



République Algérienne Démocratique et Populaire



Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université des Sciences et de la Technologie Houari Boumediene**

**Département Informatique**

**Module : Data Mining**

## **Rapport de la 2eme partie du projet Data Mining**

**Réalisation par :**

**-CHABANE Nouar 201400007379**

**-BEZGALI Meriem 201300005476**

**Groupe : 01**

## **I. Introduction :**

Le Data mining a pour but l'extraction d'une connaissance à partir de grandes quantités de données, par des méthodes automatiques ou semi-automatiques. Elle utilise un ensemble d'algorithmes issus de disciplines scientifiques diverses telles que les statistiques, l'intelligence artificielle ou l'informatique, pour construire des modèles à partir des données, c'est-à-dire trouver des structures intéressantes ou des motifs selon des critères fixés au préalable, et d'en extraire un maximum de connaissances.

Dans ce TP, nous allons essayer de mieux comprendre 3 tâches de Data mining, en application 3 algorithmes qui sont :

- Extraction de motifs : Apriori.
- Classification : KNN,
- Clustering DBSCAN.

## **II. Plan à suivre :**

Afin d'implémenter ces 3 tâches nous avons fixé les objectifs suivants :

- Effectuer une recherche sur les 3 algorithmes et comprendre leurs fonctionnement (état de l'art),
- Implémentation de l'algorithme d'extraction de motifs Apriori,
- Implémentation de l'algorithme de classification KNN,
- Implémentation de l'algorithme de clustering DBSCAN,
- Appliquer les 3 algorithmes sur des datasets de **weka** (numérique et nominal),
- Comparaison des différentes exécution et analyse des 3 algorithmes.

## **III. Etat de l'art :**

### **1. APriori :**

L'algorithme APriori est un algorithme d'exploration de données conçu en 1994, par *Rakesh Agrawal* et *Ramkrishnan Srikant*, dans le domaine de l'apprentissage des règles d'association. Il sert à reconnaître des propriétés qui reviennent fréquemment dans un ensemble de données et d'en déduire une catégorisation, c'est un algorithme classique dans l'exploration de données, il est nommé Apriori car il utilise des connaissances préalables sur les propriétés fréquentes des ensembles d'éléments, il applique une approche itérative ou une recherche par niveau dans laquelle les ensembles d'éléments k-fréquents sont utilisés pour trouver les  $k + 1$  éléments suivants.

## A) Quelques notions utiles pour l'algorithme d'Apriori :

### 1. Une règle d'association :

Une règle d'association est une application sous la forme  $X \rightarrow Y$ , où  $X$  et  $Y$  sont des ensembles d'items disjoints. La force d'une règle d'association peut être mesurée en utilisant son support et sa confiance.

Le problème de la recherche de règle d'association peut se formuler comme suit : Etant donnée un ensemble de transaction  $T$ , trouver toutes les règles d'association ayant un support  $\geq \text{minsup}$  et une confiance  $\geq \text{minconf}$  où  $\text{minsup}$  et  $\text{minconf}$  sont des seuils pour le support et la confiance

### 2. Support minimum :

Le support d'un motif d'association est le pourcentage des transactions de données relatives aux tâches pour lesquelles le modèle est vrai.

### 3. Confiance :

La confiance est définie comme la mesure de la certitude ou la fiabilité associée à chaque découverte modèle.

## B) Principe général d'Apriori :

Le principe général de l'algorithme est donné ci-dessous comme suit :

Pour une base de données de transactions «  $T$  », ainsi qu'un seuil de support «  $\text{minSupport}$  », un ensemble de candidats  $C_k$  initialisé à vide au départ, et une liste d'item fréquent  $L_k$  initialisée à vide on commence par calculer à partir de «  $T$  »  $C_1$  en incrémentant de 1 la fréquence de chaque Item si on le trouve dans la base «  $T$  » ensuite on élimine les items qui ont un support inférieur à «  $\text{minSupport}$  » pour obtenir  $L_1$ , puis on calcule  $C_2$  en prenant chaque éléments de  $L_1$  jointure élément suivant de la table  $L_1$  jusqu'à parcourir toute cette table, et ainsi de suite le processus se répète jusqu'à ce qu'on ne puisse plus générer de nouveau candidats, et les itemset fréquent sont obtenus en faisant l'union des  $L_i$  trouvés.

