Forage de données
Devoir 1
8INF854

Papa Alioune FALL, Quentin LETORT

18 février 2020

UQAC
Université du Québec
à Chicoutimi

Résumé

Modification d'un algorithme existant de classification par arbre de décision et application de l'algorithme pour la détection des intrusions sur un réseau informatique (cybersécurité).

1 Synthèse de l'article

L'article porte sur la détection des intrusions dans les systèmes informatiques qui sont en forte augmentation. Il présente des méthodes pour construire un système de détection d'intrusion efficace et adaptatif sur un système ou réseau informatique.

Les systèmes commerciaux de détection d'intrusion se basent principalement sur la reconnaissance de motifs communs à des attaques connues. Ce type de systèmes présente cependant une grande faiblesse car ils nécessitent des mises à jour fréquentes des règles de détection et ne sont pas capables de détecter des attaques encore inconnus. Par ailleurs, des systèmes de détection d'anomalies ont été développées pour faire face à ce dernier problème mais ces derniers détectent un trop grand nombre de faux positifs (trop d'attaques prédites alors qu'elles n'en sont pas réellement).
Ainsi, une approche proposée dans cet article est d'utiliser les techniques de forage de donnée et donc intégrer un agent intelligent dans un système de détection d'intrusion.

Les auteurs proposent un nouveau modèle d'apprentissage automatique de classification basé sur un arbre de décision dans le but d'améliorer le taux de détection et de diminuer le nombre de faux positifs. L'apprentissage par arbre de décision définit un ensemble de règles sur les attributs des individus d'un jeu de données afin de les séparer en différentes classes.
L'article s'intéresse tout particulièrement à l'algorithme ID3 tout en proposant une amélioration pour le calcul de l'entropie avec le remplacement de l'entropie de Shannon par celle de Havrda et Charvat.

Pour pouvoir tester leurs hypothèses, les auteurs proposent de comparer l'algorithme ID3 classique avec leur version modifié sur le jeu de données KDD99, très populaire dans le domaine de la détection d'intrusion. Ce jeu se compose de 5 grandes classes (attaque par déni de service (DOS), attaque « Remote to User », attaque « User to Root », attaque par sondage et connexion normale). Les deux métriques utilisées pour estimer la performance du système ont pu démontrer que l'algorithme (ID3 modifié) proposé par les auteurs étaient encore meilleur que ID3 standard avec une augmentation du taux de détection et une diminution du taux d'erreur.

2 Modification de l'algorithme ID3

Nous avons modifié l'algorithme ID3 de Weka en remplaçant l'entropie de *Shannon* par celle de *Havrda et Charvat* comme proposé dans l'article.
La modification concerne la méthode **computeEntropy()** où l'entropie vaut

désormais :

$$H(x) = \frac{1}{2^{1-\alpha} - 1}(\sum_{x} p^{\alpha}(x) - 1)$$

```
    private double computeEntropy(Instances data) throws Exception {

        double[] classCounts = new double[data.numClasses()];
        Enumeration instEnum = data.enumerateInstances();
        while (instEnum.hasMoreElements()) {
            Instance inst = (Instance) instEnum.nextElement();
            classCounts[(int) inst.classValue()]++;
        }

        double entropy = 0;
        for (int j = 0; j < data.numClasses(); j++) {
            if (classCounts[j] > 0) {
                entropy += Math.pow(classCounts[j] / (double) data.
                    numInstances(), getAlpha()) - 1;
            }
        }
        return entropy * Math.pow(Math.pow(2, 1 - getAlpha()) - 1, -1);
    }
```

3 Expérimentation

Le jeu de donnée utilisé pour cette expérimentation est celui fourni sur moodle, il contient 494 021 instances possédant chacune 14 attributs nominaux et un label.
Une première étape était d'extraire un sous échantillon de ce jeu de donnée afin de pouvoir entrainer et tester nos modèles (le jeu de donnée est trop volumineux pour être utilisé entièrement).
Afin de récupérer ce sous échantillon, nous avons appliqué le filtre *Resample* proposé par Weka ce qui a permis de ne retenir que 5% du jeu de données soit 24 698 instances (figure 1).
L'étape suivante consistait à tester les algorithmes de classification ID3 standard et ID3 modifié afin de le les comparer comme dans l'article. Nous avons choisi d'utiliser 75% de notre échantillon pour l'entrainement du modèle et 25% pour les tests grâce à l'option "*Percentage split*" proposé par Weka. Nous avons obtenu des résultats identiques pour les deux algorithmes (en choisissant alpha = 0,5 dans la version modifiée) comme en atteste les figures 2 et 3.
En effet, les taux de détection et d'erreur sont respectivement de 97,473% et 0,2915% pour les deux algorithmes. On remarque par ailleurs que 2.235% des instances n'ont pas pu être classé grâce au modèle.

