

TD3 : Anonymisation et Analyse de Données

M2 IMIS et MIAGE

Temps prévu : 2h00

Dans ce TD, nous allons travailler avec les logiciels ARX (logiciel d'anonymisation) et WEKA (logiciel d'analyse de données). Nous nous intéresserons à la qualité des opérations de classification et de clustering.

Préparation du TD :

Téléchargez les logiciels ARX et WEKA. Ces logiciels sont réalisés en Java, il est donc souvent plus simple de télécharger juste le JAR exécutable, qui fonctionnera peu importe votre plateforme. Il existe également des versions installables.

ARX : <https://arx.deidentifier.org/downloads/>

WEKA : https://waikato.github.io/weka-wiki/downloading_weka/

Téléchargez également le fichier contenant les données à analyser : il s'agit d'un fichier sur des analyses de diabète des indiennes de la tribu Pima.

Données : <https://benjamin-nguyen.fr/ENS/4ASTI-EA-BIGDATA-SECU/pima-indians-diabetes.csv>

Ce fichier peut être chargé tel quel dans WEKA et ARX.

I- Analyse de données brutes

Nous débutons ce TD en analysant le fichier contenant les micro-données brutes.

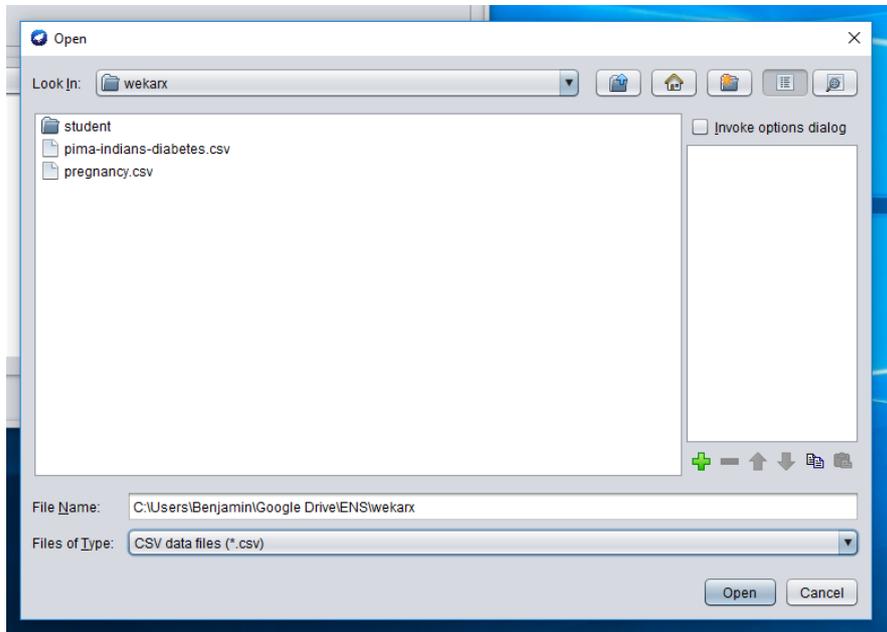
1) Lancement de WEKA GUI CHOOSER

Lancez WEKA et cliquez sur Explorer

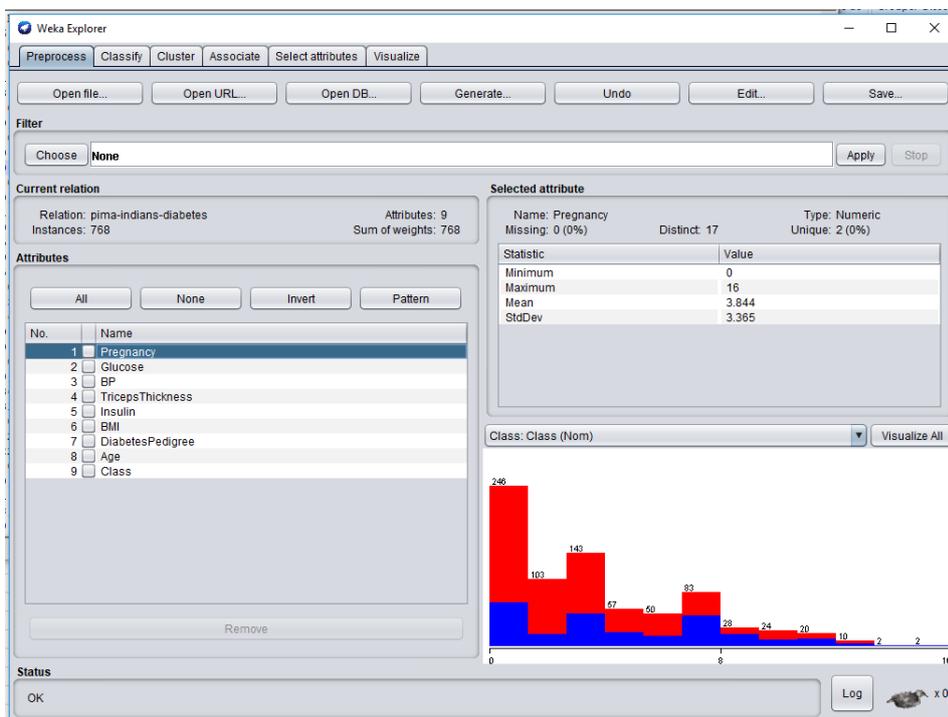


2) Exploration du jeu de données

Dans la fenêtre Explorer ouvrez le fichier que vous venez de télécharger. Pensez bien à choisir le type de fichier CSV.



Une fois chargé l'écran devrait ressembler à ceci :



On voit que ce jeu de données comporte 9 attribut (incluant l'attribut class que l'on souhaite prédire) et 768 instances (lignes ou individus). Les 8 attributs que nous allons analyser sont :

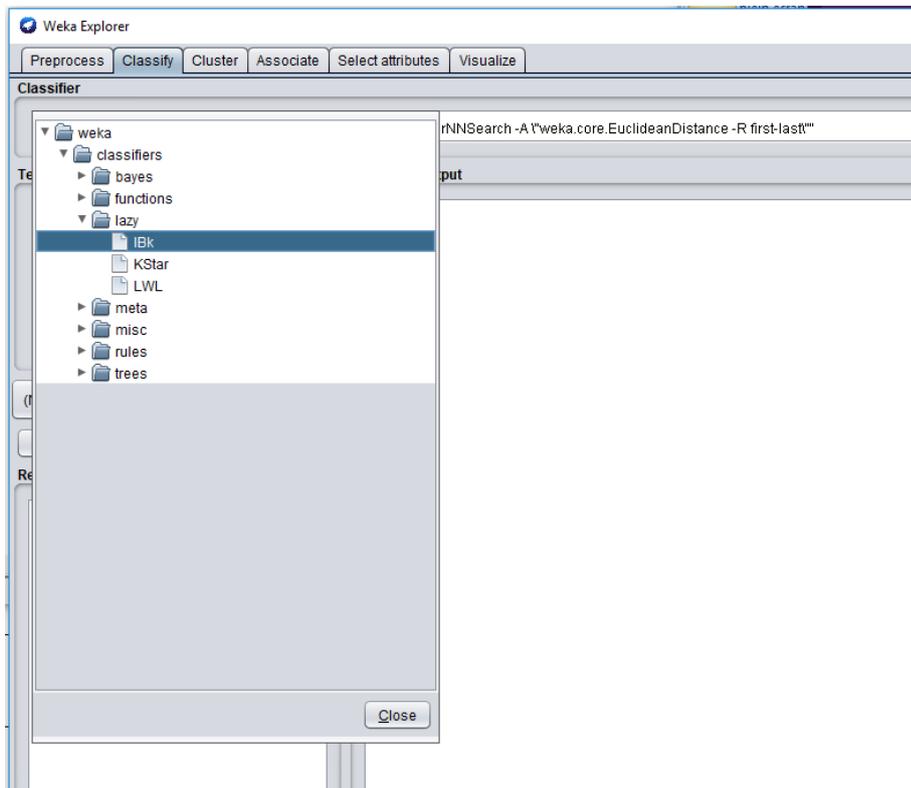
- le nombre de fois que la patiente a été enceinte (Pregnancy)
- son taux de glucose après ingestion au bout de 2h (Glucose)

- sa tension artérielle (BP en mm Hg)
- l'épaisseur de la peau de son triceps (TricepsThickness en mm)
- la prise d'insuline au bout de 2h (Insulin en mu U/ml)
- l'indice de masse corporelle (BMI en $(\text{kg}/\text{m})^2$)
- la fonction pedigree de diabete (DiabetesPedigree)
- son age en années (AGE)

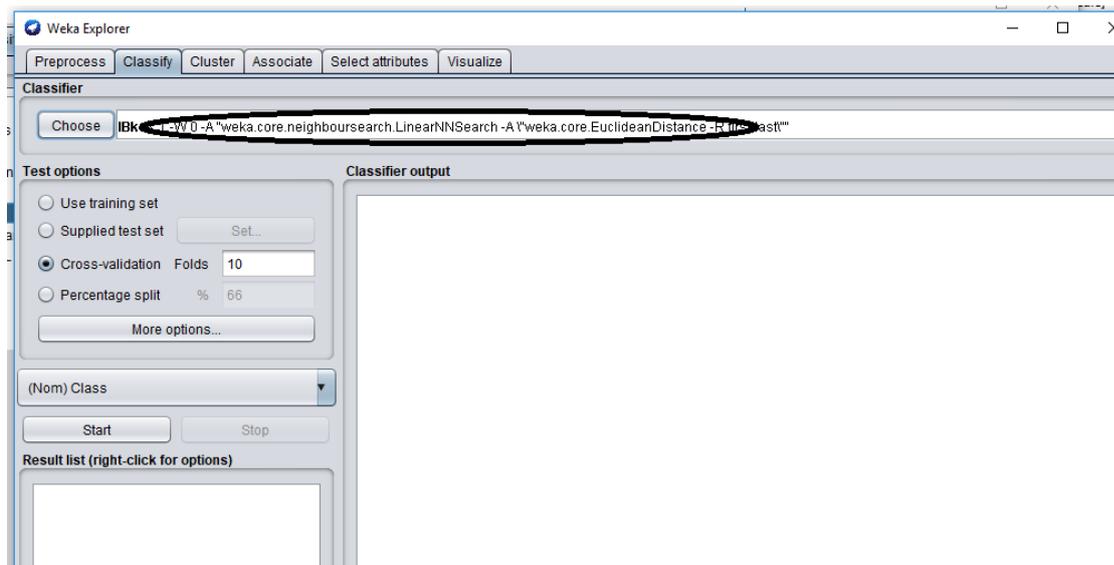
Et ce, afin de prédire le diagnostic de diabete (Class, YES ou NO).

