TP 3 - Panorama des méthodes de fouille de données

Introduction

L'objectif de cette séance est de passer en revue les différents types de méthodes utilisées en fouille de données, que ce soit en analyse exploratoire ou en analyse prédictive.

1 Analyse exploratoire

1.1 Classification / Clustering

Dans cette première partie, vous avez à étudier les données « eurostat » téléchargées depuis le site Web d'Eurostat [1], organisme attaché à la Commission européenne. Ces données fournissent des indicateurs économiques mesurés entre 2011 et 2013 pour les 28 pays de l'Union européenne, ainsi que la Suisse, le Japon et les États-Unis. Vous trouverez un descriptif des variables étudiées dans le fichier `description.txt` de l'archive `eurostat.zip` à télécharger.

1. Convertissez le fichier `eurostat-2013.csv` au format arff reconnu par WEKA en utilisant l'*Arff viewer* accessible dans le menu *Tools*.

   Certains attributs sont favorisés par la taille du pays ; quels sont ces attributs ? À l'aide du logiciel ou du script de votre choix, divisez les valeurs de ces attributs par la population du pays. Supprimez ensuite dans le .arff les données correspondant à la population et sauvegardez le fichier .arff obtenu.

2. Il est souvent utile d'appliquer le filtre `Normalize` sur tous les attributs avant d'utiliser des méthode de *clustering*. Quel est l'effet de ce filtre sur les données ? A-t-on besoin ici d'employer ce filtre sur les données étudiées ?

3. Effectuez une analyse en composantes principales (ACP) sur les données.

   Vous pouvez utiliser WEKA en utilisant le filtre `unsupervised.attribute.PrincipalComponents` et en veillant à ce que l'attribut « Code » soit de type nominal et soit utilisé ici comme la classe à prédire [2]. Pour visualiser les instances, allez sur l'onglet *Visualize*.

   La visualisation n'étant pas le point fort de WEKA, vous pouvez utiliser au choix le logiciel GGOBI. Pour ce faire, ouvrez le fichier `eurostat-2013.ggobi.csv`. Sélectionnez dans le menu *Interaction* l'item *Identify*. Choisissez *Record Label* puis cliquez sur *Label all*. Réalisez une ACP en sélectionnant dans le menu *Tools* l'item *Sphering (PCA)*. Sélectionnez l'ensemble des variables et cliquez sur *Update screen plot* puis sur *Apply sphering, add PCs to data*. Vous

1. `http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home/`
2. « Code » n'est pas à proprement parlé une classe mais cela permet de distinguer les instances grâce à la couleur lors de leur visualisation.

pouvez alors visualiser les instances suivant les 2 axes principaux en revenant dans la fenêtre affichant le graphique puis en sélectionnant *PC1* et *PC2* comme axes X et Y.

Joignez au rapport l'affichage des instances suivant les 2 axes principaux obtenus soit avec WEKA, soit avec GGOBI.

4. Que représentent les 4 premières composantes principales de l'ACP ? Commentez le graphique obtenu pour l'ACP en discutant la proximité entre les pays. Combien de groupes de pays définiriez-vous sur ces données ?
5. Reprenez le fichier `.arff` tel qu'il était avant la transformation des données par l'ACP. Réalisez un *clustering* avec la méthode des k-moyennes (modèle `SimpleKMeans` sous WEKA), où k représente le nombre souhaité de *clusters*. En fixant k à 8, commentez les profils des groupes contenant la Grèce, la France, l'Allemagne et enfin les États-Unis.
6. Réalisez un *clustering* au moyen d'une méthode hiérarchique ascendante en fixant le nombre de *clusters* à 1 (paramètre `numClusters`) et en utilisant un lien de type Ward (paramètre `linkType`). De manière à voir une étiquette pour chaque individu, il est nécessaire de définir l'attribut « code_pays » comme étant de type *string*. Vous pouvez visualiser l'arbre obtenu en sélectionnant *Visualize tree* dans le menu contextuel de la sortie du modèle. La visualisation n'étant pas le point fort de WEKA [3], vous pouvez employer l'outil T-REX disponible sur le Web [4] qui permet de voir sous forme graphique la sortie textuelle faite par WEKA dans le format Newick.