Une limitation par rapport à cette première approche est la faible représentation de certaines classes comme U2R qui comporte seulement 2 instances dans tout notre échantillon et ces dernières se sont retrouvées dans l'échantillon d'entrainement ce qui nous a empêcher de tester notre modèle pour des données de cette

classe.
Dans ce cas, l'utilisation de la validation croisée s'avère intéressante car on va pouvoir retrouver ces instances dans un échantillon de test et ainsi tester le modèle sur des données de cette classe.
Nous avons donc utilisé l'option de Weka permettant d'entrainer le modèle en utilisant la technique 10-fold cross-validation. Cette technique divise notre échantillon en 10 sous-échantillons puis sélectionne un de ces sous-échantillons pour les tests tandis que les autres sont utilisés pour l'entrainement du modèle. Elle procède de même pour chaque sous-échantillon de notre échantillon initial. Ainsi, cette technique est plus lourde car elle va entrainer autant de modèle que de sous échantillon mais on s'assure que le modèle a pu être testé sur toutes les instances de l'échantillon. On peut observer sur la figure 3 que cette technique a permis d'augmenter le taux de détection a 97,846% et diminuer le taux d'erreur à 0,223%. Cette technique a été utilisé sur les deux algorithmes mais nous avons obtenu encore une fois les mêmes résultats.

Après ces premières expérimentations, nous souhaitions pouvoir entraîner notre modèle sur une plus grande quantité de donnée du jeu de donnée et nous rapprocher de la taille de l'échantillon utilisé dans l'article.
Nous avons donc choisi d'appliquer une sélection des meilleurs attributs ce qui nous permet la taille de chaque instance. Pour ce faire, nous avons utilisé la fonctionnalité Weka de sélection d'attributs en utilisant l'algorithme de recherche *best-first* avec l'évaluateur *CfsSubsetEval* (évalue la valeur d'un sous-ensemble d'attributs en considérant la capacité prédictive individuelle de chaque attribut et le degré de redondance entre eux). Nous avons conservé uniquement 3 variables grâce la méthode de sélection (service réseau, état de connexion et nombre d'échec pour la connexion) et ainsi pu utiliser l'ensemble de l'échantillon (soit 494021 instances ou 10% du jeu de donnée KDD99) pour entraîner et tester de nouveaux modèles de classification, ceci dans le but de nous rapprocher de l'expérimentation réalisé dans l'article qui se basait sur cette quantité d'instances.
Les résultats sont encore une fois équivalent entre les deux algorithmes ID3 et ID3 modifié mais on note une légère baisse du taux de détection avec 96,9% et une hausse du taux d'erreur à 3,1% (mais moins d'instances non classifiées). Même si le modèle est un peu moins performant que ceux construits précédemment, il reste efficace et présente certains avantages tels que la génération d'un arbre de décision moins complexe avec une hauteur faible (3) mais également un temps d'entrainement du modèle fortement diminué.
Le principal problème est que le taux de vrais positifs est particulièrement faible pour les classes faiblement représentés dans le jeu de données (U2R et R2L ont des ratios proches de 0).

Nous avons testé une dernière approche afin d'essayer de remédier à ce problème. Il nous semblait intéressant d'utiliser un algorithme d'augmentation de données afin d'augmenter le nombre d'instances pour les classes sous représentées. Weka propose notamment un filtre à cet effet avec SMOTE (*Synthetic Minority Over-*

sampling Technique).
Nous avons choisi d'appliquer cette technique sur les trois classes sous représentées :

- U2R : multiplication du nombre d'instances par 100
- R2L : multiplication du nombre d'instances par 50
- Attaque par sondage ("probing") : multiplication du nombre d'instances par 40

Nous avons ensuite appliquer le filtre *Resample* pour ne conserver qu'un échantillon de 5% de l'ensemble de ce jeu de donnée augmenté.
Le modèle généré à partir de cet échantillon a un taux de détection moins élevé que lors des approches précédentes (96,31%) mais le ratio de vrais positifs a considérablement augmenté pour les classes sous représentées (proche de 1 pour U2R et R2L et 0.89 pour les attaques par sondage) alors que le ratio de faux positifs est resté faible pour toutes les classes.
L'augmentation synthétique des classes sous représentées a ainsi permis d'améliorer l'efficacité du classifieur a reconnaître ces classes d'attaques (à nuancer néanmoins car les facteurs multiplicatifs importants ont sûrement engendré du sur-apprentissage).

