UQAC
Université du Québec
à Chicoutimi

Devoir #1 : Classification par arbre de décision

Session : Automne2014

Forage de données
(8INF954)

Département d'informatique et de mathématiques

Présenté à : Professeur: A.Bouzouane

Travail de : AMAMOU Houssem

Date de remise : 07/10/2014

Synthèse de l'article

Le taux d'intrusion dans le domaine des communications est en constante croissance chaque année. L'utilisation d'algorithme de forage de données est un moyen efficace d'accroître la détection des utilisations non-autorisées des ordinateurs. Plusieurs types de modèles ont été mis en œuvre dans ces algorithmes. Dans ce papier, Kumar et Jain se sont intéressé à la méthode de classification en arbre de décision basé sur des algorithmes d'apprentissage augmentant ainsi le taux de détection et réduisant les faux positifs.

L'arbre de décision est généré grâce à l'algorithme ID3 modifié. Cet algorithme construit l'arbre en question en divisant les attributs en fonction du gain maximum. Afin d'obtenir cette valeur maximale, un calcul d'entropie doit être fait. L'algorithme classique d'ID3 utilise l'entropie de Shannon. Cependant l'utilisation de cette entropie augmente le nombre de nœud, de feuille et de règles de décision ce qui augmente le temps de prise de décision. Afin de minimiser ces problèmes, l'algorithme ID3 a été modifié en utilisant l'entropie de Havrda et Charvat. Cet algorithme donne des résultats intéressants en un temps raisonnable. Grâce aux données de KDD-99, les résultats des travaux de Kumar et Jain ont pu être expérimenté sur un cas réel. Les données de cette expérience sont réparties en 5 classes, une normale et 4 comme étant « Attack ». Chaque donnée est un vecteur composé de 41 variables. L'expérience a permis de mettre en avant de meilleurs résultats pour l'algorithme proposé en comparaison avec l'algorithme ID3 classique.

Travail réalisé

Afin de réaliser ce travail, le projet Weka a été implémenté dans Eclipse et il a été modifié. Après quelques tests avec l'algorithme ID3 et J48 de Weka, on a conclu que le deuxième algorithme était beaucoup plus performant sur les données de test de KDDcup99. C'est pour cette raison qu'on a choisi d'opter pour l'utilisation de J48 et de modifier cet algorithme. Afin de procéder aux changements nécessaires, on s'est d'abord attaqué à la modification du calcul d'entropie implémenté dans J48. Dans J48 classique, les développeurs ont choisi d'utiliser l'entropie de Shannon. Après la lecture de l'article « Intrusion Détection and Classification Using Improved ID3 Algorithm of Data Mining », une modification de cette entropie s'avère être nécessaire afin de la rendre plus efficace dans le cas présent qui la détection des intrusion grâce à un apprentissage donnée. C'est pour cela que le choix s'oriente vers l'entropie de Havrda et Charvat qui est calculé à partir de la formule suivante :

$$H(X) = (2^{1-\propto} - 1)^{-1} (\sum_x p^{\propto}(x) - 1)$$

Afin d'effectuer ces changement, la classe weka.classifiers.trees.J48 devait être modifié afin de rajouter une fonction permettant la saisie de la variable $\propto$. Une classe Window a dû être ajouté afin de réaliser cette opération. Concernant le calcul de l'entropie la classe EntropySplitCrit et la classe EntropyBasedSplitCrit ont été modifié afin d'appliquer la formule citée précédemment. Finalement afin d'afficher le classeur modifié appelé ici j48Mod, le fichier de configuration GenericObjectEditor.props a été modifier et la ligne weka.classifiers.trees.J48Mod,\ a été ajouté. Voici quelques copies d'écran permettant de visualiser le travail effectué.