### 2. KNN :

La méthode des  $K$  plus proches voisins est une méthode d'apprentissage supervisé. En abrégé K-NN ou KNN, de l'anglais *k-nearest neighbors*, inventé par Marcello Pelillo en 1967.

Dans ce cadre, on dispose d'une base de données d'apprentissage constituée de  $N$  couples « entrée-sortie ». Pour estimer la sortie associée à une nouvelle entrée  $x$ , la méthode des  $k$  plus proches voisins consiste à prendre en compte (de façon identique) les  $k$  échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée  $x$ , selon une distance à définir. Par exemple, dans un problème de classification, on retiendra la classe la plus représentée parmi les  $k$  sorties associées aux  $k$  entrées les plus proches de la nouvelle entrée  $x$ .

Les voisins sont pris depuis un ensemble d'objets pour lesquels la classe (en classification k-NN) est connue. Ceci peut être considéré comme l'ensemble d'entraînement pour l'algorithme, bien qu'un entraînement explicite ne soit pas particulièrement requis.

**Principe de K-NN : dis-moi qui sont tes voisins, je te dirais qui tu es !**

A) Quelques notions utiles pour l'algorithme KNN :

Afin de calculer la similarité entre une instance et le dataset de training nous devons appliquer certaine formule de calcul de distance. Nous allons citer quelques-unes dans ce qui suit.

1. Distance (cas nominal et numérique) :

Il existe plusieurs formules pour calculer la distance entre les voisins, pour le cas numérique nous avons utilisé la distance euclidienne, qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points

$$D_e(x, y) = \sqrt{\sum_{j=1}^n (x_j - y_j)^2}$$

Pour le cas nominal, nous avons calculé la distance entre un attribut d'une instance donnée avec une autre instance, si les 2 attributs ont la même valeur alors on met distance =0 sinon on la met à 1.

$$D_h(x, y) = \sum_{i=1}^k |x_i - y_i|$$

avec

- $x = y \implies D = 0$
- $x \neq y \implies D = 1$

2. Le paramètre empirique K :

Le K représente le nombre de voisins à prendre en considération pour la classification de l'instance (on prend la classe des K plus fréquentes instances), ce paramètre est laissé à l'utilisateur pour l'entrer.

### 3. DBSCAN :

DBSCAN est un algorithme de clustering de données proposées par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu en 1996. Il s'agit d'un algorithme de clustering basé sur la densité car il trouve un numéro des clusters à partir de la distribution de densité estimée des nœuds correspondants. DBSCAN est l'un des algorithmes de classification les plus courants et aussi le plus cité dans la littérature scientifique.

#### A) Quelques notions utiles pour DBSCAN :

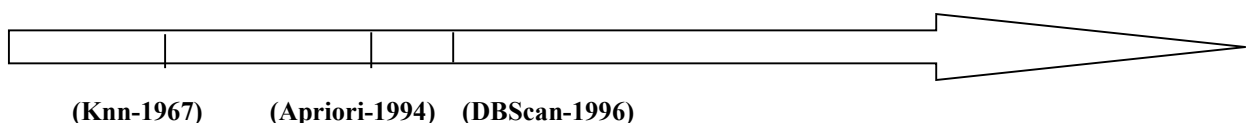
DBSCAN utilise 2 paramètres qui sont :

1. **Eps** : représente la distance maximale pour qu'un point soit considéré dans un cluster.
2. **MinPts** : le nombre minimum de points devant se trouver dans un rayon pour que ces points soient considérés comme un cluster