4 Comparaison des résultats

En se basant sur les métriques de taux de détection et de taux d'erreur pour ID3 avec l'entropie de Havrda et Charvat, nos résultats sont semblables à ceux de l'article avec respectivement 97,5% et 0,3% (2,5% si on y ajoute les instances non classifiées).
Cependant, l'article montrait un amélioration pour ID3 modifié grâce au changement de formule de l'entropie par rapport à ID3 standard (Shannon) alors que nous n'avons relevé aucune différence entre ces deux algorithmes en terme de performance.

Les différences de résultats observées avec les résultats de l'article peuvent être dues à un échantillonnage différent et des opérations de transformation des données différentes ("preprocessing").

Comme énoncé dans l'article, l'utilisation de l'entropie de Havrda et Charvat à la place de celle de Shannon permet de diminuer la complexité de l'arbre (nombre de nœuds et feuilles) et ainsi diminue le temps d'exécution de l'algorithme.

5 Comparaison des résultats

Pour déterminer la valeur optimale de alpha, nous avons dans un premier temps testé différentes valeurs manuellement puis nous avons cherché un moyen d'effectuer ce processus automatiquement.
Un outil intéressant proposé par Weka est le meta-classifieur *CVParameterSelection*, celui-ci permet de trouver le meilleur paramètre dans une fourchette de valeurs données (exemple : en fournissant comme paramètre "A 0.1 0.9 5",

le classifieur teste les valeurs de alpha entre 0,1 et 0,9 sur 5 pas) en utilisant notamment la validation croisée.
Cependant, les différentes méthodes testées (manuel et automatique) ont montré que la valeur de alpha n'avait pas d'influence importante sur le résultat obtenu.

6 Bonus

Pour pouvoir paramétrer facilement le modèle ID3 modifié, nous avons intégré le choix de alpha dans les options proposées par l'interface de Weka comme présenté en figure 5.

