TD : Differential Privacy et Analyse de Données

Benjamin NGUYEN et Patrice CLEMENTE

STI-4A-EA Big Data 2020/2021

Temps prévu : 1h20

Dans ce TD, nous continuons de travailler avec le logiciel WEKA, et nous utilisons maintenant une bibliothèque Python sur la *differential privacy*. Nous allons nous intéresser ici à l'aspect classification uniquement.

Préparation du TD :

WEKA

Il faut disposer du logiciel WEKA (déjà installé dans le TD précédent).

IBM DiffPrivLib

Il faut installer la bibliothèque Python IBM DiffPrivLib en faisant :

```
pip install diffprivlib
```

Si l'installation automatique ne fonctionne pas, vous pouvez installer la bibliothèque à la main avec :

```
git clone https://github.com/IBM/differential-privacy-library
```

puis en ce plaçant dans le répertoire :

```
pip install .
```

Installez également le module matplotlib : `pip install matplotlib`

Fichier de données

Téléchargez également le fichier contenant les données à analyser : il s'agit d'un fichier sur les notes d'élèves de lycée au Portugal et de certaines caractéristiques sociales des élèves.

Données : https://benjamin-nguyen.fr/ENS/4ASTI-EA-BIGDATA-SECU/student-mat.csv

La signification de champs est donnée ici :

https://archive.ics.uci.edu/ml/datasets/Student+Performance

L'objectif est de prévoir la valeur de l'attribut **G3** pour chaque enregistrement

Ce fichier peut être chargé tel quel dans WEKA.

I- Analyse de données brutes

Chargez le fichier dans WEKA. Combien y-a-t-il d'attributs et combien de lignes ?

On remarque également que contrairement au fichier sur le diabète, il n'y a plus de couleurs. En effet, l'objectif ici est de prédire la valeur de l'attribut **G3** qui est un attribut numérique et non pas une classe. Ainsi si on prévoit une note de 12, alors que la note est en réalité 13, cette prédiction est meilleure que d'avoir prévu une notre de 5 alors que la note est en réalité 13. Aussi, on n'évaluera pas de la même manière que dans le TD précédent la qualité d'une prédiction.

Evaluation de la qualité

RMSE

La qualité d'une prédiction numérique se mesure par l'erreur sur chacun prédiction. Plusieurs grandeurs existent, comme valeur *la racine caarée de l'erreur quadratique moyenne* (ou Root Mean Square Error, MSE), qui se calcule de la manière suivante (*n* est le nombre d'enregistrements dans l'échantillon) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(G3 - G3est\right)^2} = \sqrt{MSE}$$

Où *G3est* représente la valeur estimé (prédite) de *G3* pour chaque ligne et *G3* représente sa valeur réelle. *MSE* est donc la somme des carrés des erreurs entre la valeur réelle et la valeur estimée et *RMSE* sa racine carée. Si on ne fait aucune erreur, alors *RMSE*=*MSE*=0. Si on se trompe toujours de 2 sur la note, alors *MSE = 4* et *RMSE*=2. Une erreur de 10% sur chaque note correspond donc à un *RMSE* égal à 2.

RRSE

Le taux d'erreur relatif (root relative square error) se calcule par :

$$RRSE = \frac{1}{n}\sum_{i=1}^{n}\frac{G3est}{G3}$$

Coefficient de correlation de Pearsons (ρ)

Notons que WEKA propose également d'autres critères tels que le coefficient de correlation de Pearsons (qui varie entre -1 et 1). On considère qu'un coefficient au dessus de 0.8 comme excellent, entre 0.8 et 0.6 très bon, entre 0.6 et 0.4 bon, entre 0.2 et 0.4 moyen et en dessous de 0.2 mauvais. La formule pour calculer ρ est la suivante (X représente l'ensemble des données exactes et Y l'ensemble des données estimées, cov la covariance et σ l'écart type) :

$$\rho = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y}$$

Nous retrouverons grâce à WEKA la valeur de ρ, *RMSE, RRSE*.

