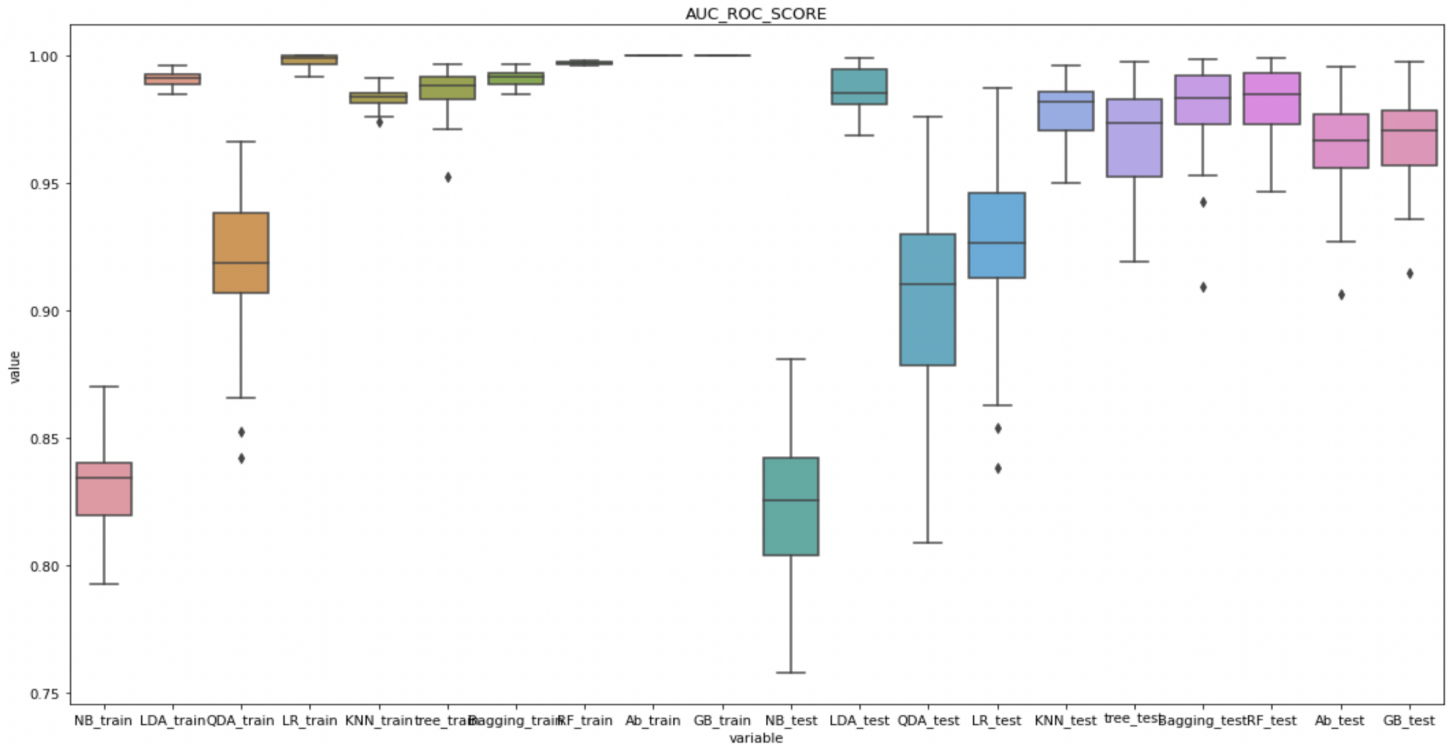

FIGURE 1 – AUC_ROC_SCORE

3.2 Recall

Comme les données ne sont pas équilibrées, il y a très peu de classification de 1, donc nous décidons de rajouter une autre score à comparer qui est Recall, ce qui s'implique le ratio de true positive et false positive+true positive, c'est à dire le taux de réussite de classifier la classification de 1, ce qui est très rare dans notre data frame, nous traçons alors le boxplot de Recall.

D'après ce boxplot, on peut voir que le modèle qui nous convient le plus est toujours le classifier Random Forest.

