Université de Lorraine – Master 2 informatique – Spécialité IL

TP3

Classifieurs – Méta-classifieurs – Comparaison de (méta-) classifieurs

I) Classifieurs et méta-classifieurs

On va travailler avec les données **soybean.arff**. Une ligne de ce jeu de données décrit de nombreuses caractéristiques d'un plant de soja et d'une maladie associée (indiquée par des experts agronomes).

1) Explorer les données pour noter les principales caractéristiques dans votre compte-rendu (cf TP1)

2) Appliquer l'algorithmes **J48** puis **Jrip** (qui construit un ensemble de règles de classification mais non structurées sous forme d'un arbre)) pour construire un classifieur pour l'attribut classe et évaluer le taux d'erreur mais aussi le F-score :
   - sur l'ensemble d'apprentissage
   - puis par validation croisée à 10 folds.

   Il est intéressant de comparer les performances moyennes mais aussi par classe.

3) Faites de même avec deux méta-classifieurs : **RandomForest** (avec les classifieurs ***random tree*** puis **JRip**) et **adaBoostM1** (avec les mêmes classifieurs).

   Faire un bilan des performances (moyennes et par classe) et comparer par rapport aux classifieurs simples.

4) Quel pourrait être l'inconvénient des méta-classifieurs en pratique ?

5) Est-ce que la validation croisée à 10 folds suffit pour avoir une bonne estimation du taux d'erreur d'un classifieur ? Pourquoi ?

II) Comparaison statistique de classifieurs – Mode *Experimenter*

La comparaison de (méta-) classifieurs s'avère rapidement longue à mettre en place avec l'outil Explorer qui est plutôt conçu pour tester un algorithme de classification de manière indépendante. Lorsque vous avez choisi les algorithmes que vous souhaitez comparer, vous pouvez planifier des expériences et comparer de manière automatique les résultats obtenus. Vous pouvez pour cela utiliser le mode **Experimenter** de Weka.

En quelques mots, ce mode offre l'accès à une nouvelle interface composée de trois onglets principaux : **Setup / Run / Analyse.**

L'onglet **Setup** permet de planifier une expérience avec son évaluation (par exemple, 10 répétitions de validation croisée à 10 folds) sur un jeu de données afin de comparer les performances de plusieurs classifieurs. On peut également appliquer un ou plusieurs algorithmes de classification (supervisée) sur plusieurs jeux de données (préparés différemment, comportant des ensembles de variables différents etc.)
Lorsque vous avez paramétré correctement votre expérience, vous pouvez la sauvegarder afin de pouvoir la réutiliser ultérieurement.
L'onglet **Run** permet d'exécuter une expérience et de mémoriser les résultats.
Enfin, l'onglet **Analyse** vous permet d'analyser l'expérience effectuée (ou sauvegardée) selon différentes métriques de qualité du classifieur. Il s'agit de réaliser un test statistique pour décider quelle combinaison algorithme-jeu de données donne, pour la métrique choisie, de meilleurs résultats ou de moins bons résultats que la combinaison *baseline* (la première lors de la configuration de l'expérience).

Vous trouverez sur Arche des liens vers des documentations de l'environnement Experimenter de Weka. Prenez le temps de lire les 7 diapos sur Slideshare et d'ouvrir le manuel plus complet (40 pages) pour y accéder si besoin.

1) Se familiariser avec Experimenter en réalisant 10 répétitions de validations croisées à 10 plis pour estimer l'erreur de classification des programmes **JRip** et **J48** sur les données **soybean.arff**. Sauvergarder le résultat dans un fichier CSV et l'ouvrir avec un tableur. Assurez-vous d'avoir compris ce que Experimenter a fait en regardant le contenu de ce fichier.

2) Evaluer les performances de **J48, JRip, RandomForest, Bagging de J48, Bagging/JRip, adaBoostM1/J48, adaBoostM1/JRip** sur les problèmes de classification des données **soybean** (10 répétitions de validation croisées à 10 folds).

3) Qu'indique le t-test apparié avec une confiance de 95% ? Le premier classifieur de la liste (J48) sert de *baseline*. **Rédiger clairement les conclusions de vos expériences et tests.**

III) On passe sous Python/ScikitLearn

Chsoisir deux questions dans les parties I et II et y répondre en écrivant un notebook Python en utilisant les librairies ScikitLearn, Pandas, MatPlotLib (cf. TP2). Expliquer ce qui change quand on passe sous Python (pré-traitement des données mais aussi programmes utilisés) et comment vous gérez ce passage.