1) Classification KNN

Commençons avec le classifieur KNN classique. Lançons le avec KNN=1 et cherchons à prédire G3. Le résultat est le suivant :

```
=== Run information ===


Scheme:                 weka.classifiers.lazy.IBk   -K   1   -W   0   -A
"weka.core.neighboursearch.LinearNNSearch                              -A
\"weka.core.EuclideanDistance -R first-last\""

Relation:     student-mat

Instances:    395

Attributes:   33

              school

              sex

              age

              address

              famsize

              Pstatus

              Medu

              Fedu

              Mjob

              Fjob

              reason

              guardian

              traveltime

              studytime

              failures

              schoolsup

              famsup

              paid
```

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier

using 1 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Cross-validation ===

=== Summary ===

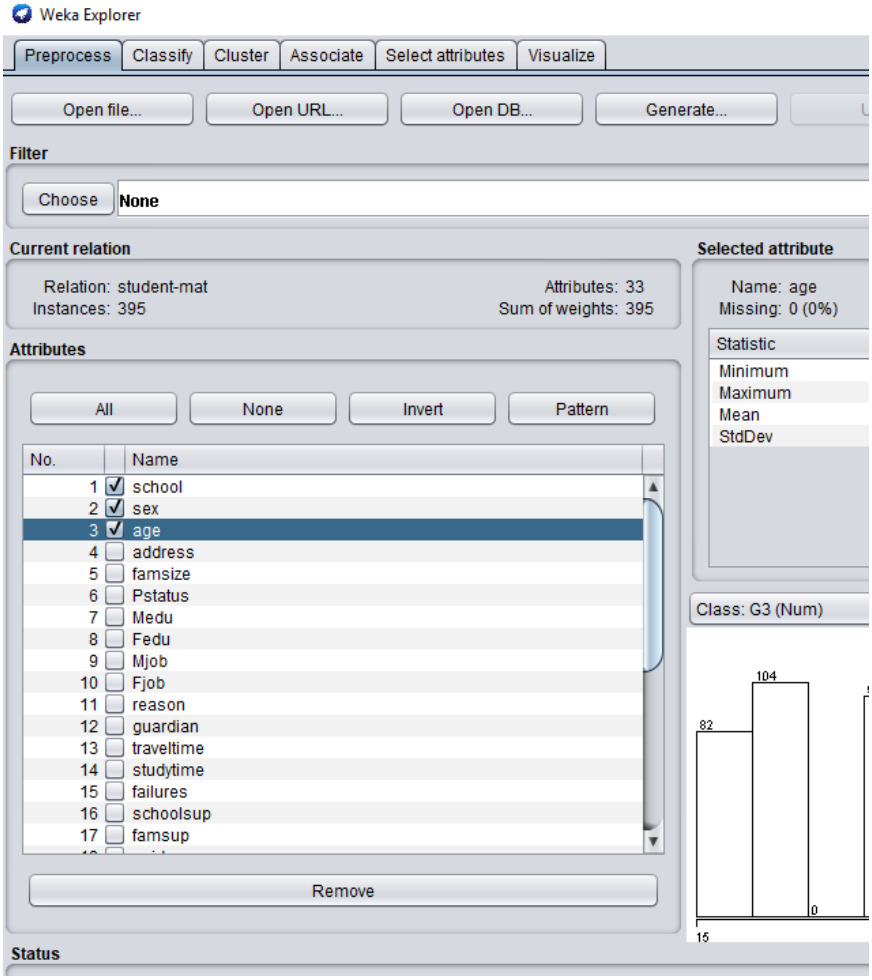
Correlation coefficient 0.2465

```
Mean absolute error                       4.0253
Root mean squared error                   5.5479
Relative absolute error                 116.9812 %
Root relative squared error             120.8057 %
Total Number of Instances                 395
```

On voit ici que la *RMSE* vaut 5.54 et que *RRSE* vaut 120% et que le coefficient de correlation est de 0.24. On en conclue que la qualité de la prédiction n'est pas terrible, mais qu'il existe tout de même une (petite) corrélation.

Q1 : Prenons KNN=10. Quelles sont les valeurs de RMSE et RRSE ? Est-ce que vous trouvez que cette prédiction est acceptable ?

Nous allons simplifier le problème en supprimant pratiquement tous les attributs. Nous allons chercher à prédire la note finale de l'élève à partir de ses notes des deux premiers semestres. Pour ce faire, nous retournons dans l'onglet *preprocess* et nous allons supprimer les champs qui ne nous intéressent pas. Attention : ces champs ne peuvent pas être récupérés à moins de recharger le fichier.