#### B) Principe général de DBSCAN :

DBScan a besoin de deux paramètres : « Eps » et le nombre minimum de points requis pour former un cluster « MinPts ». Cela commence par un point de départ arbitraire qui n'a pas été visité. Le voisinage de ce point est récupéré et, s'il contient suffisamment de points, un cluster est démarré. Sinon, le point est appelé bruit. Notez que ce point pourrait ultérieurement être trouvé dans un environnement suffisamment dimensionné d'un point différent et donc faire partie d'un cluster. Si un point est reconnu comme faisant partie d'un cluster, son voisinage est également inclus dans ce cluster. Par conséquent, tous les points qui se trouvent dans le voisinage sont ajoutés, de même que leur propre voisinage. Ce processus se poursuit jusqu'à ce que le cluster soit complètement trouvé. Ensuite, un nouveau point non visité est extrait et traité, ce qui conduit à la découverte d'un autre cluster ou bruit.

- **Ordre chronologique d'apparition des 3 algorithmes :**



### IV. Conception et structures de données :

Dans cette partie nous allons présenter les structures de donnée utilisée ainsi que les pseudo-code des algorithmes implémenté.

#### A) Structure de données :

Nous n'avons utilisé aucune structure de données pour la manipulation des instances dans les trois algorithmes, car nous avons utilisé la class **Instances** de **Weka** qui nous a permis de stocker les instances des fichiers «.arff» pour ensuite effectué une normalisation grâce a une

méthode que nous avons implémenté et par la suite d'appliqué un des 3 algorithmes implémentés.

### 1) Structure de donnée des règles d'association :

Pour les règles d'associations d'apriori nous les avons stockés dans une liste chaînée de type **String** qui a la structure suivante :

Règle1	Règle2	Règle3	Règle4	.....	Règle N
--------	--------	--------	--------	-------	---------

Où une règle est de la forme :

Partie droite<string>	Partie gauche<string>
-----------------------	-----------------------

### 2) Structure de donnée des tables Lk et Ck :

Nous avons utilisé 2 matrices L et C dans Apriori, qui représente respectivement l'ensemble des itemset fréquent et l'ensemble des candidats, qui possèdent la structure suivante :

ItemSet	Nombre d'occurrence
Itemset1	Occ1
	↓
ItemSetn	Occn

### 3) Structure de donnée des clusters :

Pour les clusters nous les avons stockés dans des listes chaînées de type <String>.

Cluster1

Element1	Element2	Element3	Element4	.....	Element N
----------	----------	----------	----------	-------	-----------



ClusterN

Element1	Element2	Element3	Element4	.....	Element M
----------	----------	----------	----------	-------	-----------

## B) Pseudocode :

### 1. Apriori :

```
Procedure-Apriori(T,minSupport)
{
    Ck : candidate itemset of size k
    Lk :frequent itemset of size k
    L1={frequent items} ;
    For(k=1 ;Lk != empty ; k++)
    Do begin
        Ck+1=candidates generated from Lk ;
        for each transaction t in Database T Do begin
            increment the count of all candidates in Ck+1 that contained in T
        Lk+1=candidates in Ck+1 with min support
    } end
    Return Lk ;
```

### 2. KNN :

```
1. Load the training and test data
2. Choose the value of K
3. For each point in test data:
- find the Euclidean distance to all training data points
- store the Euclidean distances in a list and sort it
- choose the first k points
- assign a class to the test point based on the majority of classes present in the chosen points
4. End
```

### 3. DBSCAN :

La fonction DBSCAN prends en entrée les paramètres Eps, minPts, les instances et distFunc : la fonction de distance.

```

DBSCAN(DB, distFunc, eps, minPts) {
  C = 0      /* Cluster counter */
  for each point P in database DB {
    if label(P) ≠ undefined then continue /* Previously processed in inner loop */
    Neighbors N = RangeQuery(DB, distFunc, P, eps) /* Find neighbors */
    if |N| < minPts then { /* Density check */
      label(P) = Noise /* Label as Noise */
      continue
    }
    C = C + 1 /* next cluster label */
    label(P) = C /* Label initial point */
    Seed set S = N \ {P} /* Neighbors to expand */
    for each point Q in S { /* Process every seed point */
      if label(Q) = Noise then label(Q) = C /* Change Noise to border point */
      if label(Q) ≠ undefined then continue /* Previously processed */
      label(Q) = C /* Label neighbor */
      Neighbors N = RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */
      if |N| ≥ minPts then { /* Density check */
        S = S ∪ N /* Add new neighbors to seed set */
      }
    }
  }
}