Annexe : Prédiction post-entrainement d'un modèle de classification

Vous pouvez utiliser l'interface graphique *Explorer* de WEKA ou utiliser l'interface en ligne de commande (*Simple CLI*). Pour ce faire il est possible d'utiliser l'interface Explorer pour créer la liste des paramètres et les utiliser dans la ligne de commande.

L'objectif est d'entrainer un programme à apprendre les différentes espèces animales (mamifères, poissons, insectes…). Le fichier à utiliser est **zoo.arff** (disponible sur Arche).

1. Ouvrir le fichier dans WEKA (Explorer).
2. Si vous repérez des attributs inutiles ; il est possible de les supprimer par le bouton *Remove*. Quels attributs avez-vous supprimé ?

   Sauvegarder le fichier après suppression des attributs (nouveau nom : zoo-light.arff).

3. Sélectionner le classifieur J48 (classify/trees/) pour construire un arbre de decision à partir de vos données. Utiliser 66% des animaux pour entrainer l'arbre et 34% pour évaluer (choisir l'option de test ***percentage split*** *66%*).
4. Construire l'arbre de decision en appuyant sur le bouton *start*. Le résultat apparait à gauche comme une ligne dans la liste *history*. L'estimation de qualité predictive de l'arbre est donnée dans le paneau à droite (matrice de confusion et différentes métriques). Afin de visualizer l'arbre, clic droit sur la ligne *history* et *visualise tree*. Dans la fenêtre qui s'affiche, le menu contextual (clic droit) vous permet de d'ajuster la taille de l'arbre.
5. Vous pouvez noter le poucentages d'instances correctement classifiées, les VP, les VN, les FP. Répéter l'expérience en utilisation l'option de test ***Use training set***. Les métriques sont ici très optimistes et représenent la limite supérieure qu'un arbre de décision peut atteindre.

   Répéter l'expérience avec l'option ***10 folds cross-validation*****.** Quelle est la performance de l'arbre construit (est-il bon ou mauvais ?). est-ce que la performance dépend de la sélection aléatoire des données ?

6. Prédiction de nouvelles instances avec l'arbre de décision

   L'interface graphique de WEKA est agréable pour visualiser les résultats et positionner les paramètres à l'aide de formulaires mais quand il faut construire un modèle de classification et le tester sur de nouvelles instances sans avoir à le reconstruire, il faut utiliser le mode ligne de commande (vous pourrez inclure ces lignes de commande dans vos programmes futurs). Un fichier **zoo_test.arff** (déposé sur Arche) sera votre nouveau jeu de test. Vous pourrez noter que la valeur du dernier attribut (*type*) est inconnu (?). Il est important que les fichiers d'apprentissage et de test aient exactement les même attributs.

7. La commande qui permet de générer le modèle de classification est :

java weka.classifiers.trees.J48 -C 0.25 -M 2 -t directory-path/zoo-removed.arff -d directory-path/zoo.model

Les options -C 0.25 et -M 2 correspondent aux options choisies pour le programme J48. L'option -t spécifie le chemin pour le fichier d'apprentissage ("zoo.arff"). Chemin à adapter à votre cas. L'option -d spécifie the nom et la localisation du modèle qui sera construit.

8. Une fois que le modèle a été stocké dans le fichier "zoo.model" on peut l'appliquer à de nouvelles instances. La commande correspondante :

   java weka.classifiers.trees.J48 -p 17 -l directory-path/zoo.model -T directory-path/zoo_test.arff

   L'option -p 17 indique que nous voulons prédire l'attribut numéro 17 (*type*). L'option -T option spécifie le chemin pour le fichier de test. Cette commande fournit un tableau de résultats en 4 colonnes This command results in a 4-column output similar to the following:

inst#	actual	predicted	error	prediction
0	1:?	1:mammal		0.75
1	1:?	2:bird	+	0.7272727272727273
2	1:?	1:mammal		0.95
3	1:?	3:reptile	+	0.8813559322033898

La première colonne est le numéro de l'instance dans le fichier de test. La deuxième est la valeur réelle du type de l'instance. Dans notre cas, il n'y pas de valeur, alors la valeur est ? et l'instance est étiquetée avec la classe majoritaire (1). La 3ème colonne est la valeur prédite pour le type, la colonne error contient un + quand la valeur prédite est différente de la vraie classe (la classe majoritaire). La dernière colonne donne la confiance dans le type prédit pour l'instance .

12. Qu'obtenez-vous comme résultat ? à quoi correspond cette confiance par rapport à l'arbre de décision ?