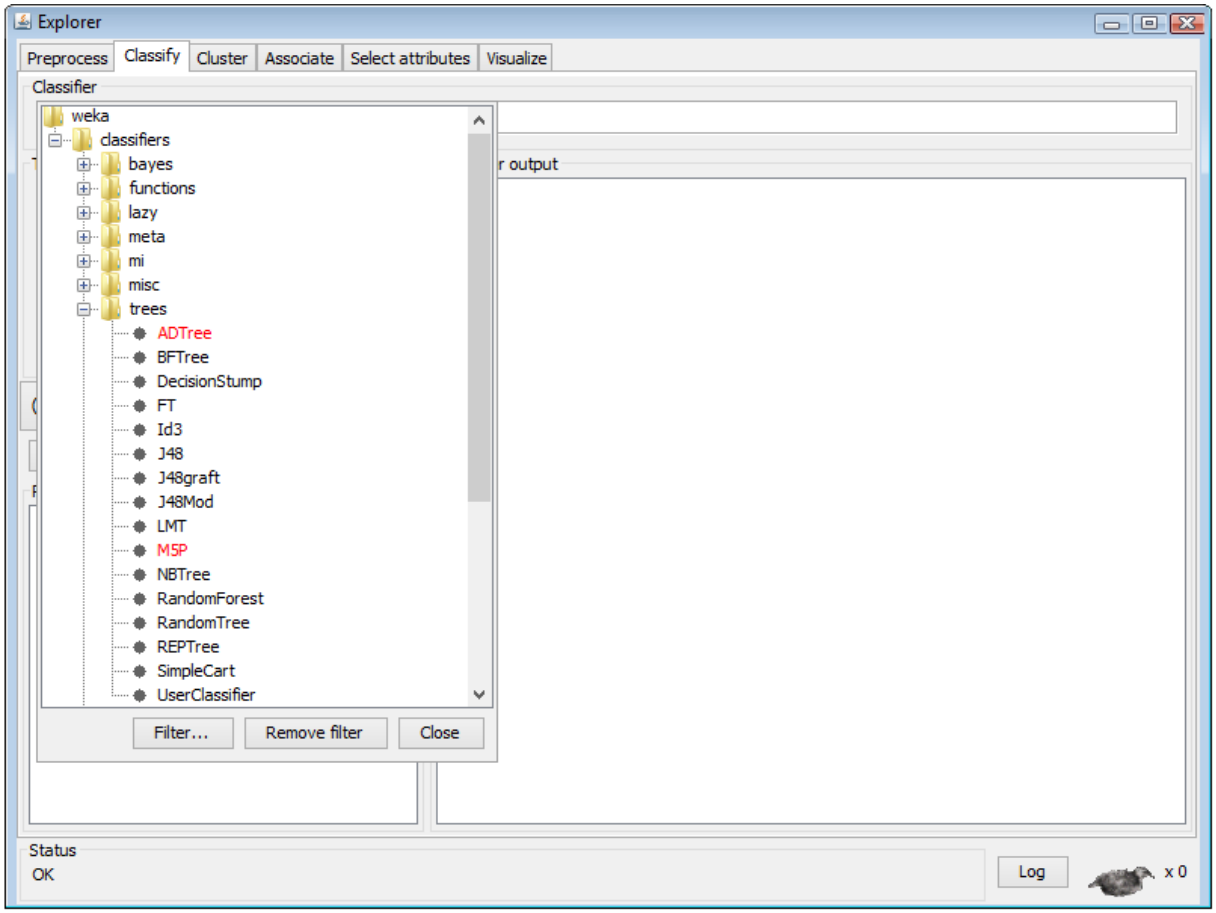

Figure1 : Le classeur J48Mod dans l'arborescence des arbres de Weka

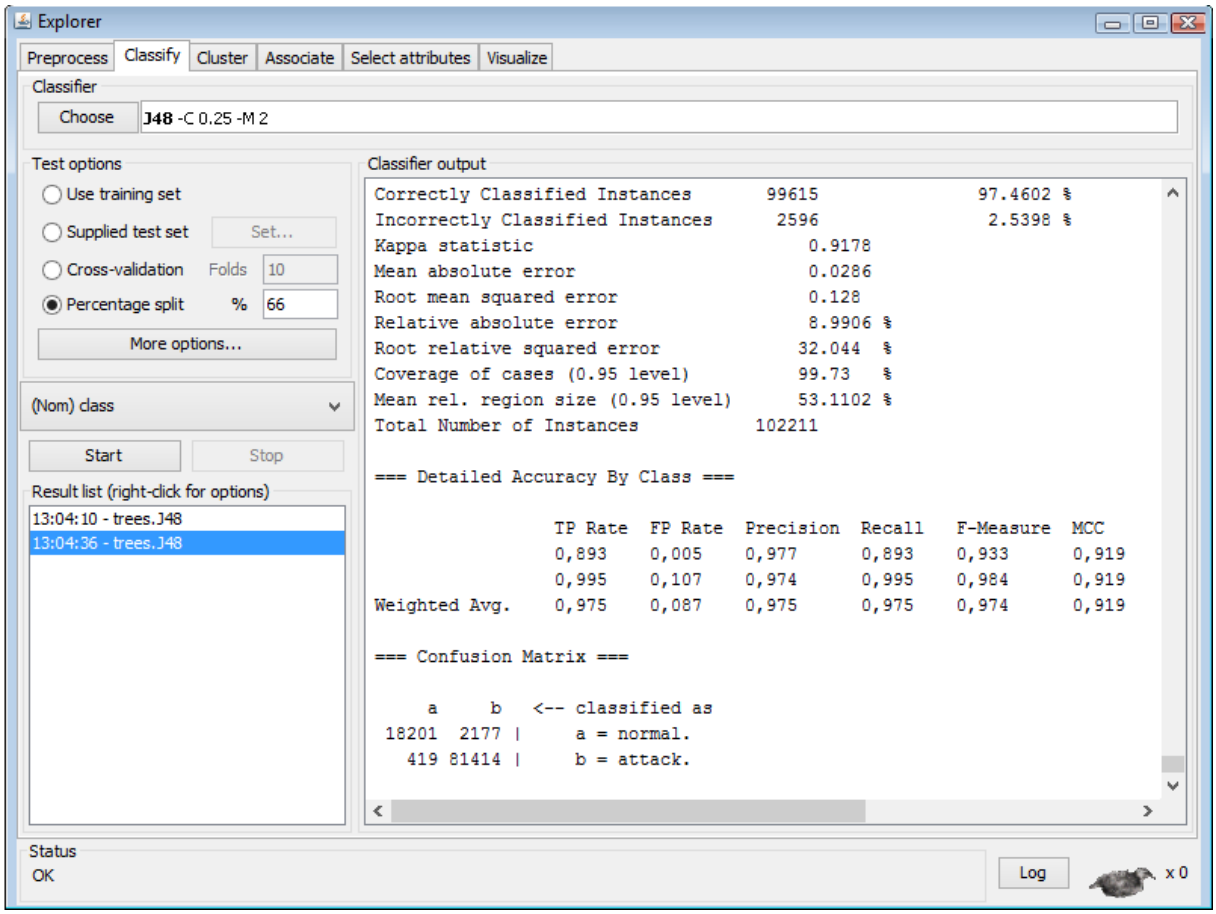

Figure 2 : Les tests réalisés avec J48 sur les données de KDD99

Grâce à la figure 1 on remarque bien que l'arbre J48Mod est présent dans l'arborescence des classeurs de Weka. La figure 2 nous permet d'illustrer le résultat des test réalisé sur les données de la KDD99, le classeur J48 a permis de classer 97,46% des cas correctement.