3) Classification avec kNN

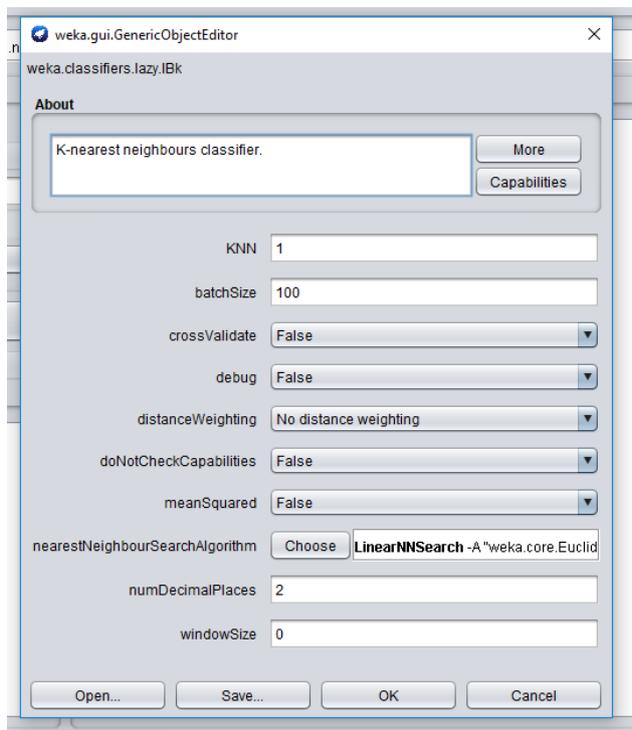
Appuyez sur l'onglet « Classify », puis le bouton « choose ». Une fenêtre s'ouvre pour choisir le classifieur que vous voulez utiliser. Choisissez le répertoire *lazy* puis le classifieur *IBk*



Cliquez ensuite sur le classifieur comme indiqué sur la figure, pour ouvrir la fenêtre de configuration du classifieur.



Vous pouvez maintenant configurer le classifieur. Plusieurs paramètres peuvent nous intéresser ici : le paramètre KNN (nombre de voisins, nous allons le faire varier de 1 à 20). Le champ `distanceWeighting` permet de pondérer les voisins selon leur distance (dit autrement, on peut donner ou pas plus d'importance aux voisins selon leur proximité). D'autres éléments comme `nearestNeighbourSearchAlgorithm` permettent de configurer le type d'algorithme de recherche utilisé pour optimiser le temps d'exécution.



Choisissez `KNN=1`, `distanceWeighting=no distance weighting` et cliquez sur OK.

Choisissez dans Test options la *cross-validation* folds = 10 et vérifiez que c'est bien (Nom) Class que vous cherchez à prédire, puis appuyez sur le bouton Start. Vous observez les résultats suivants dans la fenêtre de droite :

=== Run information ===

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

Relation: pima-indians-diabetes

Instances: 768

Attributes: 9

Pregnancy

Glucose

BP

TricepsThickness

Insulin

BMI

DiabetesPedigree

Age

Class

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

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances

536

69.7917 %

Incorrectly Classified Instances	232	30.2083 %
Kappa statistic	0.3211	
Mean absolute error	0.3027	
Root mean squared error	0.5488	
Relative absolute error	66.5897 %	
Root relative squared error	115.1446 %	
Total Number of Instances	768	

=== Detailed Accuracy By Class ===

Area	PRC Area	TP Rate Class	FP Rate	Precision	Recall	F-Measure	MCC	ROC
0,646	0,465	0,522 YES	0,208	0,574	0,522	0,547		0,322
0,646	0,729	0,792 NO	0,478	0,756	0,792	0,773		0,322
Weighted Avg. 0,646	0,637	0,698	0,384	0,692	0,698	0,694		0,322

=== Confusion Matrix ===

```

a   b   <-- classified as
140 128 |   a = YES
104 396 |   b = NO

```

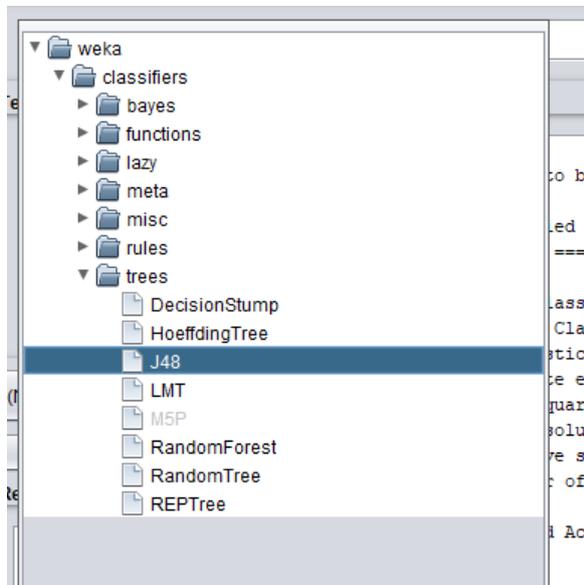
La ligne Weighted Avg. peut être utilisée pour mesurer la qualité de ce classifieur. Nous allons considérer la métrique F-Measure (qui vaut ici 0.694) comme indication de la qualité de la classification. Plus cette valeur est proche de 1, meilleure est la prédication.

Q1- QUESTION : Faites varier les paramètres KNN, nearestNeighbourSearchAlgorithm et distanceWeighting pour essayer d'améliorer la qualité de votre classifieur, que vous estimerez par la valeur F-MEASURE comme indiquée plus haut. Tracez par exemple la fonction représentant la F-Measure, en fonction de K. Pour quelle valeur observez-vous la meilleure qualité de classification ? Indiquez cette qualité.

Dans la suite, nous prendrons KNN = 7 et distanceWeighting = 1-distance. Notez la valeur de la F-Measure, qui servira de référence.

4) Classification avec J48

Choisissez le classifieur par arbre trees/J48



Le paramètre qui va nous intéresser ici est minNumObj qui indique le nombre minimum d'instances qu'on souhaite avoir dans une feuille de décision, ainsi que confidenceFactor (le facteur de confiance) de chaque classe.

Voici ce qui est affiché pour la valeur minNumObj=2 et confidenceFactor=0.25

```
=== Run information ===
```

```
Scheme:          weka.classifiers.trees.J48 -C 0.25 -M 2
```

```
Relation:        pima-indians-diabetes
```

```
Instances:       768
```

```
Attributes:      9
```

```
Pregnancy
```

```
Glucose
```

```
BP
```

```
TricepsThickness
```

```
Insulin
```

```
BMI
```

```
DiabetesPedigree
```

Age

Class

Test mode: 10-fold cross-validation

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

J48 pruned tree

Glucose <= 127

```
| BMI <= 26.4: NO (132.0/3.0)
| BMI > 26.4
| | Age <= 28: NO (180.0/22.0)
| | Age > 28
| | | Glucose <= 99: NO (55.0/10.0)
| | | Glucose > 99
| | | | DiabetesPedigree <= 0.56: NO (84.0/34.0)
| | | | DiabetesPedigree > 0.56
| | | | | Pregnancy <= 6
| | | | | | Age <= 30: YES (4.0)
| | | | | | Age > 30
| | | | | | | Age <= 34: NO (7.0/1.0)
| | | | | | | Age > 34
| | | | | | | | BMI <= 33.1: YES (6.0)
| | | | | | | | BMI > 33.1: NO (4.0/1.0)
| | | | | | | | | Pregnancy > 6: YES (13.0)
```

Glucose > 127

```
| BMI <= 29.9
| | Glucose <= 145: NO (41.0/6.0)
| | Glucose > 145
```

```

|   |   |   Age <= 25: NO (4.0)
|   |   |   Age > 25
|   |   |   |   Age <= 61
|   |   |   |   |   BMI <= 27.1: YES (12.0/1.0)
|   |   |   |   |   BMI > 27.1
|   |   |   |   |   |   BP <= 82
|   |   |   |   |   |   |   DiabetesPedigree <= 0.396: YES (8.0/1.0)
|   |   |   |   |   |   |   DiabetesPedigree > 0.396: NO (3.0)
|   |   |   |   |   |   |   BP > 82: NO (4.0)
|   |   |   |   |   |   |   Age > 61: NO (4.0)
|   |   |   |   |   |   |   BMI > 29.9
|   |   |   |   |   |   |   Glucose <= 157
|   |   |   |   |   |   |   |   BP <= 61: YES (15.0/1.0)
|   |   |   |   |   |   |   |   BP > 61
|   |   |   |   |   |   |   |   Age <= 30: NO (40.0/13.0)
|   |   |   |   |   |   |   |   Age > 30: YES (60.0/17.0)
|   |   |   |   |   |   |   |   Glucose > 157: YES (92.0/12.0)