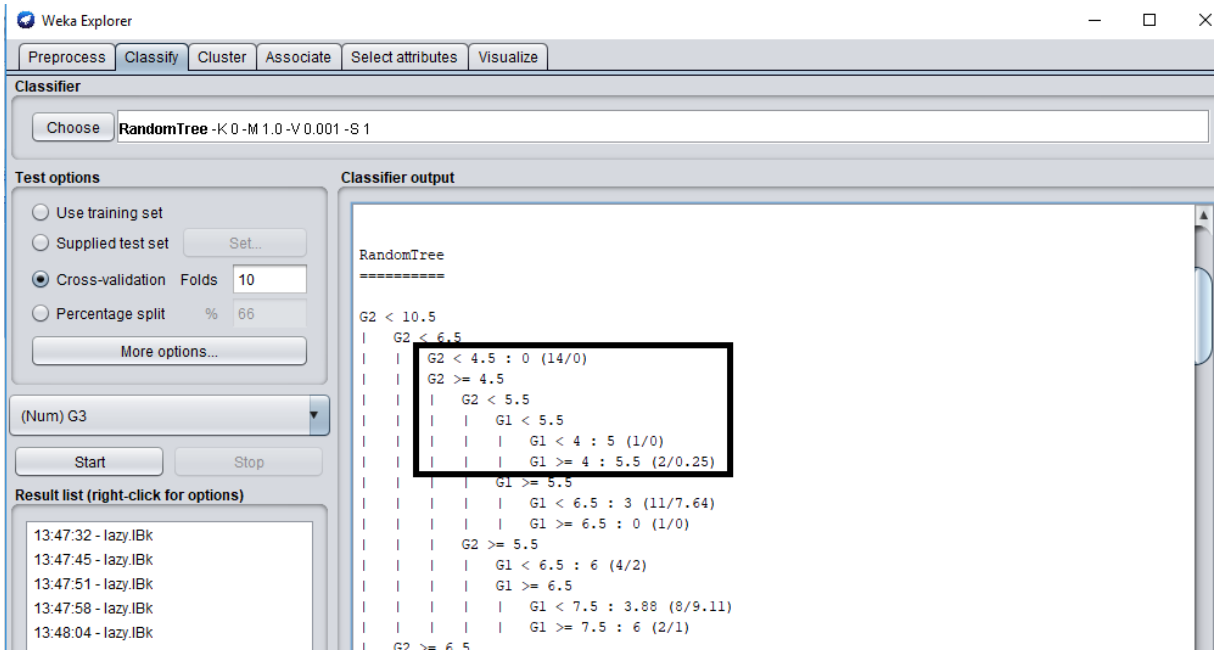
Cochez tous les attributs sauf G1, G2 et G3 (attributs 1 à 30), puis appuyez sur « Remove ». Il ne vous reste plus que 3 attributs.

Q2 : Appliquez de nouveau une classification KNN en prenant plusieurs valeurs de KNN (1, 5, 10, 15, 20, 25). Indiquez et commentez les résultats.

2) Classification RandomTree

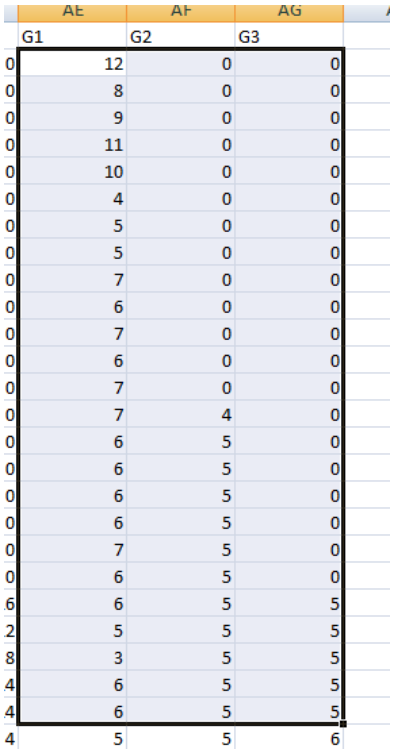
Le classifieur *Random Tree* permet également de faire des prédictions numériques en construisant un arbre de décision à partir d'un échantillonnage aléatoire du jeu de données. Nous restons ici avec une prédiction de G3 à partir de G1 et G2.

Lancez le classifieur *RandomTree*.