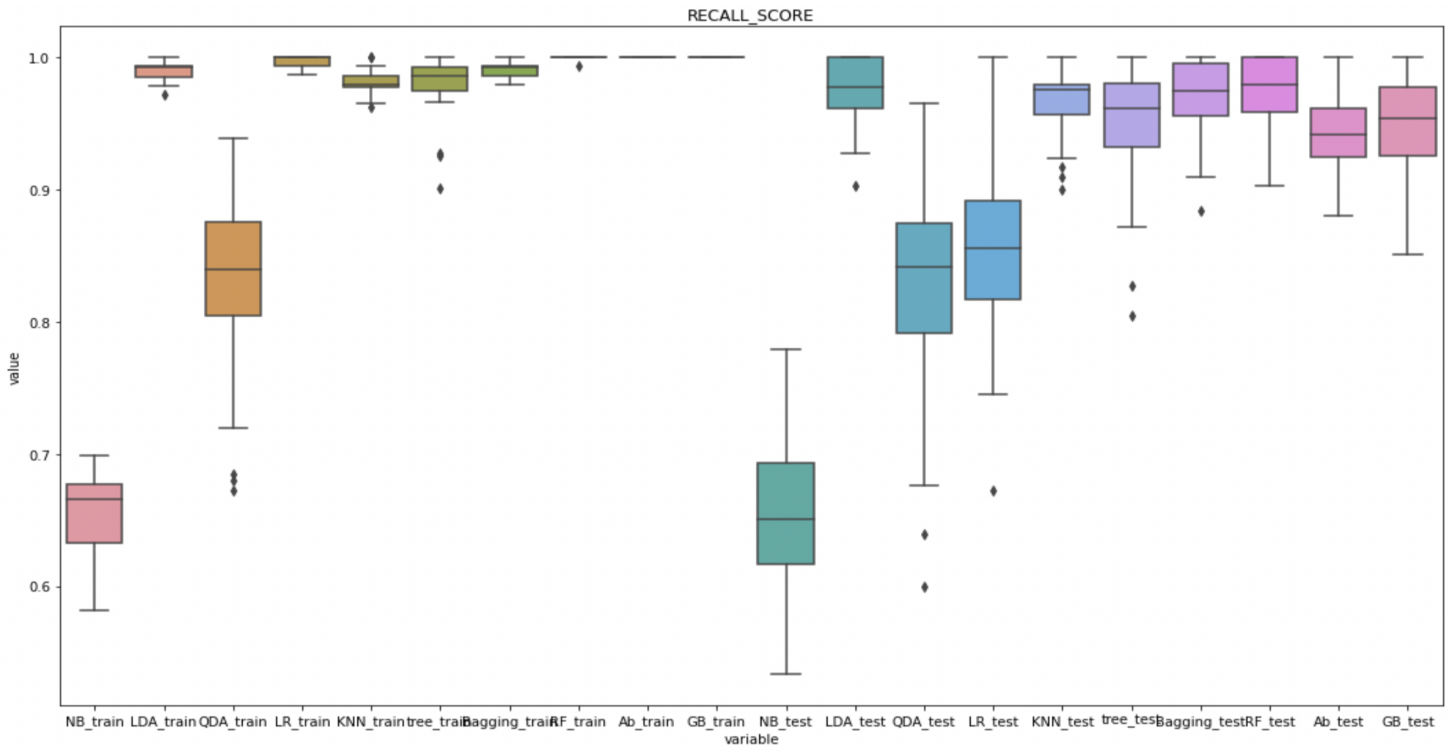

FIGURE 2 – RECALL_SCORE

4 Conclusion

Nous choisissons donc le classifier Random Forest de paramètre $max\_depth = 4$. C'est le modèle qui nous satisfait le plus, avec accuracy et presicion près de 1. De plus, le score de recall et le score de AUC sont à la fois le meilleur parmis ces 10 modèles de classification que nous avons utilisé, Ainsi, le score F1 nous semble aussi le meilleur parmis ces modèles.

1. Accuracy : 0.9886685552407932
2. Precision : 0.9777777777777777
3. Recall : 0.8627450980392157
4. F1-score : 0.9166666666666665
5. AUC : 0.930609190240982

Ensuite, on s'intéresse sur l'importance des variables dans le modèle Random Forest, et on a trouvé que deriv feat X26, deriv feat X2, deriv feat X25 contiennent plus l'importance que les autres variables donc ils sont les variables importantes. fft feat X24, fft feat X17 et fft feat X23 qui ont l'importance nulle donc ils ne sont pas les variables importantes.

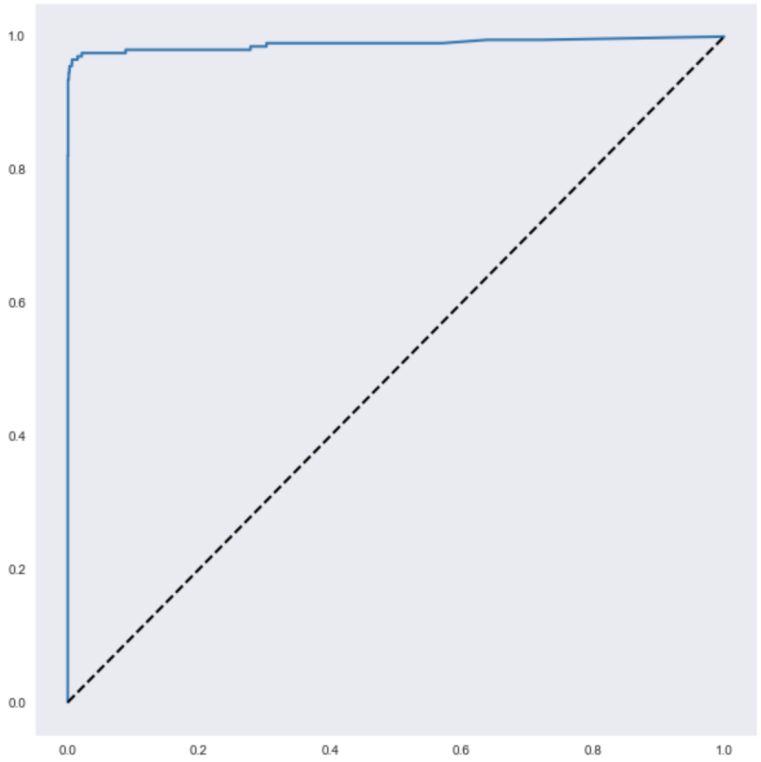

FIGURE 3 – ROC CURVE DE RF

On rappelle que AUC est l'aire sous la courbe de ROC. Si AUC est plus poche de 1, alors ce classifier est meilleur. D'après la figure 3, on peut justifier que AUC est très poche de 1, d'où Random Forest est un bon classifier ici.