```

Number of Leaves : 20

Size of the tree : 39

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	567	73.8281 %
Incorrectly Classified Instances	201	26.1719 %

```

Kappa statistic          0.4164
Mean absolute error     0.3158
Root mean squared error 0.4463
Relative absolute error 69.4841 %
Root relative squared error 93.6293 %
Total Number of Instances 768

```

=== Detailed Accuracy By Class ===

Area	PRC Area	TP Rate Class	FP Rate	Precision	Recall	F-Measure	MCC	ROC
0,751	0,572	0,597 YES	0,186	0,632	0,597	0,614		0,417
0,751	0,811	0,814 NO	0,403	0,790	0,814	0,802		0,417
Weighted Avg. 0,751	0,727	0,738	0,327	0,735	0,738	0,736		0,417

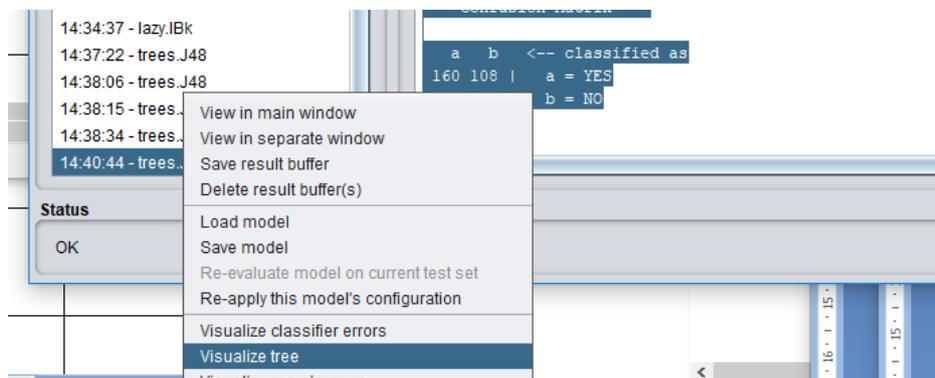
=== Confusion Matrix ===

```

a  b  <-- classified as
160 108 |  a = YES
93  407 |  b = NO

```

Regardez bien le champ J48 Pruned Tree qui représente l'arbre de décision. Vous pouvez également visualiser l'arbre via un clic droit sur votre exécution (dans Result List -> visualize tree)

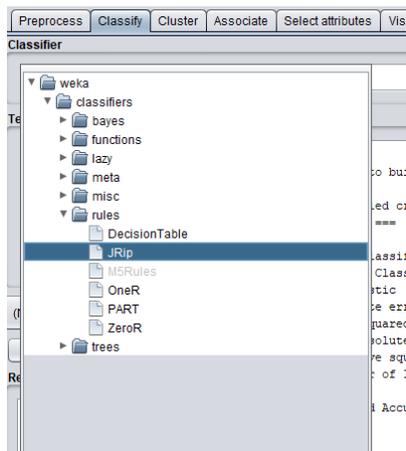


Q2- QUESTION : Faites varier les valeurs minNumObj et confidenceFactor. Notez l'évolution de la qualité de la F-Measure en la traçant. Notez la valeur pour laquelle vous obtenez la meilleure qualité, ainsi que sa valeur.

Dans la suite nous prendrons minNumObj=20 et confidenceFactor=0.75

5) Classification par règles : RIPPER

Nous allons utiliser un 3^e type de classifieur : un classifieur à base de règles logique. Choisissez le classifieur Rules/JRIP



Nous allons simplement utiliser JRIP avec les paramètres de base. Nous obtenons le résultat suivant :

```
=== Run information ===
```

```
Scheme:          weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1
```

```
Relation:        pima-indians-diabetes
```

```
Instances:       768
```

```
Attributes:     9
```

```
Pregnancy
```

```
Glucose
```

```
BP
```

```
TricepsThickness
```

```
Insulin
```

```
BMI
```

```
DiabetesPedigree
```

```
Age
```

```
Class
```

Test mode: 10-fold cross-validation

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

JRIP rules:

=====

(Glucose >= 167) => Class=YES (79.0/11.0)

(Glucose >= 112) and (DiabetesPedigree >= 0.529) and (BMI >= 30) and (Age >= 30) => Class=YES (48.0/9.0)

(Glucose >= 115) and (BMI >= 42.1) => Class=YES (34.0/12.0)

(Age >= 31) and (Glucose >= 108) and (BMI >= 28.2) => Class=YES (88.0/38.0)

=> Class=NO (519.0/89.0)

Number of Rules : 5

Time taken to build model: 0.14 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	572	74.4792 %
Incorrectly Classified Instances	196	25.5208 %
Kappa statistic	0.4171	
Mean absolute error	0.3461	
Root mean squared error	0.4351	
Relative absolute error	76.1388 %	
Root relative squared error	91.2746 %	
Total Number of Instances	768	

=== Detailed Accuracy By Class ===

Area	PRC Area	TP Rate Class	FP Rate	Precision	Recall	F-Measure	MCC	ROC
0,708	0,576	0,556 YES	0,154	0,659	0,556	0,603		0,420
0,708	0,782	0,846 NO	0,444	0,780	0,846	0,812		0,420
Weighted Avg. 0,708	0,710	0,745	0,343	0,738	0,745	0,739		0,420

=== Confusion Matrix ===

```
a  b  <-- classified as
149 119 |  a = YES
77  423 |  b = NO
```

On s'intéresse plus précisément aux règles générées par JRIP qui sont les suivantes :

JRIP rules:

=====

(Glucose >= 167) => Class=YES (79.0/11.0)

(Glucose >= 112) and (DiabetesPedigree >= 0.529) and (BMI >= 30) and (Age >= 30) => Class=YES (48.0/9.0)

(Glucose >= 115) and (BMI >= 42.1) => Class=YES (34.0/12.0)

(Age >= 31) and (Glucose >= 108) and (BMI >= 28.2) => Class=YES (88.0/38.0)

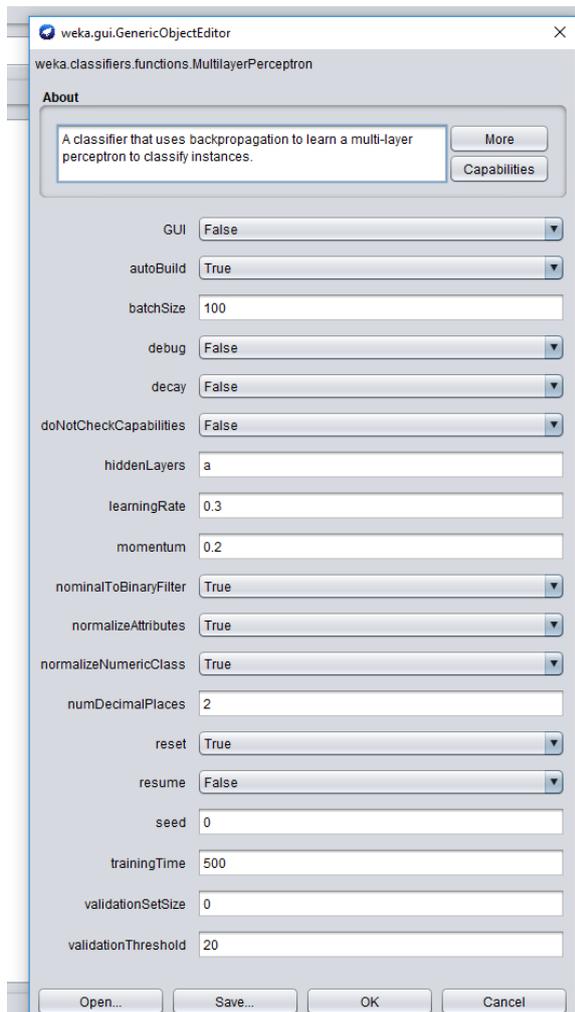
=> Class=NO (519.0/89.0)

Number of Rules : 5

On voit qu'il serait également possible de construire un arbre de décision à partir de ces règles.

6) Classification avec des réseaux de neurones profonds

Nous utilisons un dernier type de classifieur : des réseaux de neurones profonds, disponible dans fonctions/multilayer Perceptron.



De nombreux paramètres sont disponibles, comme le nombre de nœuds dans les niveaux cachés (hiddenLayers, qui vaut « a » ou une valeur entière), le taux d'apprentissage (learningRate) ou encore l'inertie (momentum). Le temps d'apprentissage peut également être contrôlé (trainingTime). Lançons le modèle avec les paramètres par défaut.