```

**RangeQuery** : elle prend un élément de la base. Elle calcule la distance entre cet élément et tous les autres éléments de la base avec la fonction **distFunc**, et retourne les éléments dont la distance est inférieure à EPS.

```

RangeQuery(DB, distFunc, Q, eps) {
  Neighbors = empty list
  for each point P in database DB { /* Scan all points in the database */
    if distFunc(Q, P) ≤ eps then { /* Compute distance and check epsilon */
      Neighbors = Neighbors ∪ {P} /* Add to result */
    }
  }
  return Neighbors
}

```



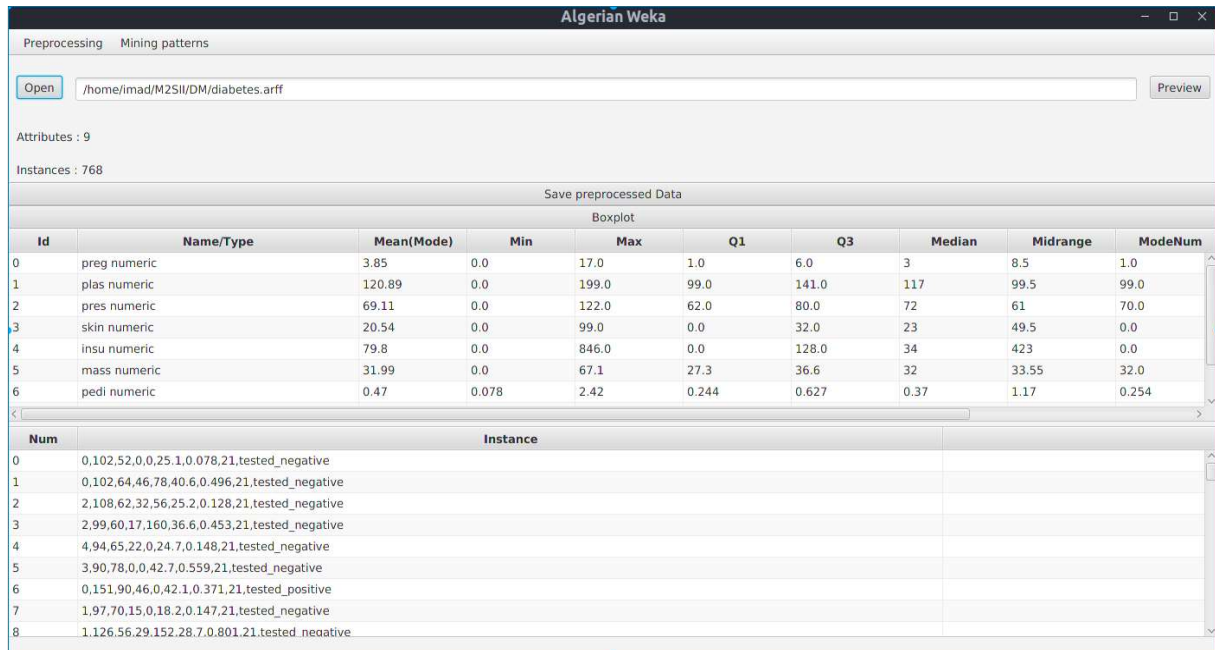
## V. Application des algorithmes sur des dataset de weka :

Dans ce qui suit nous allons présenter des screens des différentes exécutions des 3 algorithmes sur des attributs nominal et numérique.

### A) Cas numérique :

Pour la démonstration on a utilisé un dataset numérique « diabètes.arff » .