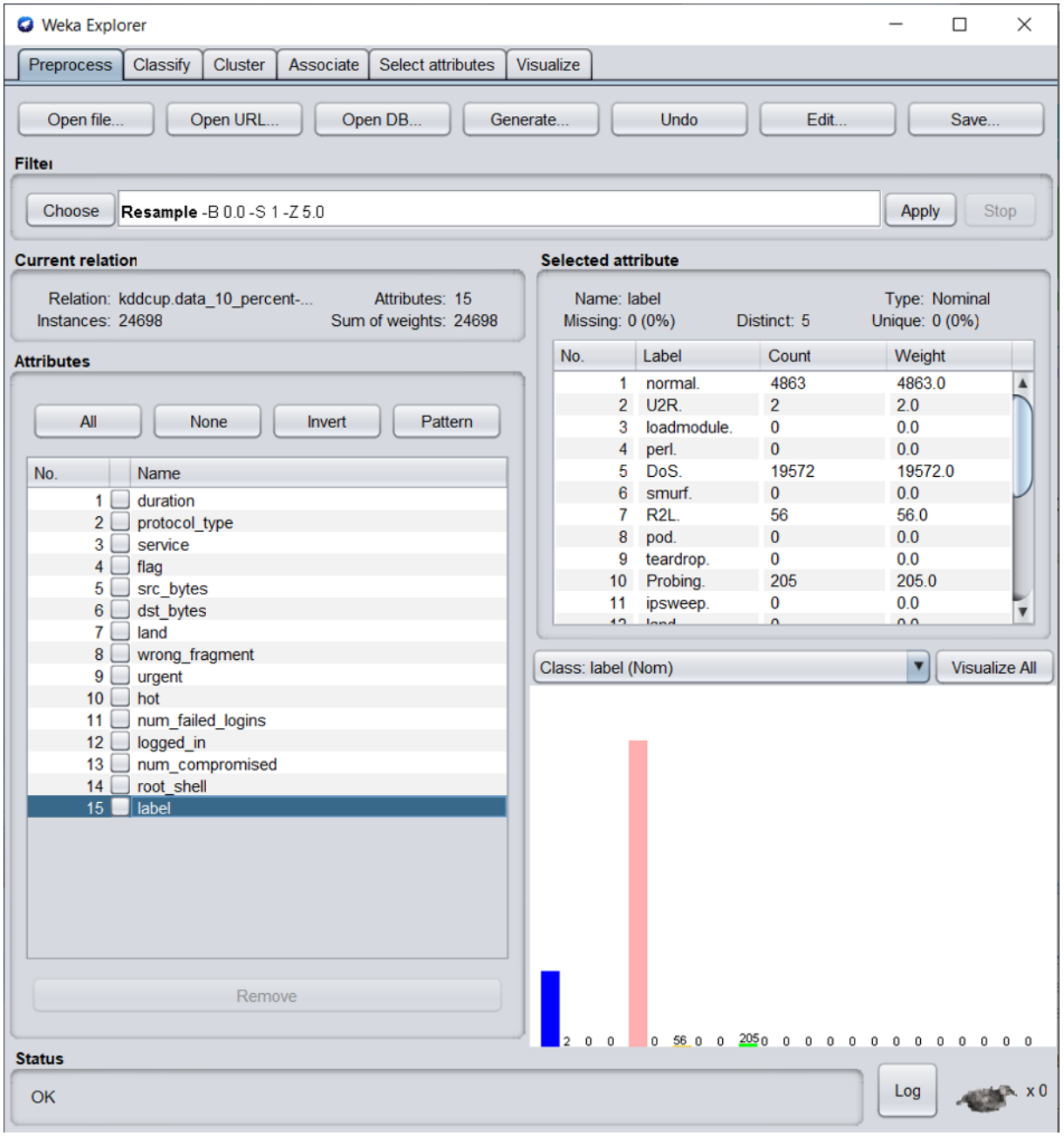

FIGURE 1 – Application du filtre *Resample*

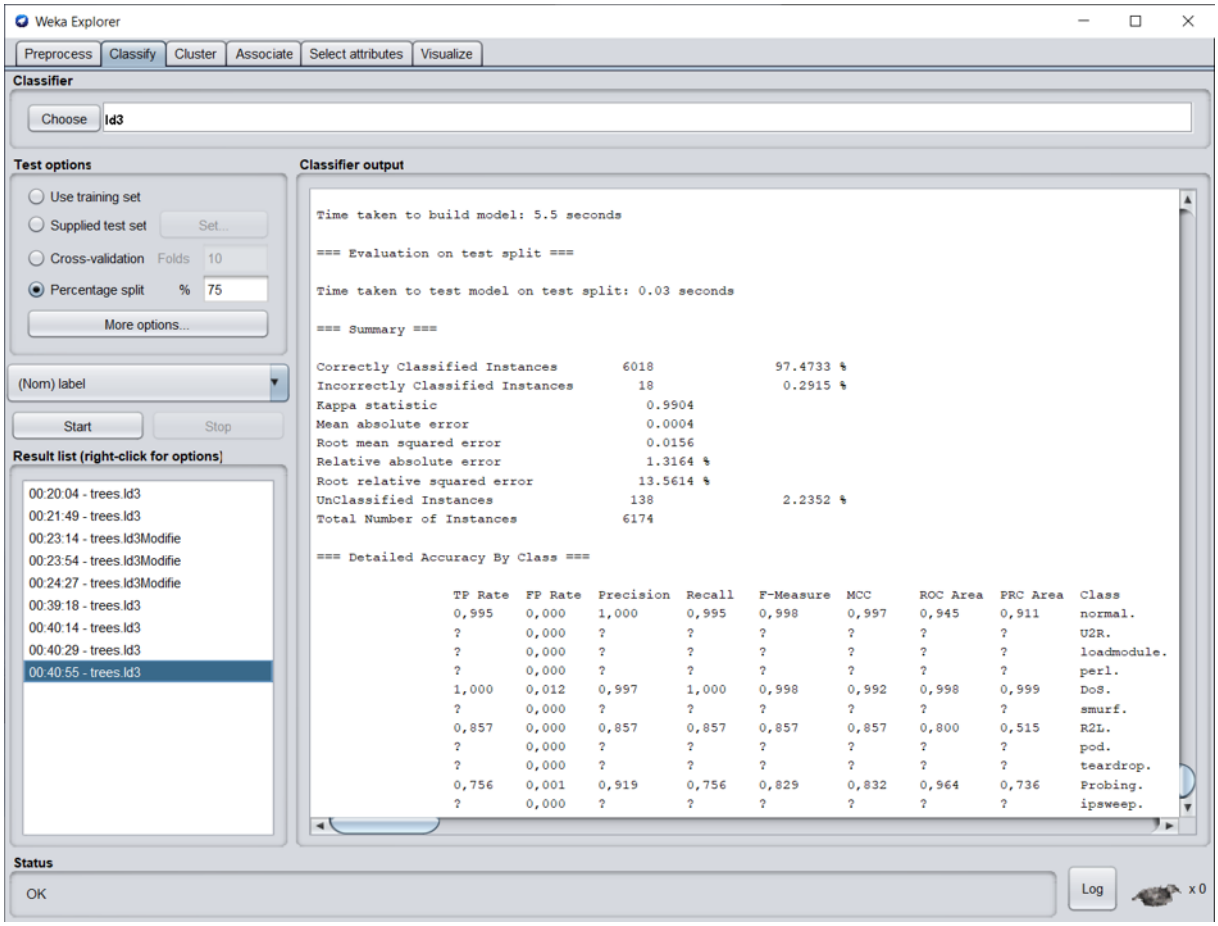