=== Run information ===

```
Schema:          weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2  
-N 500 -V 0 -S 0 -E 20 -H a
```

```
Relation:        pima-indians-diabetes
```

```
Instances:       768
```

```
Attributes:      9
```

```
                Pregnancy
```

Glucose
BP
TricepsThickness
Insulin
BMI
DiabetesPedigree
Age
Class

Test mode: 10-fold cross-validation

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

Sigmoid Node 0

Inputs	Weights
Threshold	3.628703319715117
Node 2	-6.313299284416023
Node 3	-2.568104564109483
Node 4	-2.8448541489871015
Node 5	-2.531131735140029
Node 6	-2.2169477046738026

Sigmoid Node 1

Inputs	Weights
Threshold	-3.6287033197148237
Node 2	6.313299284413041
Node 3	2.5681045641093068
Node 4	2.844854148987099
Node 5	2.531131735139932
Node 6	2.216947704673593

Sigmoid Node 2

Inputs	Weights
--------	---------

Threshold -5.726636007684776
Attrib Pregnancy 0.47208258366115546
Attrib Glucose -6.5452774656896135
Attrib BP 1.9803888455019736
Attrib TricepsThickness -4.1742177349813785
Attrib Insulin 7.048470366054944
Attrib BMI 0.48531328921316397
Attrib DiabetesPedigree 1.9877172681824125
Attrib Age -7.597191482777374

Sigmoid Node 3

Inputs Weights
Threshold -2.7031829318357508
Attrib Pregnancy 3.0657225750732575
Attrib Glucose -5.4957416963918035
Attrib BP -9.569096675837606
Attrib TricepsThickness 0.06271383682773789
Attrib Insulin -1.4401488539528373
Attrib BMI -9.10971393376813
Attrib DiabetesPedigree -3.1749225222292794
Attrib Age 12.985230736208651

Sigmoid Node 4

Inputs Weights
Threshold -2.202935559669145
Attrib Pregnancy -1.4493533000047487
Attrib Glucose -12.089311850014818
Attrib BP -1.3398675531560917
Attrib TricepsThickness 2.967829312368748
Attrib Insulin 1.1802832608623406
Attrib BMI -6.286813870465455
Attrib DiabetesPedigree -6.776007065799867

Attrib Age 2.8238051786015523

Sigmoid Node 5

Inputs Weights

Threshold -6.414462475572164

Attrib Pregnancy 7.527527969290638

Attrib Glucose -11.389428921109165

Attrib BP 6.345771244438931

Attrib TricepsThickness 0.22924527140725157

Attrib Insulin -1.341715914555352

Attrib BMI -5.840278388013632

Attrib DiabetesPedigree -1.841522948252757

Attrib Age -16.80717183085853

Sigmoid Node 6

Inputs Weights

Threshold 1.5331683098797988

Attrib Pregnancy -10.455311936699879

Attrib Glucose -4.7970387233467635

Attrib BP 3.3118562514122276

Attrib TricepsThickness 1.2804738190655798

Attrib Insulin -0.7339822735340477

Attrib BMI -10.25191137035137

Attrib DiabetesPedigree -2.88526462850306

Attrib Age 8.695516148522088

Class YES

Input

Node 0

Class NO

Input

Node 1

Time taken to build model: 0.73 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	578	75.2604 %
Incorrectly Classified Instances	190	24.7396 %
Kappa statistic	0.4508	
Mean absolute error	0.2942	
Root mean squared error	0.4224	
Relative absolute error	64.7386 %	
Root relative squared error	88.6256 %	
Total Number of Instances	768	

=== Detailed Accuracy By Class ===

Area	PRC Area	TP Rate Class	FP Rate	Precision	Recall	F-Measure	MCC	ROC
0,794	0,658	0,627 YES	0,180	0,651	0,627	0,639		0,451
0,794	0,859	0,820 NO	0,373	0,804	0,820	0,812		0,451
Weighted Avg. 0,794	0,789	0,753	0,306	0,751	0,753	0,751		0,451

=== Confusion Matrix ===

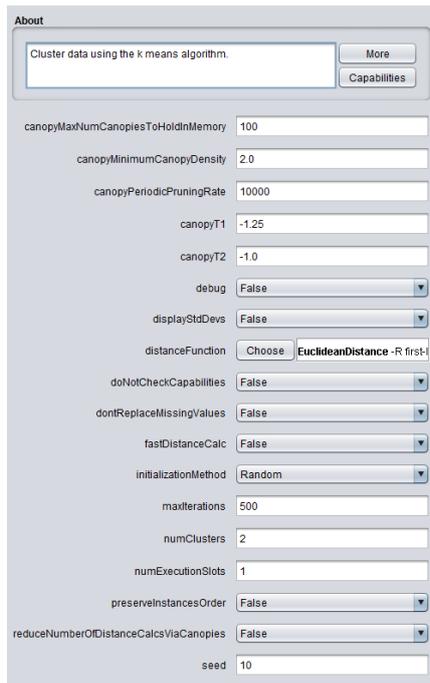
```
a  b  <-- classified as
168 100 |  a = YES
90  410 |  b = NO
```

Les informations de sortie sont récupérées dans les nœuds 0 et 1. La F-Measure vaut ici 0.751.

Q3 : QUESTION : Jouez un peu avec le paramétrage du réseau de neurones. Indiquez les valeurs de paramètres que vous utilisez pour obtenir une meilleure F-Measure.

7) Clustering avec K-Means

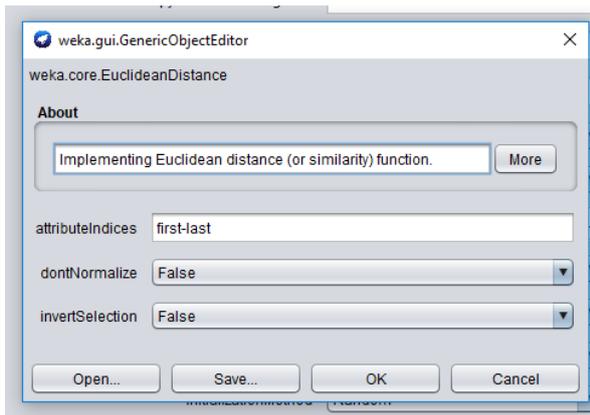
Nous allons maintenant effectuer des tâches de clustering. Choisissons l'onglet cluster, puis l'algorithme *SimpleKMeans*.



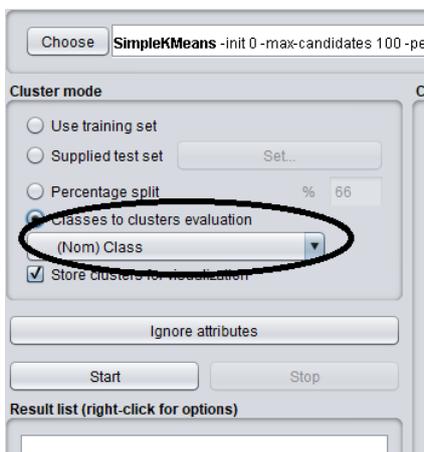
The screenshot shows the configuration window for the SimpleKMeans algorithm. The parameters are as follows:

Parameter	Value
canopyMaximumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first
doNotCheckCapabilities	False
doNotReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	2
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalculationsViaCanopies	False
seed	10

Deux paramètres sont pertinents ici : *distanceFunction* qui permet de choisir le type de distance (Euclidienne, Manhattan, etc.) et *numClusters* qu'on va fixer à 2 (car il y a 2 clusters YES et NO). Lorsqu'on utilise beaucoup d'attributs catégoriels, il est pertinent d'utiliser la distance Manhattan. Lorsqu'on a des données numériques, on peut préférer utiliser la distance euclidienne. Par ailleurs, lorsqu'on clique sur la métrique qu'on utilise, on peut choisir de ne l'appliquer que sur un sous-ensemble des dimensions. On peut également décider de normaliser les dimensions ou pas (si on met *doNotNormalize* = false c'est en fait qu'on normalise, c'est-à-dire qu'on considère que toutes les dimensions ont le même poids. Si on met *doNotNormalize* = true les dimensions ayant des valeurs très grandes seront privilégiées. Ici, comme on souhaite donner un poids équivalent à chaque dimension, on choisit *doNotNormalize* = false.



Afin d'avoir un retour sur la qualité de notre clustering, il faut régler le mode de clustering sur « Classes to clusters evaluation » et choisir l'attribut qui correspond aux classes, à savoir *Class*. Le bouton « *Ignore Attributes* » permet d'ignorer certaines dimensions. Dans un premier temps, on décide de toutes les conserver.



L'exécution de Simple K-Means donne le résultat suivant :

=== Run information ===

```
Scheme:          weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -
periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A
"weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
```

Relation: pima-indians-diabetes

Instances: 768

Attributes: 9

Pregnancy

Glucose

BP

TricepsThickness

Insulin
BMI
DiabetesPedigree
Age

Ignored:

Class

Test mode: Classes to clusters evaluation on training data

=== Clustering model (full training set) ===

kMeans

=====

Number of iterations: 5

Within cluster sum of squared errors: 122.87437030193571

Initial starting points (random):

Cluster 0: 1,126,56,29,152,28.7,0.801,21

Cluster 1: 8,95,72,0,0,36.8,0.485,57

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Full Data	Cluster#	
		0	1
	(768.0)	(513.0)	(255.0)
=====			
Pregnancy	3.8438	2.0468	7.4588

Glucose	120.8945	115.4678	131.8118
BP	69.1055	65.9708	75.4118
TricepsThickness	20.5365	21.8285	17.9373
Insulin	79.7995	84.9025	69.5333
BMI	31.9926	31.7854	32.4094
DiabetesPedigree	0.4719	0.4709	0.4738
Age	33.2409	26.8168	46.1647

Time taken to build model (full training data) : 0.03 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 513 (67%)

1 255 (33%)

Class attribute: Class

Classes to Clusters:

0 1 <-- assigned to cluster

134 134 | YES

379 121 | NO

Cluster 0 <-- NO

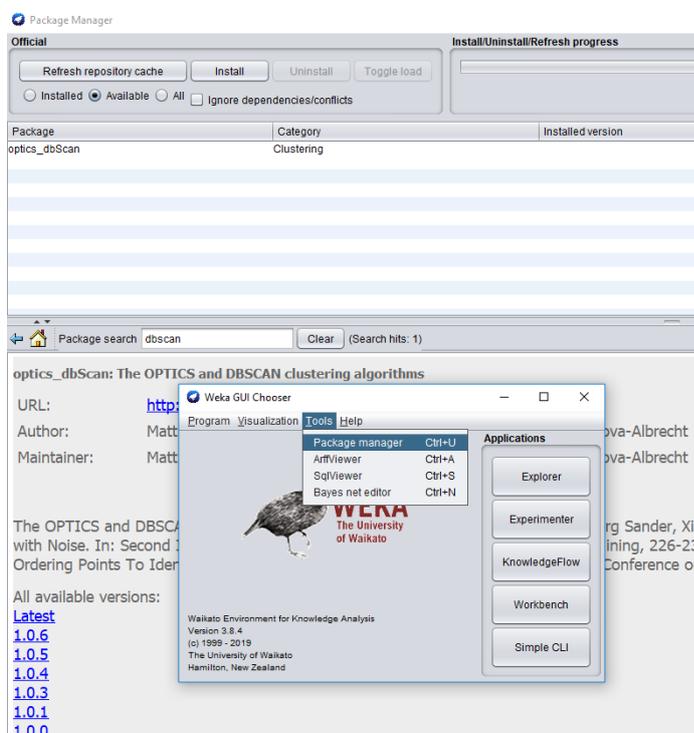
Cluster 1 <-- YES

Incorrectly clustered instances : 255.0 33.2031 %

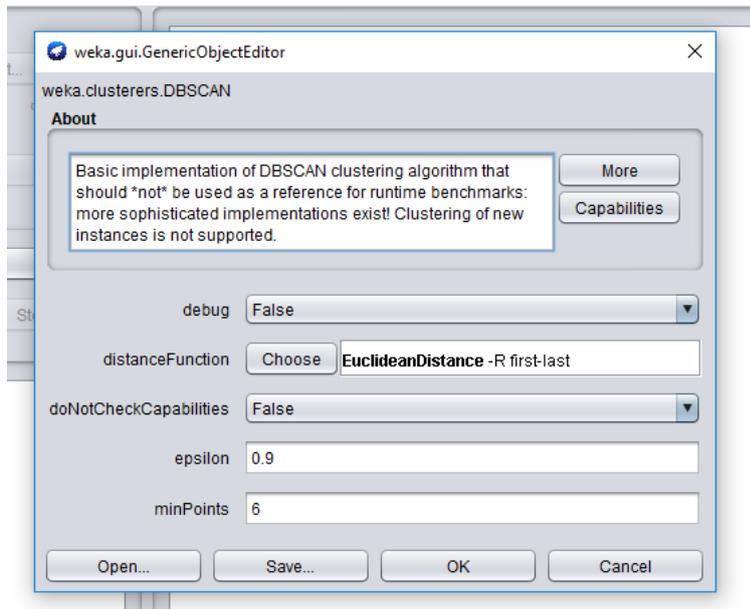
La qualité de la classification est indiquée sur la dernière ligne : 33% des instances sont mal classifiées (et donc 66% donc bien classifiées). Le tableau « Classes to Clusters » permet de comprendre un peu mieux la qualité de la classification : ici le cluster 0 correspond aux individus avec la valeur « NO », il y a 379 corrects (vrai positifs) et 134 erreurs (faux positifs). Pour le cluster 1 (« YES ») il y a 134 corrects (vrai positifs) et 121 erreurs (faux positifs).

8) Clustering par densité

Nous allons utiliser un algorithme de clustering par densité, DBSCAN. Il faut commencer par l'installer, car il n'est pas chargé par défaut. Pour ce faire, fermez la fenêtre WEKA et retournez dans le GUI. Choisissez de WEKA et allez dans le menu Tools -> Package Manager ce qui ouvre une nouvelle fenêtre.



Tapez « dbscan » dans la ligne « *package search* » puis enter et installez le package « *optics_dbscan* ». Lorsqu'on recharge notre fichier csv et qu'on se rend dans l'onglet « clustering » on dispose maintenant de l'algorithme DBSCAN. Les paramètres à considérer comme vu dans le cours sont minPoints (nombre de voisins) et epsilon (la distance). On peut également changer de fonction distance comme pour KMeans si on le souhaite.



Lançons l'algorithme avec le paramétrage de base. On obtient le résultat suivant :

Time taken to build model (full training data) : 0.18 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 768 (100%)

Class attribute: Class

Classes to Clusters:

0 <-- assigned to cluster

268 | YES

500 | NO

Cluster 0 <-- NO

Incorrectly clustered instances : 268.0 34.8958 %

L'interprétation qu'on peut donner ici, c'est que toutes les instances ont été mises dans le même cluster ! Le résultat n'est donc pas du tout satisfaisant. Pour augmenter le nombre de clusters on peut agir de deux manières : augmenter le nombre de voisins et/ou réduire la distance.

Q4 : QUESTION : La configuration de DBSCAN est relativement délicate. Essayez de modifier ces paramètres pour obtenir un clustering de relativement bonne qualité (au moins 2 clusters, au moins 100 instances clusterées dont moins de 30% d'instances mal clusterées). Indiquez vos paramétrages et commentez la qualité de vos résultats.

Utilisez le paramétrage suivant : epsilon = 0.15 et minPoints = 4. Interprétez les résultats.

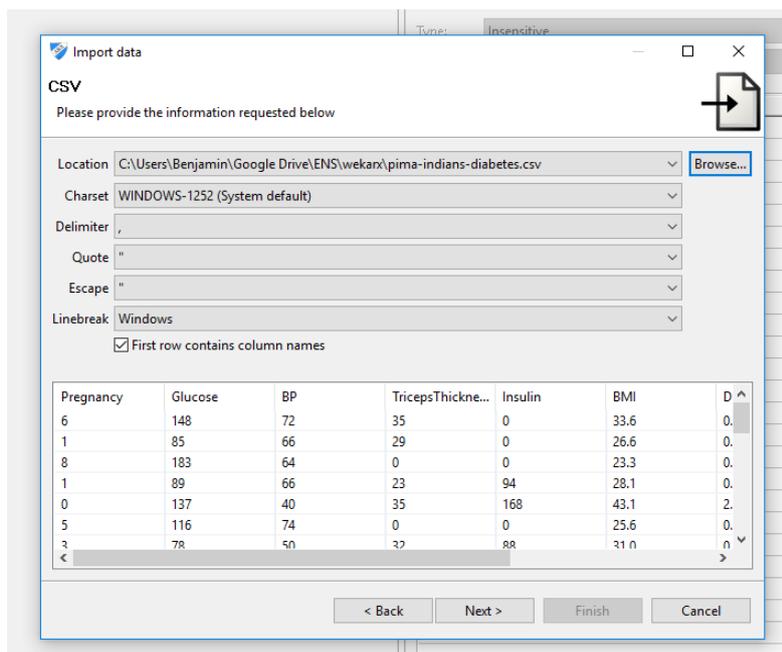
Voilà, vous avez pu expérimenter plusieurs techniques d'analyse de données classiques. Nous pouvons constater que leur paramétrage n'est pas si simple, et que la qualité obtenue est très variable. Les techniques d'analyse supervisées (classification) produisent de manière assez naturelle des résultats de meilleure qualité que les techniques non supervisées (clustering).

II- Anonymisation avec ARX

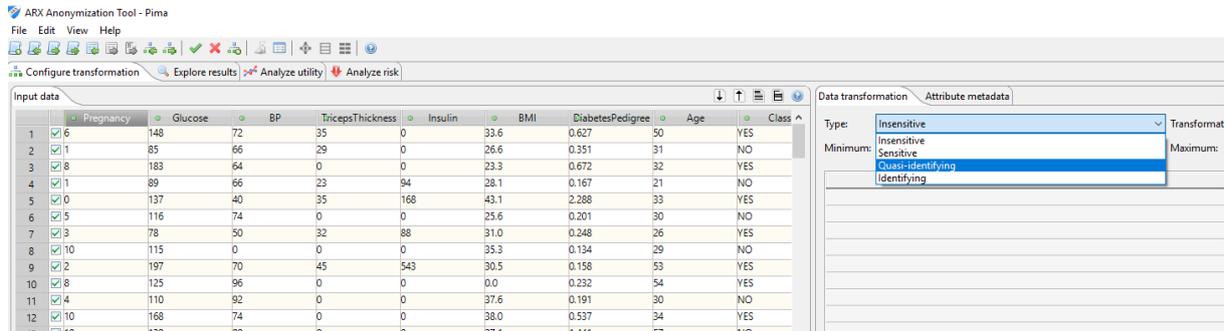
Nous allons maintenant utiliser ARX pour anonymiser les données en utilisant le modèle du k -anonymat. Puis nous allons exécuter des analyses de données comme dans la partie I et comparer les résultats.

9) Mise en place

Lancez ARX et créez un nouveau projet (appelez le « Pima »), puis choisissez le menu File-> import data et chargez le .csv En principe les délimiteurs par défaut sont correctement configurés. Validez les paramètres de typage auto-déTECTÉS.