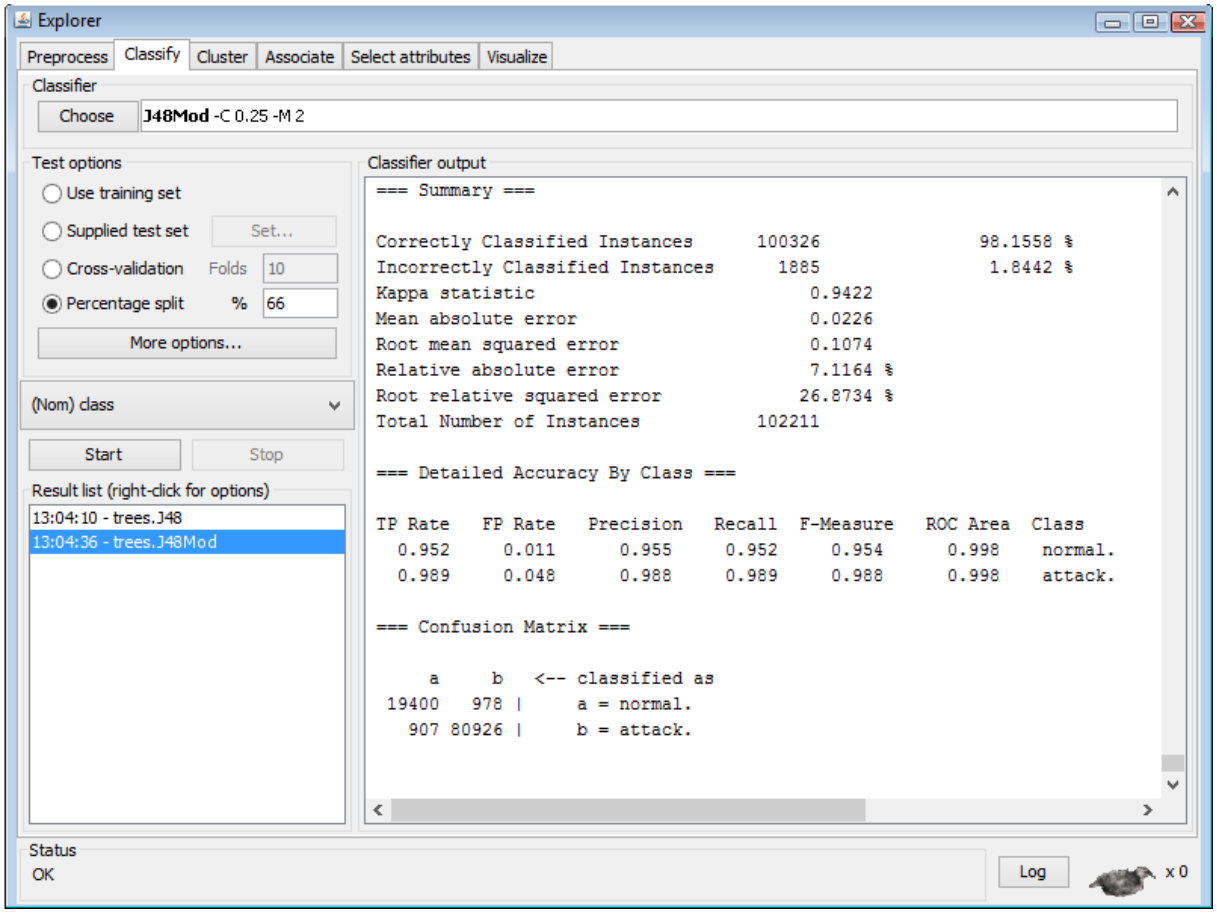

Figure 3 : Les tests réalisés avec l'algorithme J48 modifié

On remarque en effectuant ce test avec une valeur $\propto$ =3 que les instances classées correctement ont légèrement augmenté atteignant une valeur de 98,155%. Par la suite, on va montrer quelques tests réalisés avec différentes valeurs de $\propto$.

Pour la figure 4 nous avons effectué les tests avec une valeur de $\propto$ égale à 0.5. On a obtenu un pourcentage d'instance classé correctement égale à 98,18%

Pour la figure 5 nous avons effectué les tests avec une valeur de $\propto$ égale à 7. On a obtenu un pourcentage d'instance classé correctement égale à 98,13%

Nous remarquons que le changement de la valeur de $\propto$ ne se répercute par d'une grande façon sur les résultats des tests. On a remarqué qu'une légère variation entre ces résultats.