On voit qu'un arbre de décision a été construit. La 3e ligne signifie que : si G2 <10.5 et G2 <6.5 et G2 < 4.5 alors G3 = 0 , qu'il y a 14 instances dans ce cas, et qu'il y a une RMSE de 0, c'est-à-dire que c'est toujours vrai. Pareil si G2 est entre 4.5 et 5.5 et G1 <3 alors la note de G3 est égale à 5. Si G2 est entre 4.5 et 5.5 et G1 est supérieur à 4 alors la note est de 5.5 (avec une RMSE de 0.25)

On peut retrouver ces résultats si on consulte directement le fichier .csv



	AE	AF	AG
	G1	G2	G3
0	12	0	0
0	8	0	0
0	9	0	0
0	11	0	0
0	10	0	0
0	4	0	0
0	5	0	0
0	5	0	0
0	7	0	0
0	6	0	0
0	7	0	0
0	6	0	0
0	7	0	0
0	7	4	0
0	6	5	0
0	6	5	0
0	6	5	0
0	6	5	0
0	7	5	0
0	6	5	0
6	6	5	5
2	5	5	5
8	3	5	5
4	6	5	5
4	6	5	5
4	5	5	6

Pour s'adapter aux règle générées, on a trié le fichier selon l'attribut G2. On voit bien qu'en effet si G2 est inférieur ou égal à 4.5 alors G3 vaut 0 et qu'il y a bien 14 instances dans ce cas. Si G2 vaut 5 et G1 est inférieur à 4 il n'y a ici qu'un seul cas (G1=3) et G3 vaut bien 5. Si G2 vaut 5 et G1 vaut 5 on a deux lignes : l'une pour laquelle G3 vaut 5 et l'autre pour laquelle G3 vaut 6. Dans ce cas, la prédiction proposée est de 5.5, ce qui fait une RMSE de $(((0.5/5)^2+(0.5/6)^2)/2)^{0.5}$ soit environ 0.25

Voici les résultats généraux :

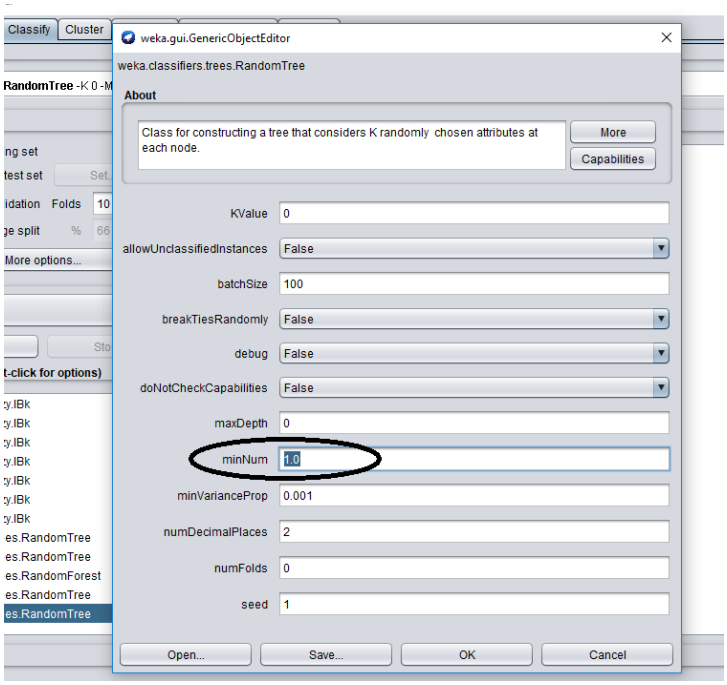
```
=== Cross-validation ===

=== Summary ===


Correlation coefficient                  0.888
Mean absolute error                      1.278
Root mean squared error                  2.1104
Relative absolute error                 37.1391 %
Root relative squared error             45.954  %
Total Number of Instances              395
```

La qualité de ce classifieur est bien meilleure (voir excellente !).

Notons que nous pouvons modifier le paramétrage de la taille minimale du poids des classes, qui par défaut vaut 1. Avec une valeur de 1, on risque d'avoir un problème de *surfitting* des résultats.



Q3 : Changeons la valeur minNums pour mettre la valeur 3. Relancez le classifieur et comparez la taille de l'arbre. Relancez avec la valeur minNum = 10 et comparez la qualité de la classification et la taille de l'arbre. Commentez.

3) Anonymisation de données en utilisant la *differential privacy* et le mécanisme Laplacien

Contrairement au k-anonymat qui modifie les quasi-identifiants (ici ce serait G1 et G2) pour les généraliser, à l'aide de la *differential privacy, n*ous allons effectuer une transformation du fichier .csv pour modifier la donnée sensible (ici G3), tout en ne touchant pas à G1 et G2. Si on estime que G1 et G2 sont également sensibles, on pourrait aussi les modifier à l'aide du mécanisme Laplacien.