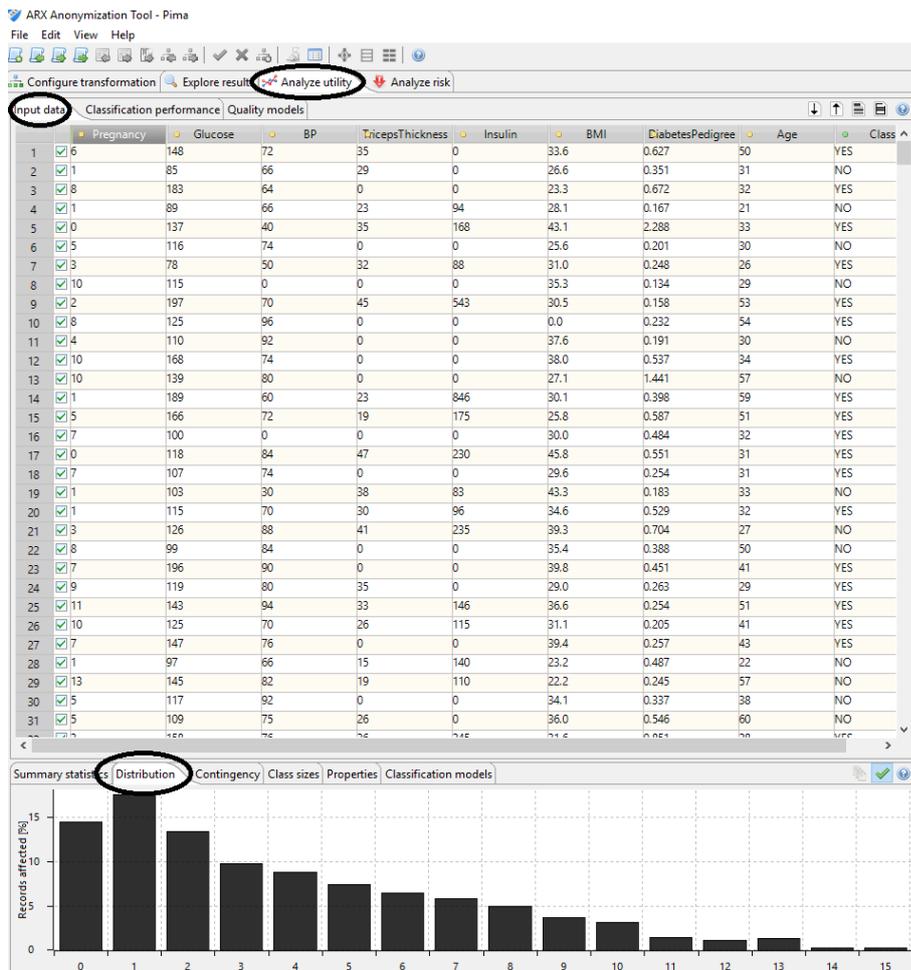


Nous devons maintenant créer 8 hiérarchies pour les données. Commençons avec l'attribut « pregnancy ». Par défaut tous les attributs sont classés comme *insensitive*, c'est-à-dire ignorés par ARX. Passons *pregnancy* en quasi-identificateur. La lumière devant lui devient jaune pour indiquer ce changement de statut.



Faisons de même pour tous les attributs sauf *class*.

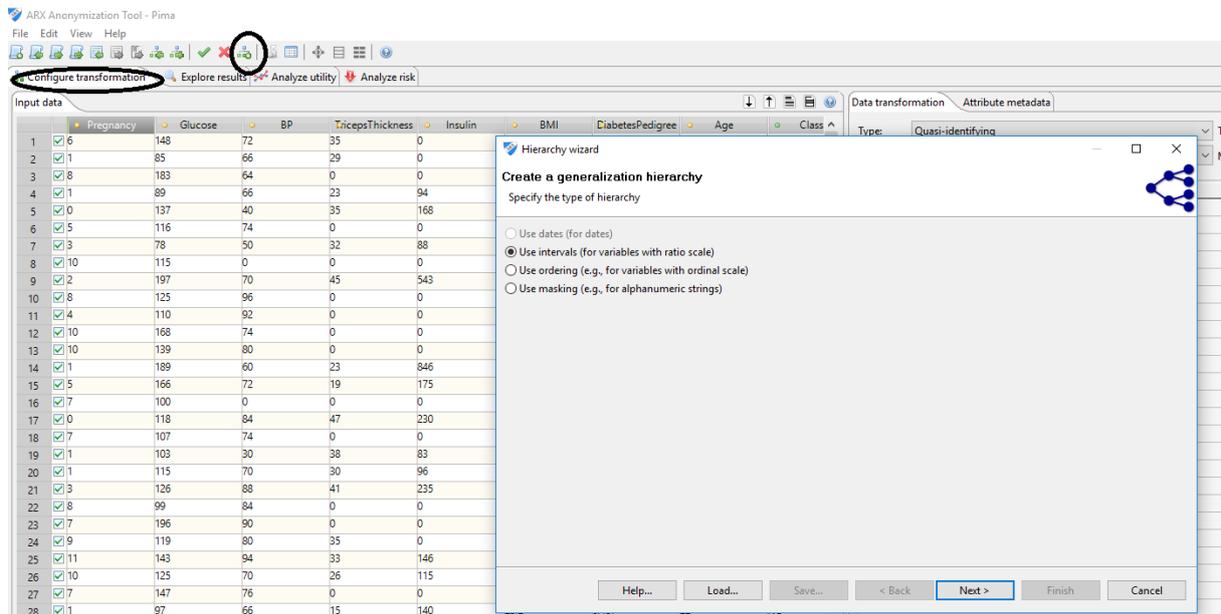
En cliquant sur l'onglet Analyse Utility et Distribution, on peut voir les valeurs que prennent chaque attribut.



On voit par exemple que l'attribut *pregnancy* varie de 0 à 15 avec une faible proportion des individus ayant une valeur > 10.

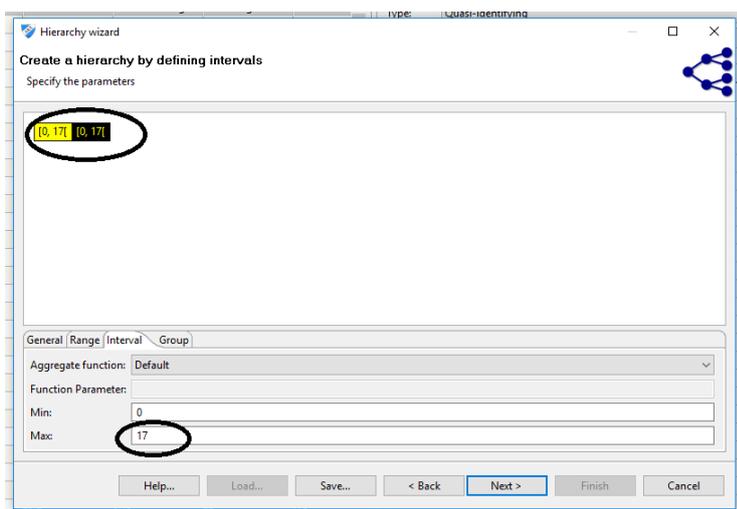
10) Construction de Hiérarchies

Construisons maintenant la hiérarchie de généralisation en retournant sur l'onglet *configure transformation* et en appuyant sur le bouton *create hierarchy* et utilisons des intervalles (premier choix proposé).

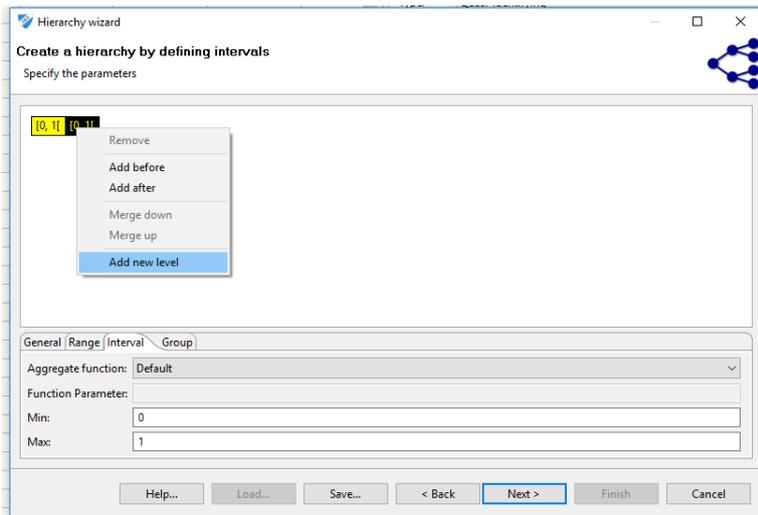


Les intervalles sont disponibles dès lors qu'on traite des données numériques. Si on traite des données catégorielles, il faut utiliser « ordering » qui permet d'ordonner ses données comme on le souhaite, puis construire des regroupements à plusieurs niveaux.

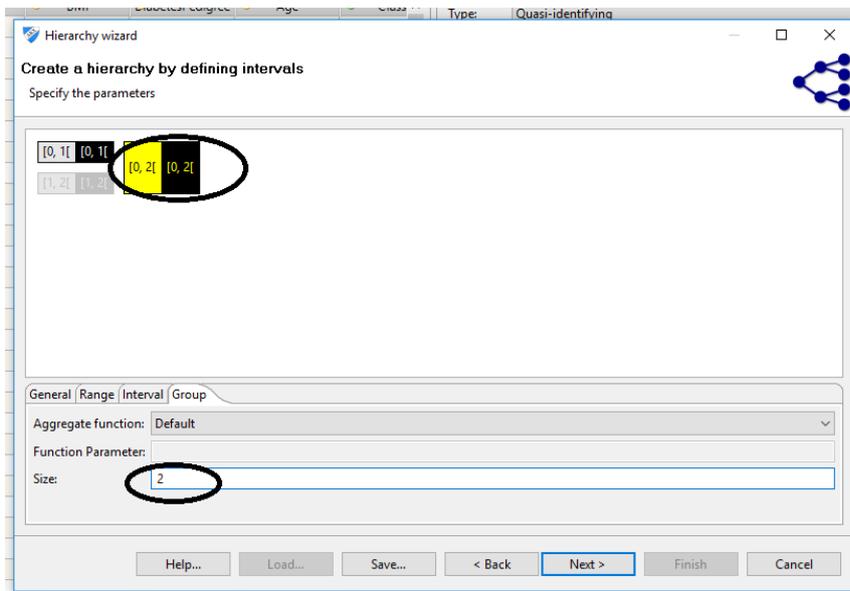
Commençons par une construction automatique de la hiérarchie.



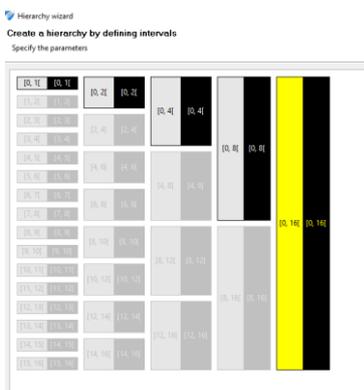
Cliquez sur le premier intervalle, puis mettez la valeur max à 1. Ensuite faites un clic droit sur ce nouvel intervalle et choisissez « add new level » ce qui va rajouter un niveau à la hiérarchie.