Afin d'utiliser l'algorithme implémenté, l'utilisateur n'a qu'à choisir l'arbre de classification J48Mod dans les choix d'arbre proposé par Weka et lancer la classification.

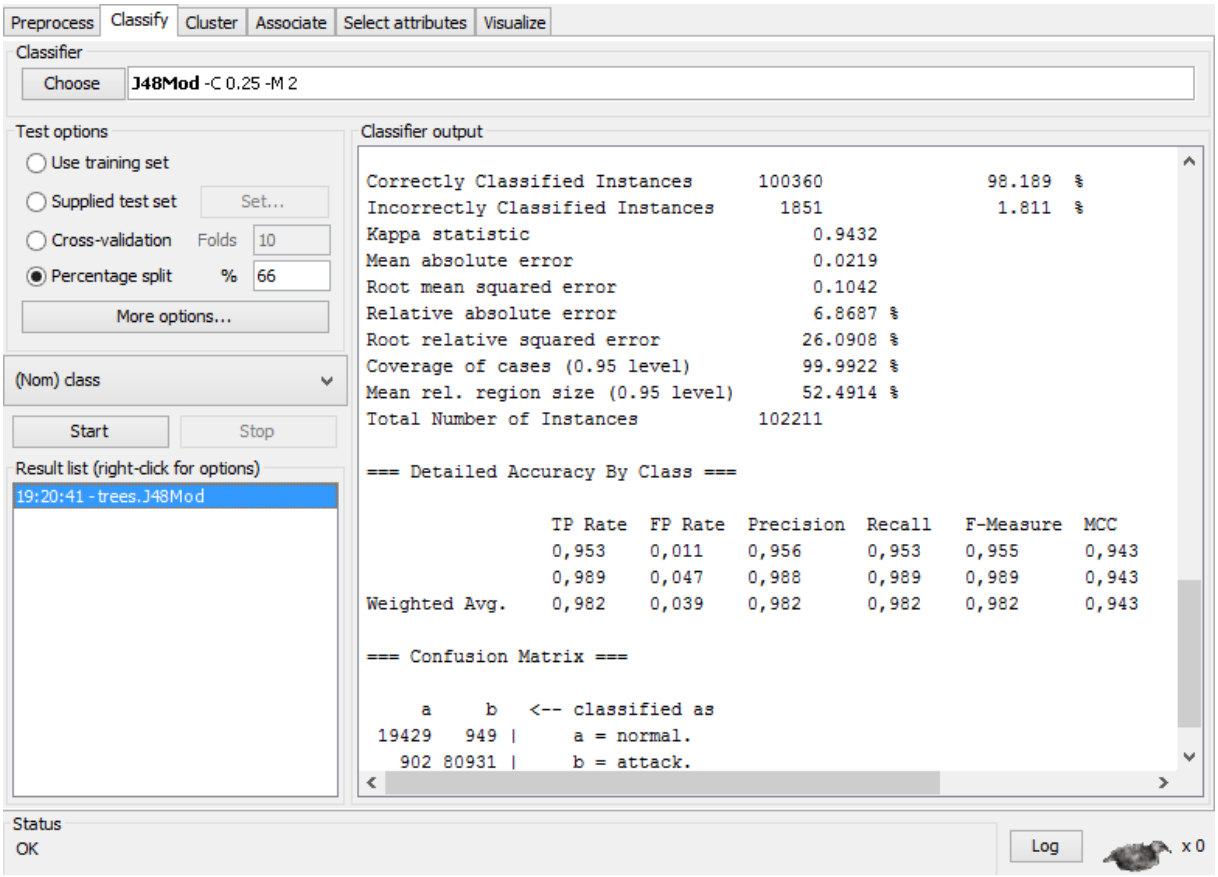

Figure 4 : les tests réalisés avec ∝=0.5