FIGURE 2 – Résultat de test avec ID3

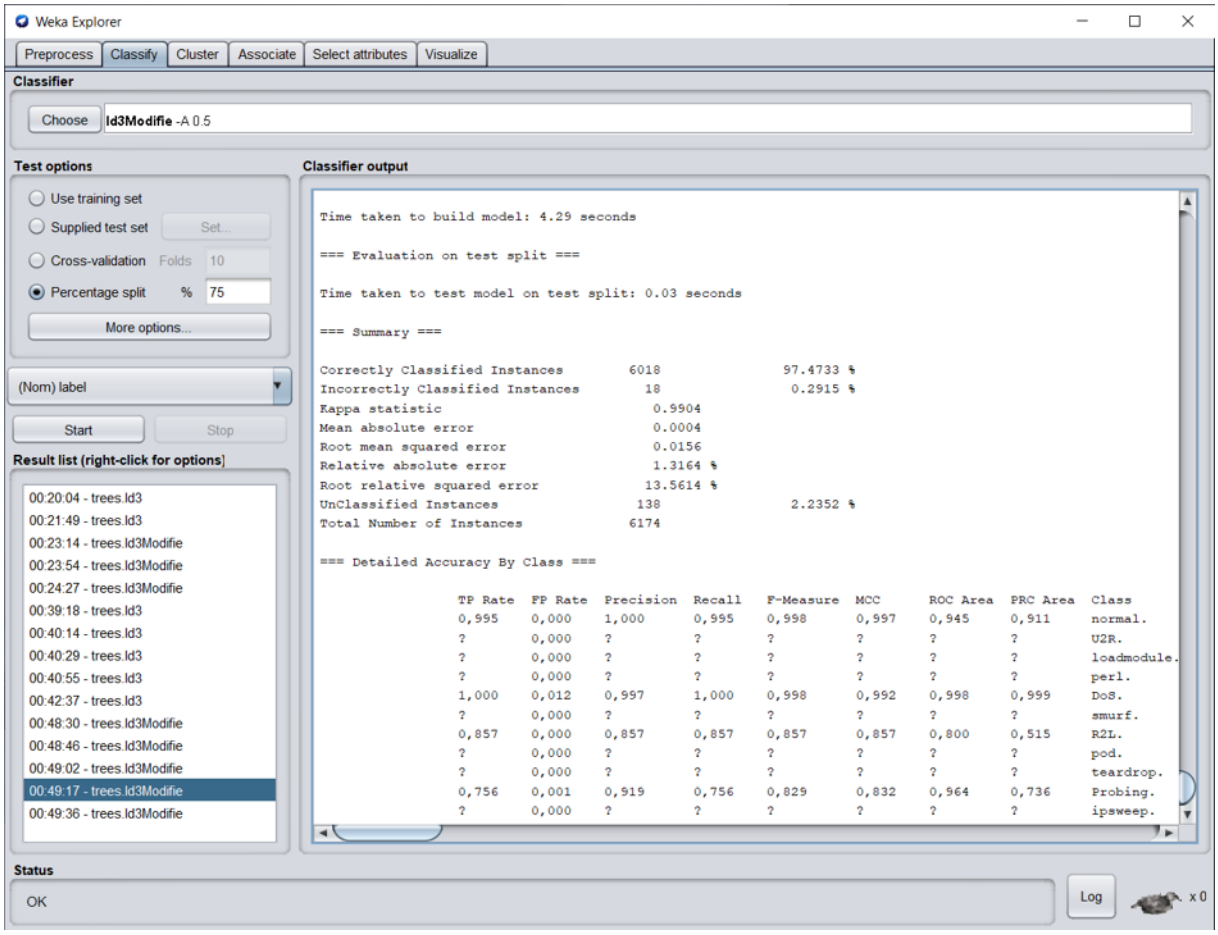