   Joignez au rapport l'affichage de l'arbre.
7. Commentez l'arbre obtenu précédemment en discutant des similitudes entre pays. À quel niveau de l'arbre feriez-vous une coupure ? Combien de groupes de pays seraient alors créés ?

1.2 Recherche de règles d'association

Cette partie aborde la recherche de règles d'association, en illustrant son principe par l'algorithme *Apriori* sur deux ensembles de données : « weather » et « vote ».

Apriori opère sur des données dont tous les attributs sont nominaux, les attributs numériques devant être discrétisés auparavant. En prenant comme exemple les données `weather.nominal.arff` portant sur la décision de faire un jeu en fonction des conditions météorologiques, l'algorithme génère 10 règles, ordonnées selon leur mesure de confiance donnée entre parenthèses à la fin de chaque ligne. La mesure de confiance représente le pourcentage d'instances que la règle permet de prédire correctement. Le nombre suivant la partie gauche des règles représente quant à lui le nombre d'instances qui respectent cet antécédent. Le nombre suivant la partie droite, la conclusion, représente le nombre d'instances satisfaisant complètement la règle.

La découverte de règles d'association est susceptible de produire un nombre très important de règles, dont la plupart sont peu pertinentes pour le problème. Afin de limiter le nombre de résultats obtenus, l'algorithme est basé sur un niveau de couverture minimal, c-à-d la proportion des instances couvertes par chaque règle. Pour des raisons d'efficacité algorithmique, la méthode *Apriori* commence par chercher des règles à un item (restreint à la classe à prédire), puis étend celles qui ont une couverture suffisante avec une valeur d'attribut pour obtenir des règles à deux items. Elle effectue ensuite de nouvelles itérations en ajoutant à chaque fois un nouvel item.

3. *Bis repetita...*
4. Se rendre sur `http://www.trex.uqam.ca/` et sélectionner le menu *Tree viewer*.

Le niveau de couverture est très dépendant des données et difficile à fixer. Il est par conséquent accompagné de plusieurs paramètres dans WEKA. Le premier, `numRules`, précise le nombre maximal de règles souhaitées. La méthode *Apriori* commence alors à rechercher des règles qui satisfont un niveau de couverture minimal intial (`upperBoundMinSupport`). Dans le cas où `numRules` n'est pas atteint, ce niveau de couverture minimal est diminué d'une valeur fixée par `delta` et on recherche alors de nouvelles règles. Ce processus est répété jusqu'à ce que l'on obtienne `numRules` règles ou que l'on atteigne un certain niveau minimal (`lowerBoundMinSupport`).

1.2.1 Recherche des règles dans un ensemble simple

1. Appliquez *A priori* disponible au niveau de l'onglet *Associate* avec ses paramètres par défaut sur l'ensemble `weather.nominal.arff`. Expliquez la valeur affichée du nombre de cycles effectués.
2. Quel est le support de la règle suivante : outloook=sunny $\wedge$ temperature=hot $\Rightarrow$ humidity=high ?
3. Quel est le nombre de règles pouvant être retournées pour chaque combinaison de valeurs indiquées dans le tableau 1 ? Commentez la variation du nombre de règles retournées en fonction des valeurs utilisées.
4. *Apriori* possède d'autres critères d'ordonnancement que celui de la confiance : *lift*, *leverage* et *conviction*. Définissez chaque métrique pour une règle sous la forme $A \Rightarrow B$ à partir des probablités de A, B, $!B$, $A \cap B$ et $A \cup B$.
5. Quelle est la meilleure règle pour l'ensemble « weather » pour chacune des métriques *lift*, *leverage* et *conviction* ?

Confiance minimale	Support minimal	Nombre de règles
0.9	0.3	
0.9	0.2	
0.9	0.1	
0.8	0.3	
0.8	0.2	
0.8	0.1	
0.7	0.3	
0.7	0.2	
0.7	0.1	

TABLE 1 – Nombre de règles pour différentes valeurs de paramètres.