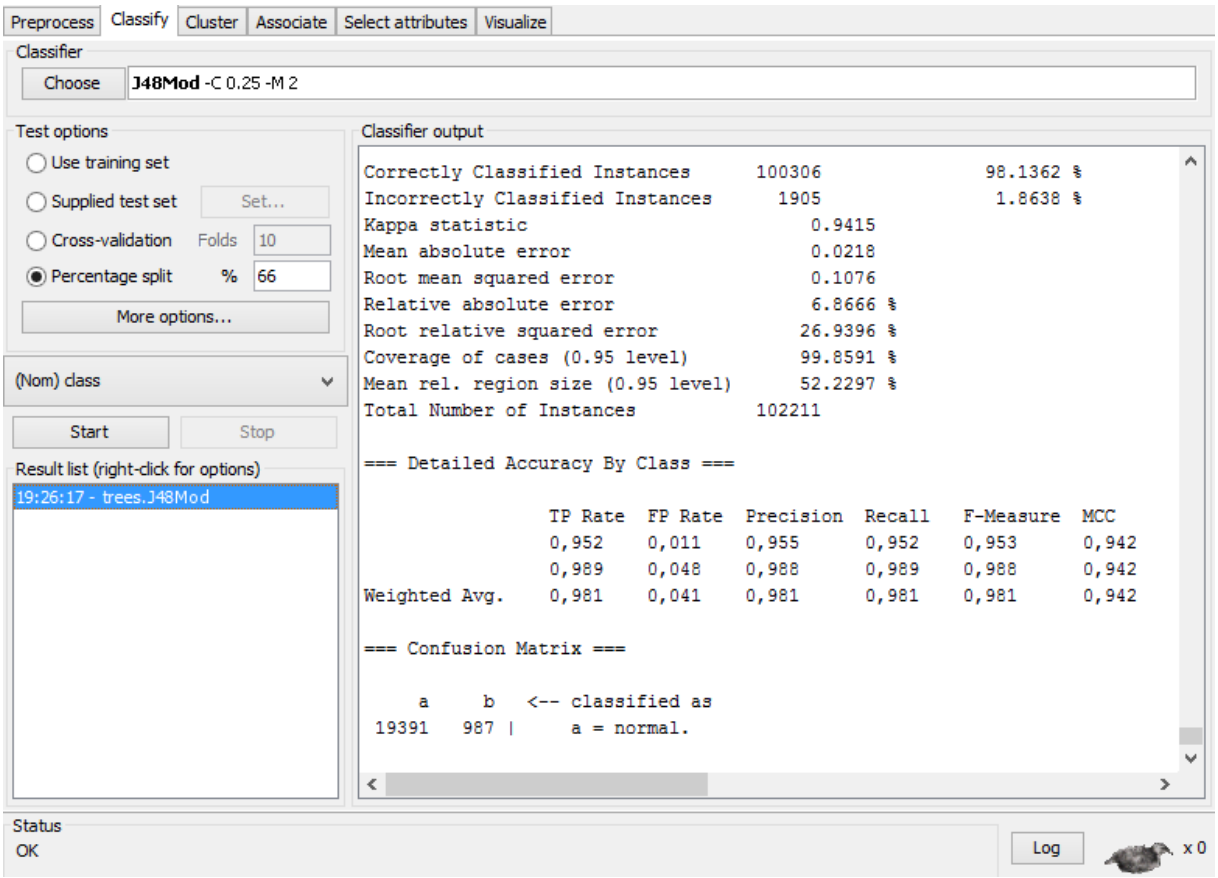

Figure 5 : les tests réalisés avec ∝=7

Afin de modifier la valeur de ∝ selon le choix de l'utilisateur, nous Avon implémenté une interface graphique permettant d'introduire la valeur correspondante la figure ci-dessous permet de visualiser cela.