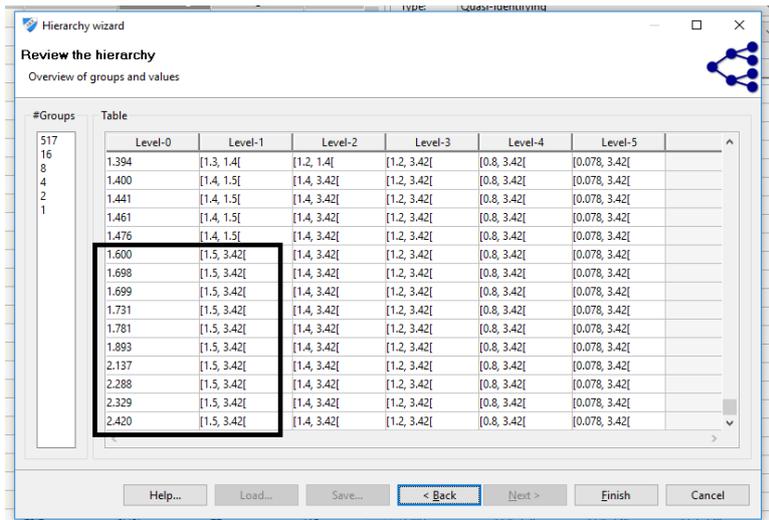


Cliquez ensuite sur ce nouveau niveau et passez la valeur *group* à 2, ce qui signifie qu'on veut grouper 2 intervalles précédents ensemble pour créer ce nouveau niveau. Un nouvel ensemble (l'intervalle [1 ;2[) est créé automatiquement (il est grisé pour indiquer qu'il est construit automatiquement).



Continuez à construire des niveaux jusqu'à arriver au niveau [0,16[Votre hiérarchie doit ressembler à ceci :

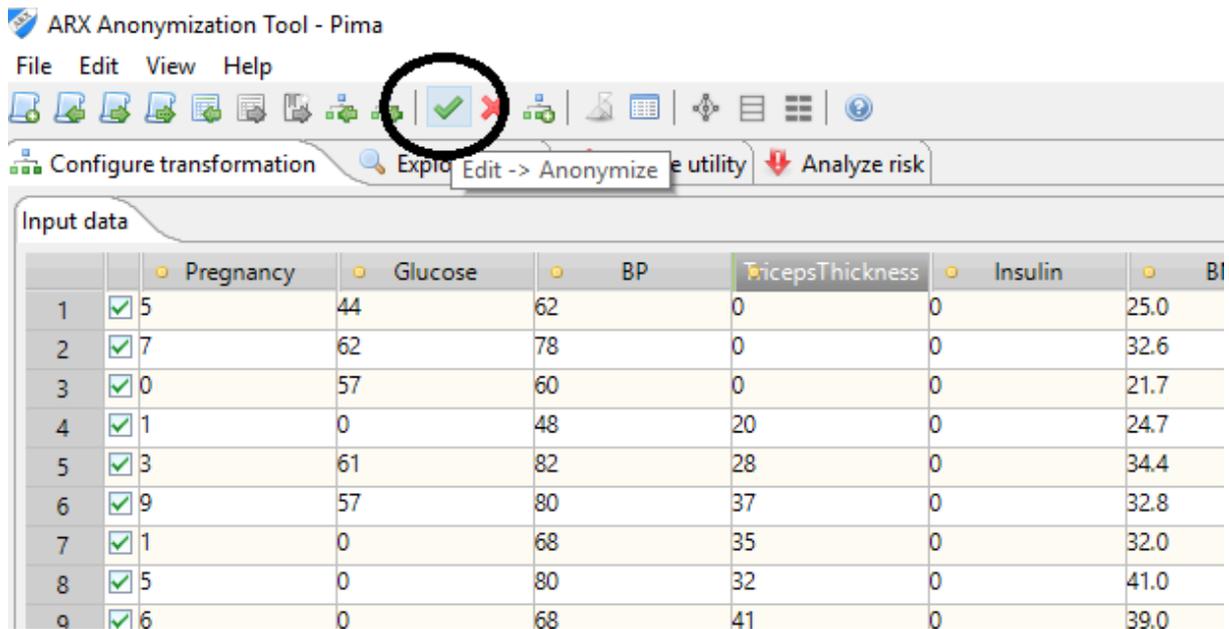




11) Anonymisation selon le k-anonymat

Il ne nous reste plus qu'à définir le modèle d'anonymisation que nous voulons utiliser !

Appuyez sur le bouton + puis choisissez le k-anonymat et k=2. Rajoutons également une limite de suppression de 5%. Puis pressez le bouton « anonymize »



La fenêtre qui s'affiche donne un nombre de pas maximum pour l'anonymisation. Si votre problème d'anonymisation est difficile (i.e. possède beaucoup d'attributs) il faudra peut être augmenter cette valeur. Mais pour notre application, c'est inutile.

Au bout de quelques instants l'anonymisation aura eu lieu.

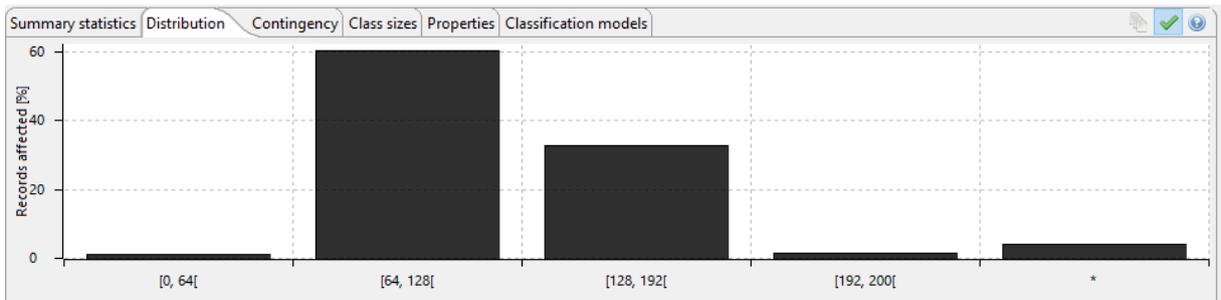
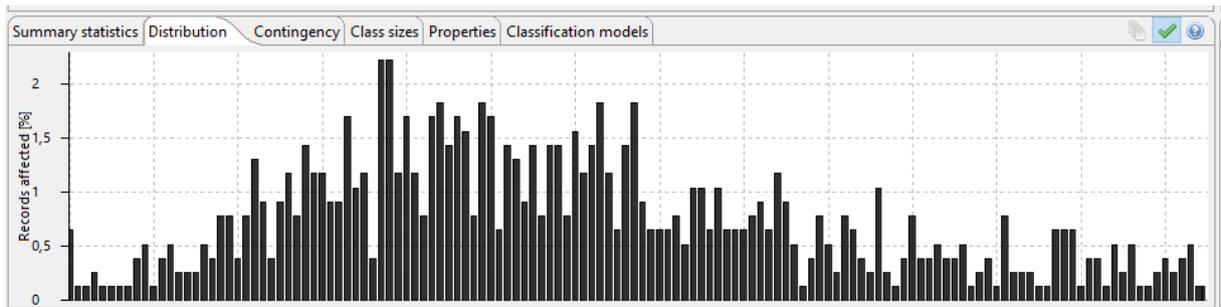
Nous pouvons visualiser les résultats en allant dans Analyze Utility, et en naviguant les onglets Summary statistics, distribution, etc. Comparons les données initiales et les données anonymes.

Measure	Value (incl. suppressed)	Value (excl. suppressed)
Average class size	1 (0.13021%)	1 (0.13021%)
Maximal class size	1 (0.13021%)	1 (0.13021%)
Minimal class size	1 (0.13021%)	1 (0.13021%)
Suppressed records	0 (0%)	0
Number of classes	768	768
Number of records	768	768

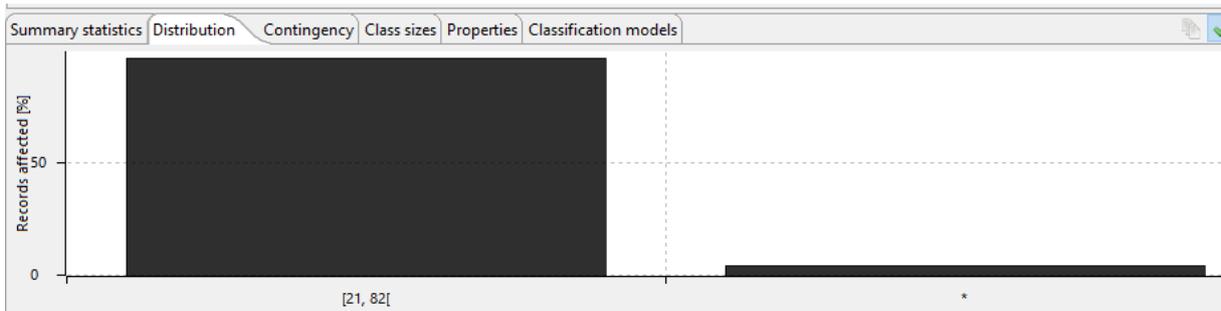
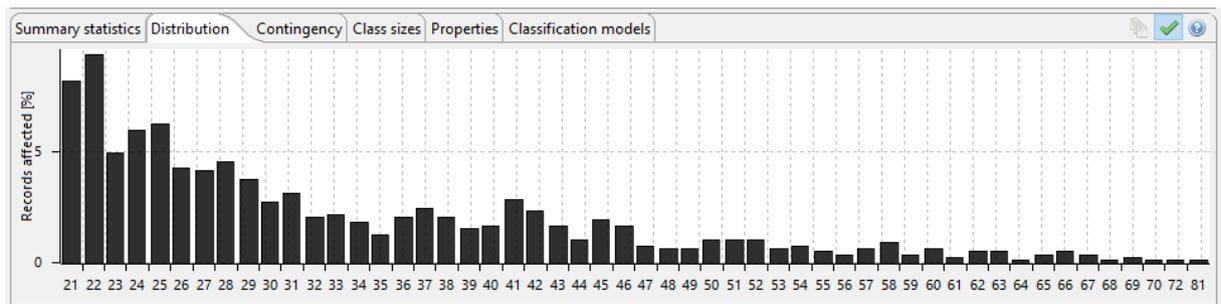
Measure	Value (incl. suppressed)	Value (excl. suppressed)
Average class size	10.20833 (1.32921%)	10.20833 (1.38889%)
Maximal class size	139 (18.09896%)	139 (18.91156%)
Minimal class size	2 (0.26042%)	2 (0.27211%)
Suppressed records	33 (4.29688%)	0
Number of classes	72	72
Number of records	768	735

On voit que les données anonymes n'ont plus que 72 classes (par opposition aux 768 initiales, c'est-à-dire que chaque individu était unique ...) et qu'on a supprimé 33 enregistrements (soit 4.3%).