On rappelle que pour avoir une protection de ε à l'aide du mécanisme Laplacien, il faut appliquer une modification de la valeur G3 avec une valeur tirée dans une distribution laplacienne paramétrée par ε et Δq (*sensibilité* de la requête).

Vous pouvez choisir à votre guise ε, ce qui permet de respecter le critère de la *differential privacy* (on considère ici que δ=0) :

$$\Pr[\Omega|D_1] \leq \exp(\varepsilon) \times \Pr[\Omega|D_2] + \delta$$

Pour ce qui est de la valeur de Δq, celle-ci dépend de l'utilisation qui est faite de la donnée. Par exemple dans le cas de KNN on va compter, pour chacun des K voisins d'un enregistrement donné, la valeur de G3 et affecter la valeur la plus probable, c'est cette affectation qui est le résultat de la fonction. Δq indique de combien peut varier au maximum le résultat si un individu est présent ou

pas dans la base de données. Prenons un exemple concret avec K=1 : imaginons que le voisin le plus proche ait la note réelle 0, et que la note générée est 20. La modification est donc de $\Delta q=20$.

On effectuera donc un tirage avec un ε donné (par exemple 0.1 ou 1) et une sensibilité de 20.

Exemple : Vous pouvez récupérer un programme exemple qui effectue un tirage ici :

https://benjamin-nguyen.fr/ENS/4ASTI-EA-BIGDATA-SECU/dp/laplace_ex.py

La fonction Laplacien prend 3 paramètres :

- epsilon ε : (une valeur en général inférieure à 1, plus epsilon est petit, meilleure est la protection, mais plus l'aléa généré est grand

- delta δ : une valeur petite mais supérieure à 0. On utilise ici $\delta=0$.

- sensitivity Δq : elle vaut 20 comme indiqué précédemment.

originalVal n'est pas un paramètre de la fonction, c'est une variable qui représente la donnée qu'on veut modifier, c'est-à-dire G3 (par exemple 10 ici)

size est un paramètre qui donne le nombre de tirages qu'on va faire (pour pouvoir observer la distribution).

Le code suivant prépare la distribution :

```
lbd = LaplaceBoundedDomain(epsilon=10, sensitivity=20, lower=0, upper=20)
```

On utilise un Laplacien « bounded » c'est-à-dire bornée sur le domaine [0 ;20] pour ne pas avoir des notes supérieures à 20 ou inférieures à 0 (on n'est pas dans un QCM) ☺

On peut ensuite appliquer ce bruit à la donnée avec l'instruction :

```
lbd.randomise(originalVal)
```

Le reste du code effectue `size` tirages et les affiche (on a utilisé une précision de 0.1 sur les notes), pour que vous puissiez voir la distribution des valeurs générées en fonction de ε.

Q4 : Affichez la distribution pour ε=0.01, ε=0.1, ε=1, ε=10 en utilisant size=1000. Interprétez cette distribution.

Si on choisit ε=10 quelle est la plus grande probabilité de tomber sur la bonne valeur tout en respectant le critère de DP (on rappelle qu'il y a 21 valeurs possibles)? Même question pour les autres ε.

A votre avis, quel(s) ε peut(ent) être utilisé(s) et pourquoi ?

Modification du fichier csv

Vous pouvez retrouver ici un exemple de script python manipulant le fichier csv qui nous intéresse (il faut bien sûr changer le chemin vers les fichiers csv dans les variables `pathin et pathout`) : https://benjamin-nguyen.fr/ENS/4ASTI-EA-BIGDATA-SECU/dp/csv_read.py

Ce script charge le fichier csv puis exporte uniquement les colonnes G1, G2 et G3 vous pouvez voir en chargeant ce nouveau fichier dans WEKA qu'on observe les mêmes résultats.

Q5 : Modifiez le script csv_read.py pour changer la valeur de G3 en appliquant une transformation laplacienne avec les paramètres ε=0.01, 0.1, 1 et 10.

Effectuez les analyses de classficiation avec KNN et RandomTree. Indiquez la qualité de votre prédiction.

Pouvez-vous proposer un meilleur ε que ceux que nous avons utilisés ? Concluez sur l'efficacité de votre protection et de votre prédiction.