FIGURE 3 – Résultat de test avec ID3Modifie

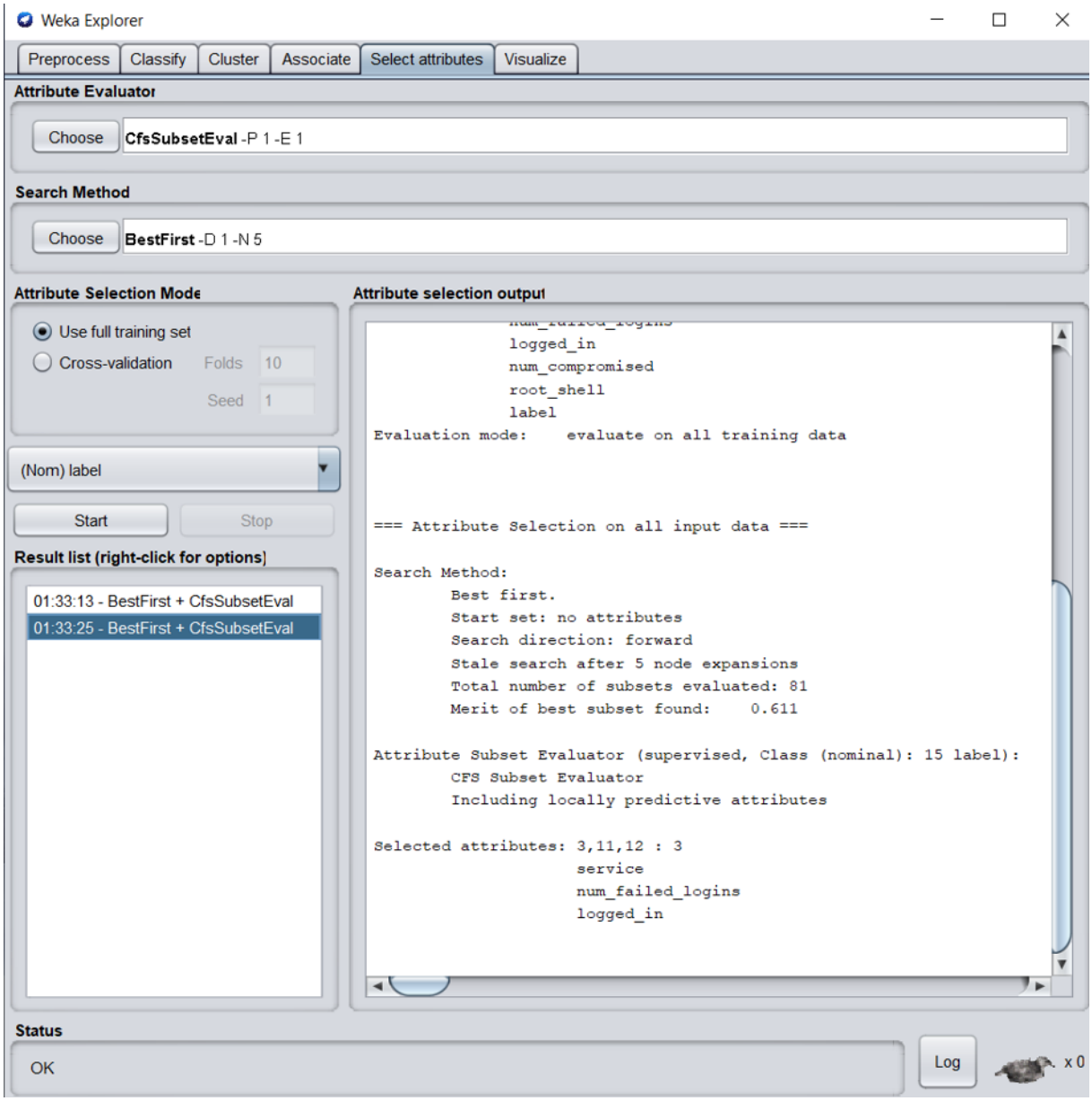

FIGURE 4 – Méthode de sélection d'attributs