On peut comparer aussi des distributions, par exemple celle de *glucose* (il faut appuyer sur l'attribut en haut de l'écran)



On voit que l'age a été totalement généralisé (ou supprimé).



Cela signifie que si on voulait utiliser l'age pour faire la prédiction c'est devenu impossible !

Pour changer les critères utilisés pour la généralisation, il faut se rendre dans l'onglet *explore results*. On voit qu'il est possible de changer certains paramètres, par contre ARX n'a pas trouvé de solution sans qu'on généralise l'age. Il serait certainement possible de trouver des solutions, mais il faudrait lui donner plus de temps pour faire le calcul. Pour effectuer une autre transformation, il suffit de cliquer sur la transformation de treillis qui nous intéresse et faire « apply transformation ». On peut par exemple changer le degré de généralisation de certains attributs.

ARX Anonymization Tool - Pima

File Edit View Help

Configure transformation Explore results Analyze utility Analyze risk

Lattice List Tiles

Filter

Attribute	0	1	2	3	4	5	6	7	8	9
BMI	×	×	×	×	×	×	×	×	×	×
DiabetesPedigree	×	×	×	×	×	×	×	×	×	×
Age	×	×	×	×	×	×	×	×	×	×

Anonymous Non-anonymous Unknown

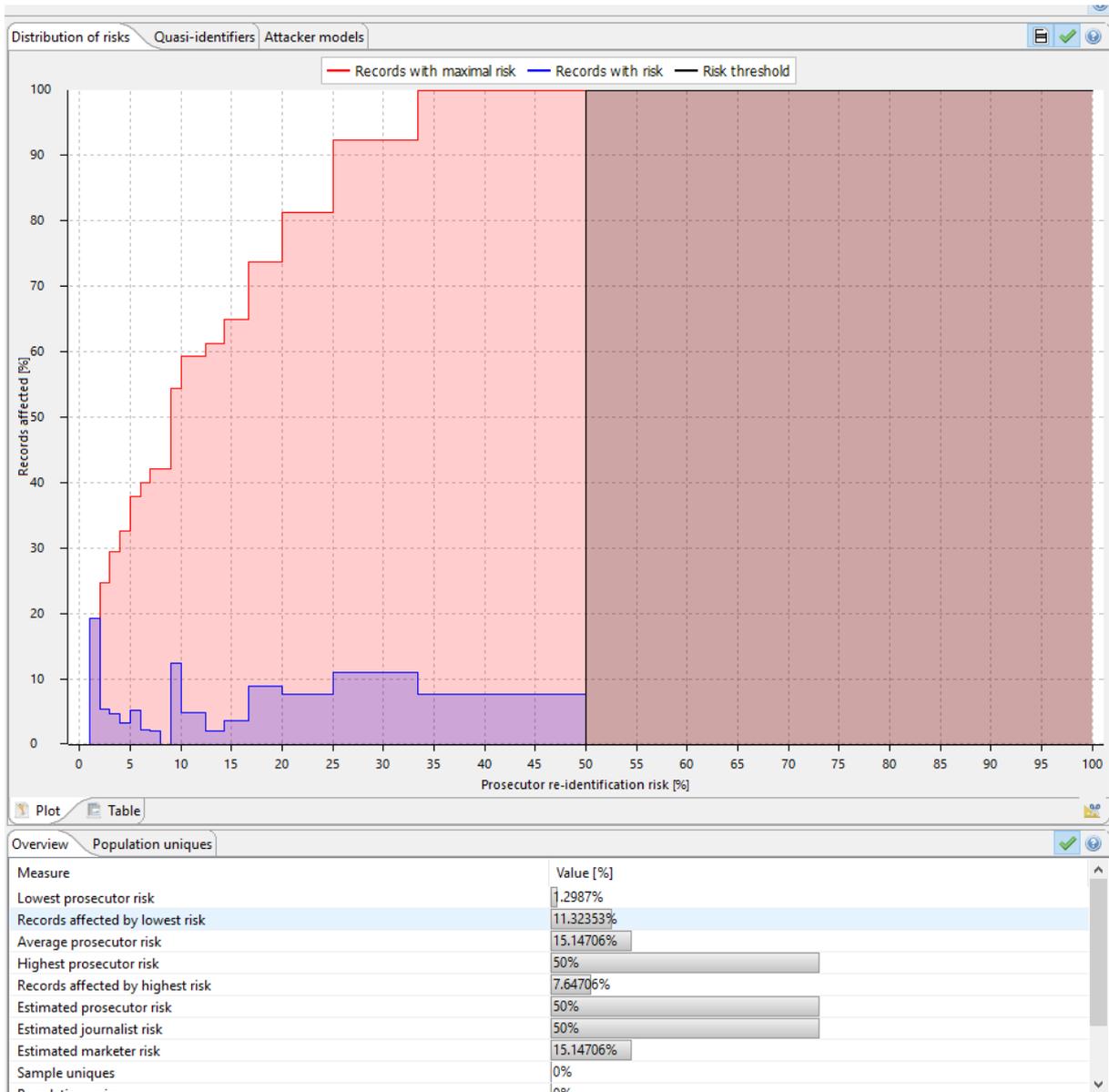
Clipboard

Transformation	Comment
[5, 7, 6, 5, 5, 6, 5, 7]	Optimum in category utility
[4, 5, 6, 4, 6, 6, 5, 7]	Rank 4 in category generalization
[4, 6, 6, 4, 6, 6, 5, 7]	Rank 3 in category utility
[4, 5, 6, 5, 6, 6, 5, 7]	Rank 4 in category utility
[4, 4, 6, 5, 6, 6, 5, 7]	Rank 7 in category generalization
[4, 6, 6, 3, 6, 6, 5, 7]	Rank 2 in category generalization
[4, 9, 6, 5, 5, 6, 5, 7]	Rank 7 in category utility

Properties: Proper, Transform, Anonymity, Score, Success, Predecessor, Check

12) Analyse de risque

L'analyse de risque est disponible via l'onglet Analyze Risk. Le risque initial sur chaque enregistrement de la base est de 100% car chaque enregistrement était unique sur les 8 QID utilisés. Après anonymisation, on peut voir une très grosse baisse de cette possibilité d'attaque via les QID. En effet, le risque maximal est de 50% (c'est-à-dire $1/k$, comme $k=2$), et on voit que seuls 7% des enregistrements de la base atteignent ce risque.



Ce graphique permet de quantifier le risque lié à la réidentification par quasi-identifiant d'un attaquant. C'est au responsable de traitement de données d'accepter ou non ce risque. Souvent une valeur de risque maximale acceptable sera plutôt à moins de 20% (soit $k \geq 5$)

13) Export et Analyse de données

Exportons maintenant nos données (en format .csv).

Input data	Classification performance	Quality models
1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Output data	Classification performance	Quality models
1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Nous pouvons importer ces données dans WEKA. Nous voyons que les données sont désormais moins précises :

Current relation
 Relation: pima-anon
 Instances: 768
 Attributes: 9
 Sum of weights: 768

Attributes

All None Invert Pattern

No.	Name
<input type="checkbox"/>	Pregnancy
<input checked="" type="checkbox"/>	Glucose
<input type="checkbox"/>	BP
<input type="checkbox"/>	TricepsThickness
<input type="checkbox"/>	Insulin
<input type="checkbox"/>	BMI
<input type="checkbox"/>	DiabetesPedigree
<input type="checkbox"/>	Age
<input type="checkbox"/>	Class

Remove

Selected attribute
 Name: Glucose
 Missing: 0 (0%)
 Distinct: 5
 Type: Nominal
 Unique: 0 (0%)

No.	Label	Count	Weight
1	[0, 64[9	9.0
2	[128, 192[251	251.0
3	[192, 200[13	13.0
4	[64, 128[462	462.0
5	*	33	33.0

Class: Class (Nom) Visualize All

Category	Blue (Bottom)	Red (Top)	Total
1	9	0	9
2	251	0	251
3	13	0	13
4	462	0	462
5	33	0	33

Q8- QUESTION : Effectuez les tâches de classification et de clustering avec les paramètres optimaux que vous avez déterminés dans la Partie I. Commentez la qualité des résultats, en particulier les règles générées avec JRIP ou encore l'arbre de décision généré avec J48.

Q9- QUESTION : Choisissez un classifieur. Comparez la qualité de ce classifieur en utilisant plusieurs sortes d'anonymisation : $k=4$, $k=20$, des taux de suppression allant jusqu'à 25%, et en privilégiant certains attributs par rapport à d'autres. Commentez la possibilité d'utiliser l'anonymat pour faire de l'analyse de données.