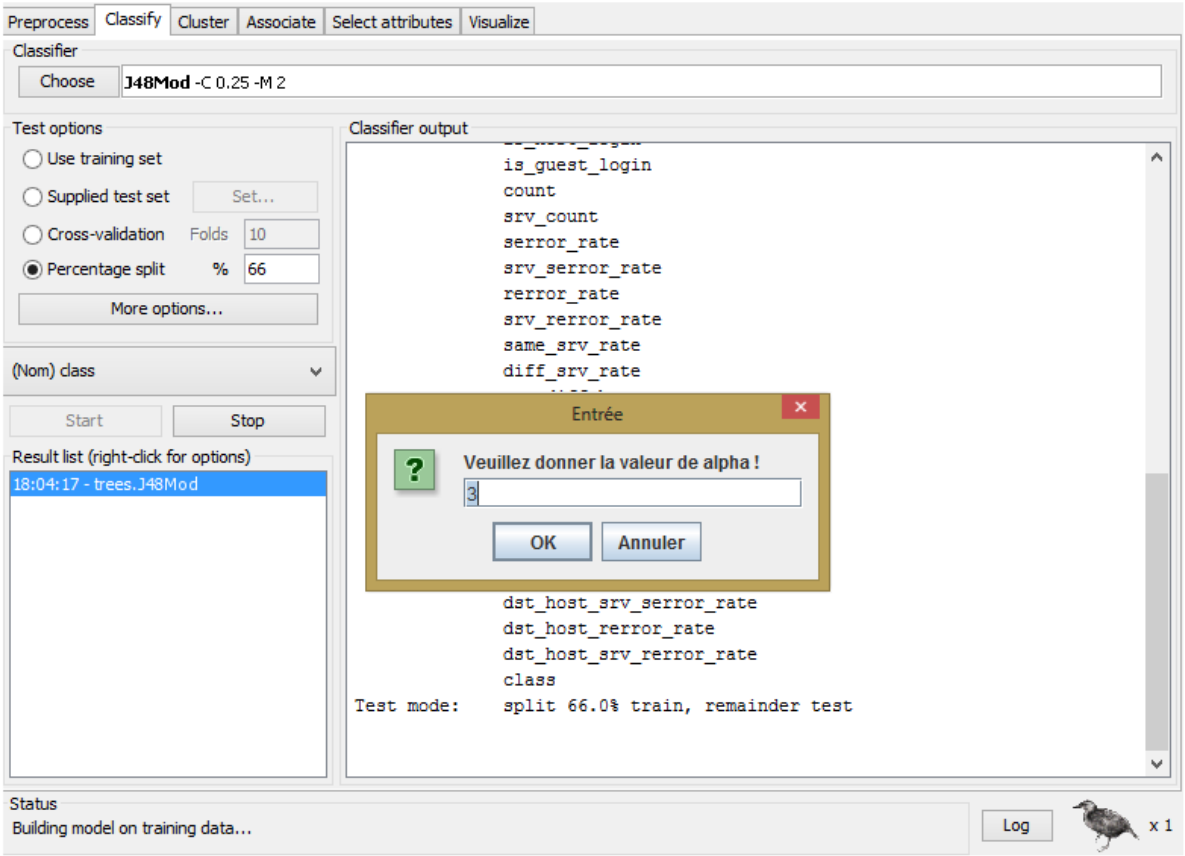

Figure 6 : Choix de la valeur de ∝

Entropie d'Harvat et Charvat (réponse à la question 2.4)

Les tests que nous avons effectués ont permis de conclure que la valeur de ∝ n'a pas une grande influence sur les résultats obtenus. Les résultats les plus pertinents obtenus ont été pour ∝=0,5, donc on peut en conclure que 0<∝<1 peut être l'intervalle optimal de ∝.

L'entropie d'Harvat et Charvat donne une meilleure solution en un temps raisonnable car la nouvelle équation de l'entropie permet de minimiser l'erreur par rapport à l'équation de Shannon. Elle permet aussi d'obtenir un arbre ayant moins besoin d'être élagué et donc amoindrir le temps de l'exécution de l'algorithme.