TP 2 - 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 « World Development Indicators » téléchargées depuis le site Web de la Banque mondiale[1], une institution financière internationale. Ces données fournissent des indicateurs économiques mesurés en 2018 pour les 27 pays de l'Union européenne, ainsi que les pays du G20. Vous trouverez un descriptif des variables étudiées dans le fichier `Data_World_Development_Indicators2.ods`.

1. Ouvrez le fichier `Data_World_Development_Indicators2.csv` à l'aide de la bibliothèque pandas. Combien y a-t-il d'attributs ? De quels types sont-ils ?
2. Y a-t-il beaucoup de valeurs manquantes dans les données ? Comment ces valeurs sont-elles représentées dans le ficher ? Remplacez chaque valeur manquante par la médiane obtenue par chaque attribut en utilisant SimpleImputer. Quel est l'intérêt d'utiliser ici la médiane plutôt que la moyenne ?
3. Il est souvent utile d'appliquer un filtre de normalisation sur tous les attributs avant d'utiliser des méthodes de *clustering*. Quel est l'effet du filtre StandardScaler sur les données ? A-t-on besoin ici d'employer ce filtre sur les données étudiées ?
4. Effectuez une analyse en composantes principales (ACP) sur les données à l'aide de la bibliothèque scikit-learn.

   Joignez au rapport l'affichage des instances étiquetées par le code du pays suivant les 2 facteurs principaux de l'ACP, puis suivant les facteurs 3 et 4 de l'ACP.
5. Que représentent les 4 premiers facteurs de l'ACP ? Vous pourrez répondre à cette question en calculant la variation des valeurs propres en fonction du rang et en traçant le cercle de corrélation entre les facteurs et les variables originales.

   Commentez les deux graphiques obtenus à la question précédente (affichage dans les deux premiers plans de l'ACP) en discutant la proximité entre les pays.
6. Reprenez les données telles qu'elles étaient avant leur transformation par l'ACP. En utilisant KMeans de la bibiothèqe scikit-learn, réalisez un *clustering* avec la méthode des k-moyennes, où k représente le nombre souhaité de *clusters*. Tracez la variation du R^2 pour k variant de 2 à 20. D'après cette courbe, quelle valeur de k reteneriez-vous ?

1. `https://databank.worldbank.org/source/world-development-indicators`

7. En fixant k à 8, commentez les profils des groupes contenant la France, le Mexique et la Bulgarie.
8. Réalisez un *clustering* au moyen d'une méthode hiérarchique ascendante avec la classe dendogram de la bibliothèque scipy.

   Joignez au rapport l'affichage du dendogramme obtenu.
9. 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*, disponible dans la bibliothèque mlxtend.

Apriori opère sur des données dont tous les attributs sont nominaux, les attributs numériques devant être discrétisés auparavant. L'algorithme génère des règles, sous la forme *prémisses* (appelées *antecedents* dans la bibliothèque utilisée) $\Rightarrow$ *conclusion* (appelée *consequents*). Ces règles sont ordonnées par défaut selon leur support, c'est-à-dire le nombre d'instances qui satisfont 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.

Dans cet exercice, il est demandé d'étudier le fichier `grocery.csv`, qui fournit des informations sur les achats dans un supermarché de la chaîne Kroger aux États-Unis, durant un weekend. L'objectif est ici de découvrir des règles s'appliquant sur le panier de consommation.

1. Ouvrez les données à l'aide de la bibliothèque csv. Quels sont les types des attributs ? Que représentent les valeurs présentes dans le fichier ? Comment agit les commandes du script fourni, permettant la discrétisation ?
2. Le niveau de couverture est très dépendant des données. Établissez l'ensemble des itemsets fréquents à l'aide de la fonction apriori() de mlxtend en fixant un support minimal à 0,002. Expliquez ce que signifie cette valeur de 0,1. Combien d'itemsets obtenez-vous ? Donnez leur fréquence en fonction du nombre d'items qu'ils contiennent.
3. Construisez les règles à partir de l'ensemble des itemsets précédemment obtenus en recourant à la fonction association_rules() de mlxtend. Pour ne pas avoir un nombre trop important de règles générées, fixez un seuil minimal de 0,5 pour la valeur de confiance.