1.2.2 Recherche des règles dans un ensemble de données réaliste

On considère maintenant un ensemble de données plus réaliste, `titanic.arff`, qui fournit des informations sur des membres de l'équipage et des passagers du navire du Titanic, tristement célèbre pour son naufrage intervenu en avril 1912.

1. Exécutez *Apriori* sur ces données avec les paramètres par défaut. Commentez les 2 premières règles générées.

2. Augmentez le nombre de règles générées à 25. Quelles règles pouvez-vous établir quant aux chances de survie d'un individu embarqué sur le Titanic ?

2 Analyse descriptive

2.1 Classement

Dans cette partie du TP, vous allez étudier et comparer le comportement de différents classifieurs sur les données « Labor ». Ces données concernent les résultats finaux de négociations du travail menées dans l'industrie au Canada en 1987 et 1988. Elles incluent tous les accords collectifs dans le secteur des affaires et des services à la personne pour des organisations locales d'au moins 500 membres (enseignants, infirmiers, employés d'université, police, etc.).

1. Les différents filtres et classifieurs de WEKA peuvent être exécutés en ligne de commande, sans avoir à employer l'interface graphique de l'*Explorer*. Cela permet d'automatiser l'application de différents classifieurs sur des données au moyen de scripts.
   À titre d'exemple, pour exécuter un simple classifieur J48 sur un ensemble de données `data.arff`, il suffit de lancer la commande `java weka.classifiers.trees.J48 -t data.arff`. Veuillez initialiser au préalable la variable d'environement `CLASSPATH` avec le chemin du jar de Weka (par exemple
   `CLASSPATH=/usr/share/java/weka.jar`). Pour connaître les différentes options d'un classifieur, utilisez le drapeau `-h`.
   Écrivez un script dans le langage de votre choix permettant d'automatiser l'exécution de plusieurs classifieurs (`ZeroR`, `OneR`, `J48`, `SimpleLogistic`, `SMO`), sur un ensemble de données (indiqué en tant que paramètre) et d'afficher les performances de chacun des modèles par ordre décroissant de performance.
2. Expliquez en une phrase en quoi consiste le classifieur `ZeroR`. Dans quel cas ce classifieur pourrait donner les meilleurs résultats parmi ceux testés ?
3. Testez votre script sur les données « Labor » en réalisant une validation croisée en 10 strates. Parmi les méthodes testées, quelle est la plus performante ? Quel est l'intérêt d'utiliser une validation croisée plutôt qu'un découpage des données en $\frac{2}{3}$ entraînement et $\frac{1}{3}$ test pour l'évaluation ? Quelle est la classe pour laquelle la prédiction fait le plus d'erreurs ?
4. L'ensemble de données « Labor » contient des données manquantes pour certains attributs, repérées par « ? ». Appliquez le filtre non supervisé sur les attributs `ReplaceMissingValues`. Quel est son effet sur les données ? Enregistrez les données modifiées par ce filtre et réappliquez votre script sur ces nouvelles données. Les résultats varient-ils par rapport à ce que vous aviez observé sans le filtre ?

2.2 Régression linéaire

On s'intéresse ici au modèle de régression le plus simple : le modèle linéaire. La régression linéaire permet de modéliser la relation entre une valeur aléatoire y et un ensemble de variables aléatoires x_i, sous la forme $y = \sum_i a_i x_i + u$ où u désigne le terme d'erreur.

Les données utilisées ici, disponibles dans `house.arff`, affichent le prix de vente des maisons en dollars US en fonction de plusieurs paramètres.

1. Chargez les données et appliquez le modèle `functions.LinearRegression` dans l'onglet *Classify*. Quel est le modèle de régression produit en sortie ? Quelle variable est la plus importante ? Quelle variable se révèle inutile ? Commentez le coefficient attribué à la taille d'une maison (premier attribut dans le fichier).
2. Prédisez la valeur de vente pour une maison possédant les caractéristiques suivantes : $(3198, 9669, 5, 1, 1)$ (la valeur réelle est : 217 894 \$).