**DataSet** avant normalisation :

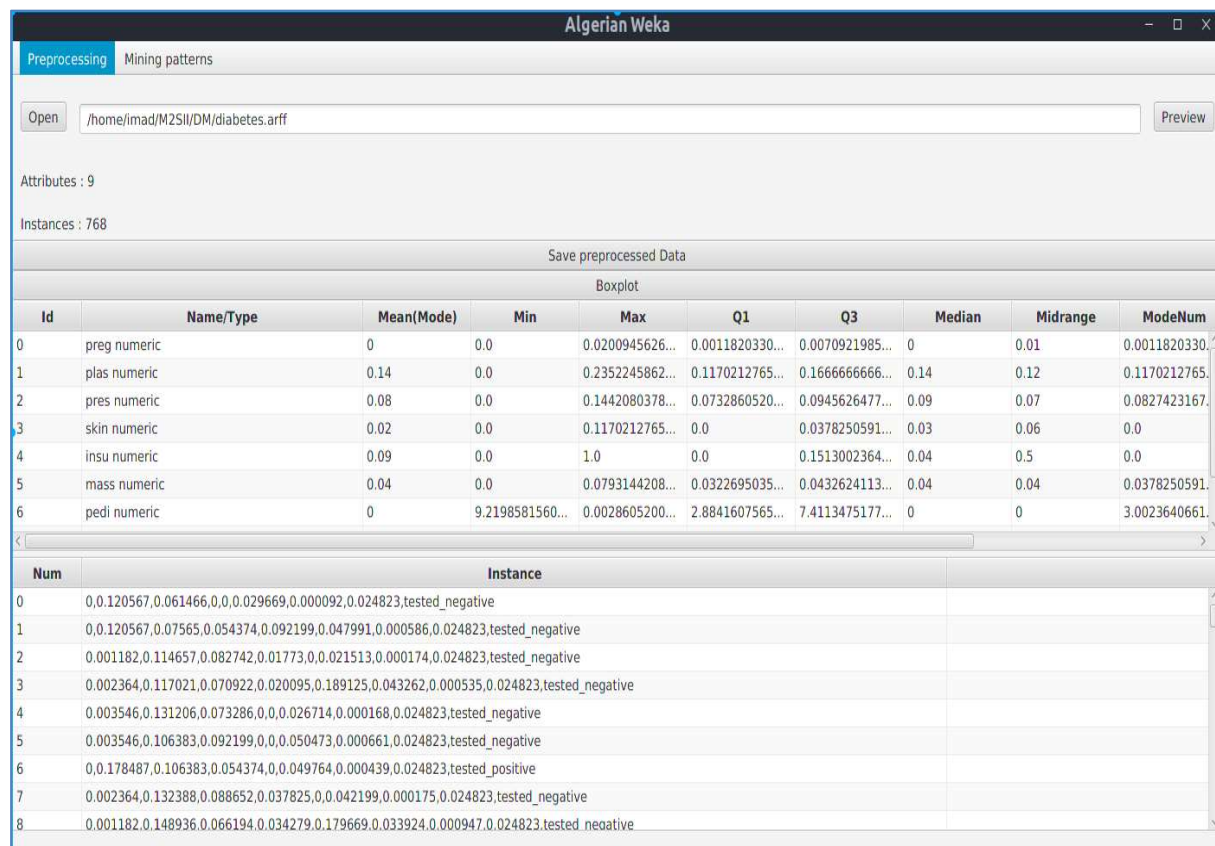


Boxplot									
Id	Name/Type	Mean(Mode)	Min	Max	Q1	Q3	Median	Midrange	ModeNum
0	preg numeric	3.85	0.0	17.0	1.0	6.0	3	8.5	1.0
1	plas numeric	120.89	0.0	199.0	99.0	141.0	117	99.5	99.0
2	pres numeric	69.11	0.0	122.0	62.0	80.0	72	61	70.0
3	skin numeric	20.54	0.0	99.0	0.0	32.0	23	49.5	0.0
4	insu numeric	79.8	0.0	846.0	0.0	128.0	34	423	0.0
5	mass numeric	31.99	0.0	67.1	27.3	36.6	32	33.55	32.0
6	pedi numeric	0.47	0.078	2.42	0.244	0.627	0.37	1.17	0.254

