Projet de Machine Learning - détection de chute

Etienne Gaucher, Paul Mazet

Novembre 2021

Table des matières

1 Introduction

Ce projet a pour but de détecter la chute d'une personne à partir de capteurs piézoélectriques. Ces derniers sont insérés dans le sol et enregistrent un signal qui a été transformé pour en obtenir des indicateurs numériques. Des volontaires ont simulé plusieurs situations, avec et sans chute, pour constituer un jeu de données. L'intérêt de détecter une chute est naturellement d'alerter les secours le plus rapidement possible afin d'augmenter les chances de survie et diminuer les éventuelles séquelles. On cherche à utiliser le machine learning pour créer un modèle capable de détecter une chute à partir des indicateurs enregistrés.

2 Le jeu de données

Le jeu de données initial est contenu dans le fichier `falldataproject.csv`. On l'importe dans le notebook sous la forme d'un dataframe avec 2 821 observations et 89 variables. On réalise ensuite l'analyse de ce dataframe :

- on commence par vérifier que toutes les données ont été correctement importées, et que les variables sont bien numériques (float ou integer)
- on supprime la variable `obs` car elle correspond uniquement au numéro des observations, elle n'apporte donc pas d'information
- on élimine ensuite les lignes et les colonnes dupliquées
- on vérifie que le dataframe ne contient pas de valeur NA

Une fois l'analyse terminée, on définit le dataframe X qui contient les covariables, et le dataframe Y qui contient la variable cible. On dispose de 85 covariables, ce qui montre que 2 covariables étaient des doublons. L'énoncé nous précisait que les données étaient normalisées, ce que nous avons contrôlé.

3 Définition du modèle idéal

On souhaite prédire si une personne a chuté à partir des covariables d'une observation. Si une chute n'est pas détectée, les secours ne sont pas prévenus immédiatement ce qui diminue les chances de survie. Ce contexte nous impose donc de détecter précisément toutes les chutes. En terme de score, cela nécessite d'avoir un score de sensitivity le plus élevé possible. Pour rappel, la sensitivity est également appelée recall ou true positive rate, et est définie de la façon suivante :

$$\text{sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False Negative}}$$

Plus la sensitivity est proche de 1 et plus le modèle a détecté précisément les chutes.

Cependant, il est important que le modèle ne détecte pas de chute si la personne n'est pas tombée. Le nombre de False Positive doit être raisonnable. En effet, si une alarme est déclenchée à chaque détection de chute, le modèle doit éviter les erreurs. On a besoin que le score precision soit aussi élevé que possible. Pour rappel, la precision est définie par :

$$\text{precision} = \frac{\text{True positive}}{\text{True positive} + \text{False Positive}}$$

Idéalement, on cherche un modèle avec un recall égal à 1 et une precision égale à 1. Puisque cela est difficilement réalisable, on préférera donc :

1. un modèle avec un recall le plus élevé possible, car la priorité est de détecter les chutes
2. parmi les modèles avec les meilleurs recall, on sélectionnera le modèle avec la precision la plus élevée.

Ainsi, le modèle sélectionné ne sera pas forcément le modèle avec le plus haut recall, car si sa precision est trop faible, un modèle avec un recall légèrement inférieur mais une precision largement supérieure sera préféré pour éviter les fausses alertes. On cherche un compromis entre recall et precision, avec une priorité pour un recall élevé.

4 Implémentation de la validation croisée

Avant d'entraîner les différents modèles, nous avons implémenté une fonction pour connaître la performance d'un modèle. Le but de la fonction `performance_metrics` est de retourner, pour un modèle passé en argument, les scores suivants : accuracy, precision, recall et f-score. Cependant, pour connaître le pouvoir de prédiction d'un modèle, on ne doit pas utiliser le même jeu de données pour entraîner et tester le modèle. Nous avons donc choisi d'utiliser la validation croisée (cross-validation) pour estimer les scores. Le nombre K de sous-dataset (fold) est un argument de la fonction, qui vaut 5 par défaut. Chaque score est calculé K fois par validation croisée. La fonction retourne la moyenne de chaque score.
Ainsi, pour chaque modèle, on pourra calculer la valeur des scores, ce qui nous permettra ensuite de savoir quel modèle est le plus efficace pour répondre à la problématique.

5 Les modèles

Nous avons testé une dizaine de modèles, paramétriques et non-paramétriques : naive Bayes classifier, logistic regression, linear and quadratic discriminant analysis, support vector machine, K-nearest neighbors, decision tree, decision tree with bagging, random forest, boosting, gradient tree boosting et stacking. Pour quelques modèles, nous avons choisi de rechercher certains paramètres optimaux, car nous avons estimé qu'ils étaient déterminants dans les performances du modèle :

- le nombre de voisins utilisé pour le modèle K-nearest neighbors
- le nombre d'arbres et la profondeur maximale des arbres pour le modèle random forest

Les paramètres ont été sélectionnés par validation croisée sur un ensemble de valeurs possibles défini manuellement.