   Combien de règles obtenez-vous ? Commentez la première règle obtenue.
4. *Apriori* possède plusieurs critères pour évaluer la pertinence d'une règle, parmi lesquelles *confidence* et *lift*[2]. Quelle est la meilleure règle pour l'ensemble « grocery » pour chacune des métriques *confidence* et *lift* ? Commentez les scores et règles obtenus.
5. Un dirigeant du supermarché souhaite améliorer ses ventes en réorganisant le positionnement des produits dans ses rayons. D'après votre analyse du panier de consommation fourni, quels produits sont souvent achetés conjointement avec du yaourt et et du café ?[3]

2. Vous pouvez vous référer au cours de l'UCE ou bien aux informations disponibles sur la page `https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/` .

3. Vous aurez peut-être besoin de revoir la valeur auparavant fixée de support minimal.

2 Analyse prédictive

2.1 Classement

Dans cette partie du TP, vous allez étudier et comparer le comportement de différents classifieurs sur les données « marketing ». Ces données concernent l'analyse de la personnalité des clients pour aider une entreprise à mieux comprendre ses clients et à leur proposer des produits plus adaptés à leurs besoins. Plusieurs informations sont disponibles et l'ojectif final des données est d'établir si un client est susceptible d'accepter l'offre faite pour la dernière campagne marketing.

Les méthodes de classement que nous testerons sur ces données seront :

— le classifieur élémentaire associant chaque instance à la classe majoritaire (DummyClassifier avec la stratégie most_frequent),
— la méthode bayésienne naïve (GaussianNB),
— l'arbre de décision (DecisionTreeClassifier),
— la forêt d'arbres décisionnels (RandomForestClassifier)
— la méthode de régression logistique (LogisticRegression),
— le modèle SVM (SVC).

1. Ouvrez les données à l'aide de la bibliothèque pandas. Combien y a-t-il d'attributs dans ces fichers ? Quels sont leurs types ? Quelle est la variable à prédire ? Quels attributs, indiqués comme numériques, prennent deux valeurs et sont donc en réalité binaires ? Transformez ces attributs pour qu'ils deviennent catégoriels. Vous pourrez également supprimer la colonne ID qui identifie de manière unique chaque client, mais n'aide pas à la prédiction sur l'efficacité de la campagne marketing.
2. Les classes sont-elles équilibrées dans les fichiers ? Y a-t-il des valeurs manquantes dans les données ? Comment ces valeurs sont-elles représentées dans le ficher ? Quels sont les attributs concernés par des valeurs manquantes ?
3. Expliquez en une phrase en quoi consiste la méthode élémentaire de classement DummyClassifier utilisant la stratégie most_frequent. Dans quel cas ce classifieur pourrait donner les meilleurs résultats parmi ceux testés ?
4. Dans un premier temps, seules les données numériques sont considérées pour prédire la classe. Remplacez chaque valeur manquante par la moyenne obtenue par chaque attribut en utilisant SimpleImputer. Normalisez ensuite les valeurs numériques à l'aide de StandardScaler.

   Calculez les performances des 6 méthodes de classement étudiées sur les données « marketing » en réalisant une validation croisée en 5 blocs sur les données. Parmi les méthodes testées, quelle est la plus performante par rapport au taux de classification ?
5. Dorénavant, on se propose de n'employer que les attributs catégoriels. Réalisez une disjonction des variables catégorielles à l'aide de la classe OneHotEncoder de la bibliothèque scikit-learn.

   Calculez les performances de classement des 6 méthodes étudiées, en réalisant à nouveau une validation croisée en 5 blocs. Observez-vous des performances similaires à ce que vous aviez à la question précédente ?
6. Utilisez enfin l'ensemble des colonnes, c-à-d en prenant en compte les attributs numériques et catégoriels. Calculez à nouveau les performances de classement en validation croisée à 5 blocs et commentez les résultats obtenus.
7. Jusqu'à présent, la fonction accuracy_score fournie dans le script ne calcule que le taux de classification (*accuracy*). Pourquoi cette métrique ne permet-elle pas d'évaluer correctement les méthodes pour des classes déséquilibréesire. Construisez un nouveau jeu de données équilibré sur les 2 classes par cette méthode (vous pourrez utiliser la fonction resample du package

sklearn.utils ou bien RandomOverSampler de la bibliothèque imbalanced-learn). Relancez l'évaluation des methodes de classement sur ce nouveau jeu de données.