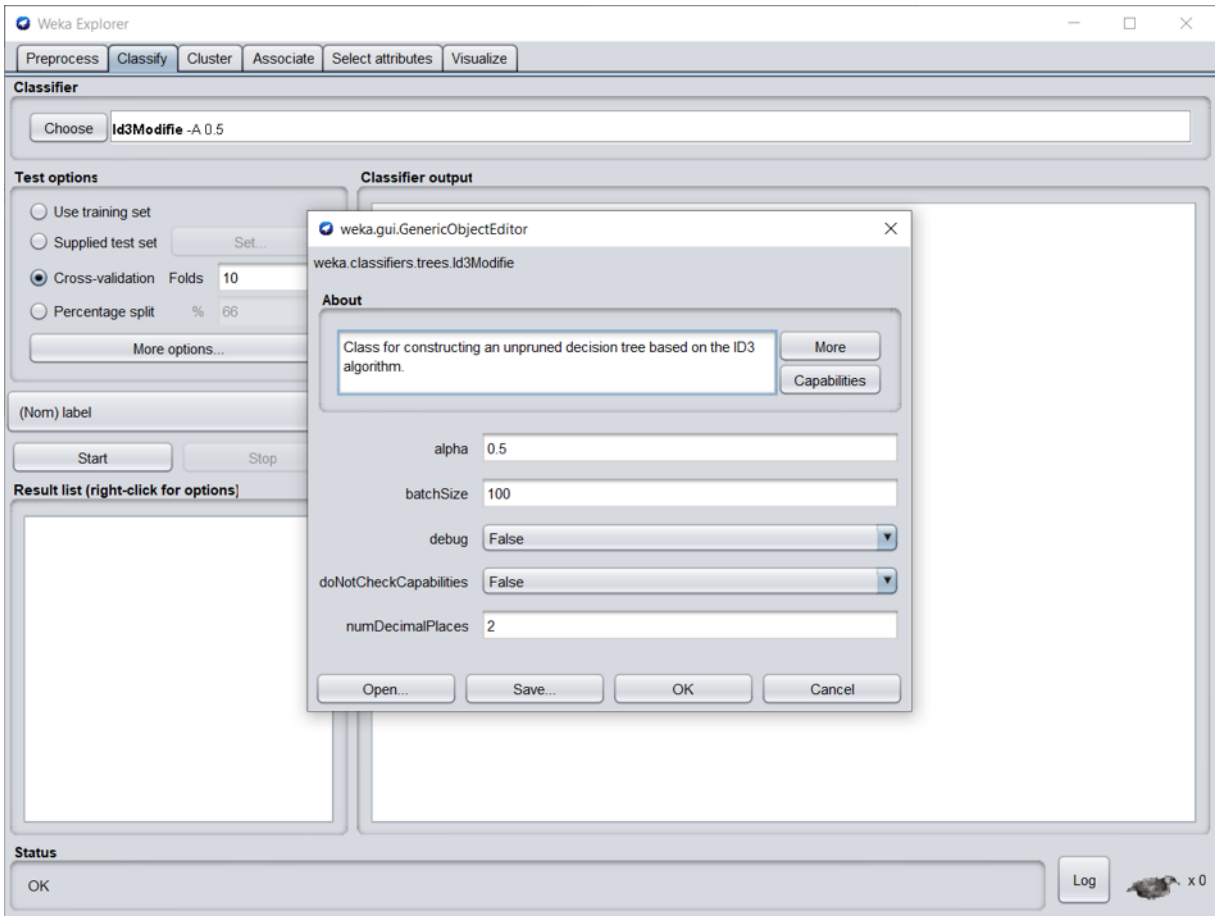

FIGURE 5 – Interface de weka avec la valeur de alpha