De plus, nous avons testé des modèles avec des spécifications différentes. Par exemple, nous avons testé le modèle support vector machine avec un kernel linéaire, polynomial et circulaire. De même, nous avons testé le modèle random forest avec et sans bagging.

6 Choix du modèle

On commence par tracer le recall et la precision de chaque modèle.

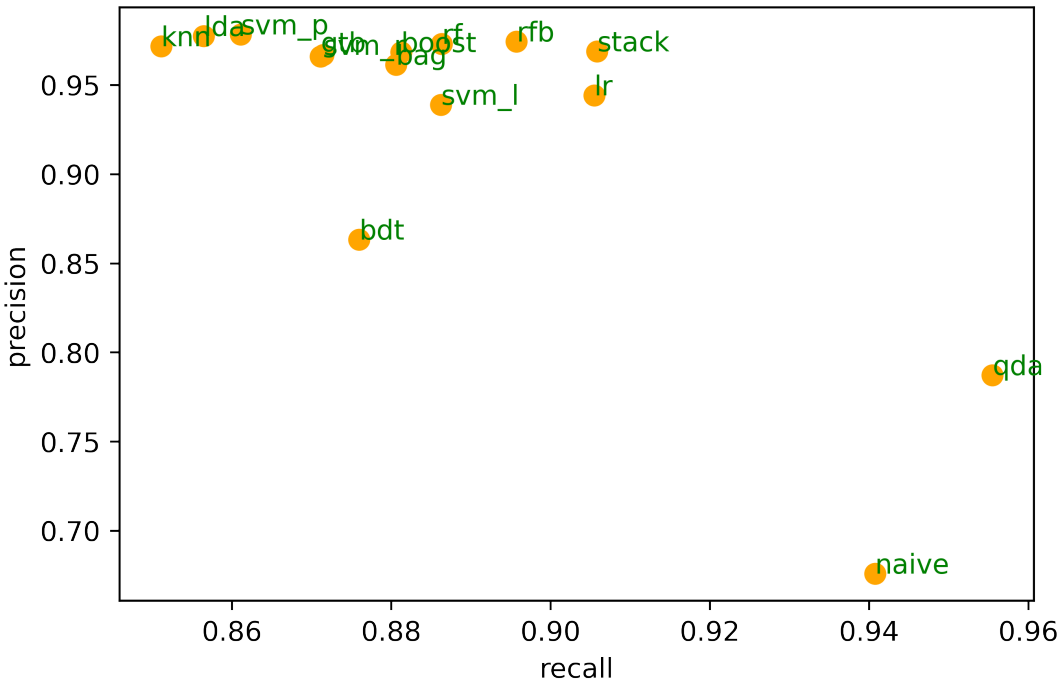

Figure 1: Recall and precision

2 modèles se distinguent car ils sont les plus proches du coin supérieur droit, c'est-à-dire qu'ils ont un recall et une precision élevés. D'un côté, le modèle quadratic discriminant analysis avec un recall légèrement supérieur à 0,95 mais avec une precision de 0,78. De l'autre côté, le modèle stacking avec un recall de 0,90 et une precision de 0,96.

Un compromis doit être réalisé entre ces deux modèles comme expliqué précédemment. La différence de recall entre le modèle quadratic discriminant analysis et le modèle stacking vaut 0,06. La différence de precision entre le modèle quadratic discriminant analysis et le modèle stacking vaut 0,18.
Nous avons considéré que la différence de precision était trop élevée par rapport à la différence de recall.

Le modèle stacking est donc notre meilleur modèle. Voici les scores du modèle stacking :

Accuracy	Precision	Recall	F-score
0.99	0.96	0.90	0.93

Concrètement, cela signifie que notre modèle est capable de détecter les chutes dans 90 % des cas. Lorsqu'il détecte une chute, il y a 96 % de chance que ce soit vrai.

7 Améliorations envisageables

Notre travail comporte bien sûr des pistes d'amélioration que nous n'avons pas traité en raison d'un manque de temps ou d'un manque de connaissances. Nous aurions pu nous intéresser aux points suivants :

- pour les modèles utilisants des arbres, nous avons toujours utilisé le critère Gini index. Nous pourrions effectuer le même travail avec un critère de cross-entropy
- un modèle avec des réseaux de neurones pourrait être envisagé
- même si la validation croisée avec 5 folds est courante, utiliser plus de folds pourrait être testé
- pour chaque modèle, nous pourrions tracer la courbe ROC obtenue par validation croisée afin de connaître le meilleur seuil de décision