Num	Instance
0	0,102,52,0,0,25,1,0,078,21,tested_negative
1	0,102,64,46,78,40,6,0,496,21,tested_negative
2	2,108,62,32,56,25,2,0,128,21,tested_negative
3	2,99,60,17,160,36,6,0,453,21,tested_negative
4	4,94,65,22,0,24,7,0,148,21,tested_negative
5	3,90,78,0,0,42,7,0,559,21,tested_negative
6	0,151,90,46,0,42,1,0,371,21,tested_positive
7	1,97,70,15,0,18,2,0,147,21,tested_negative
8	1,126,56,29,152,28,7,0,801,21,tested_neogative

**DataSet** après normalisation :



Boxplot									
Id	Name/Type	Mean(Mode)	Min	Max	Q1	Q3	Median	Midrange	ModeNum
0	preg numeric	0	0.0	0.0200945626...	0.0011820330...	0.0070921985...	0	0.01	0.0011820330...
1	plas numeric	0.14	0.0	0.2352245862...	0.1170212765...	0.1666666666...	0.14	0.12	0.1170212765...
2	pres numeric	0.08	0.0	0.1442080378...	0.0732860520...	0.0945626477...	0.09	0.07	0.0827423167...
3	skin numeric	0.02	0.0	0.1170212765...	0.0	0.0378250591...	0.03	0.06	0.0
4	insu numeric	0.09	0.0	1.0	0.0	0.1513002364...	0.04	0.5	0.0
5	mass numeric	0.04	0.0	0.0793144208...	0.0322695035...	0.0432624113...	0.04	0.04	0.0378250591...
6	pedi numeric	0	9.2198581560...	0.0028605200...	2.8841607565...	7.4113475177...	0	0	3.0023640661...

Num	Instance
0	0,0.120567,0.061466,0,0,0.029669,0.000092,0.024823,tested_negative
1	0,0.120567,0.07565,0.054374,0.092199,0.047991,0.000586,0.024823,tested_negative
2	0.001182,0.114657,0.082742,0.01773,0,0.021513,0.000174,0.024823,tested_negative
3	0.002364,0.117021,0.070922,0.020095,0.189125,0.043262,0.000535,0.024823,tested_negative
4	0.003546,0.131206,0.073286,0,0,0.026714,0.000168,0.024823,tested_negative
5	0.003546,0.106383,0.092199,0,0,0.050473,0.000661,0.024823,tested_negative
6	0,0.178487,0.106383,0.054374,0,0.049764,0.000439,0.024823,tested_positive
7	0.002364,0.132388,0.088652,0.037825,0,0.042199,0.000175,0.024823,tested_negative
8	0.001182,0.148936,0.066194,0.034279,0.179669,0.033924,0.000947,0.024823,tested_negative

1) Exécution de l'algorithme Apriori :

Load:

Load preprocesse...

Confidence:

0.5

Minimum support:

10

Find

Id	Itemset
0	duration__0.025 wage-increase-first-year__0.07 wage-increase-second-year__0.099293 wage-increase-third-year__0.099293
1	duration__0.025 wage-increase-first-year__0.0525 wage-increase-second-year__0.099293 wage-increase-third-year__0.099293
2	duration__0.075 wage-increase-first-year__0.15 wage-increase-second-year__0.15 wage-increase-third-year__0.15
3	duration__0.025 wage-increase-first-year__0.15 wage-increase-second-year__0.099293 wage-increase-third-year__0.099293

Id	Patterns
0	[bereavement-assistance__yes, class__good, contribution-to-health-plan__full, cost-of-living-adjustment__none]
1	[bereavement-assistance__yes, class__good, contribution-to-health-plan__full, cost-of-living-adjustment__none]

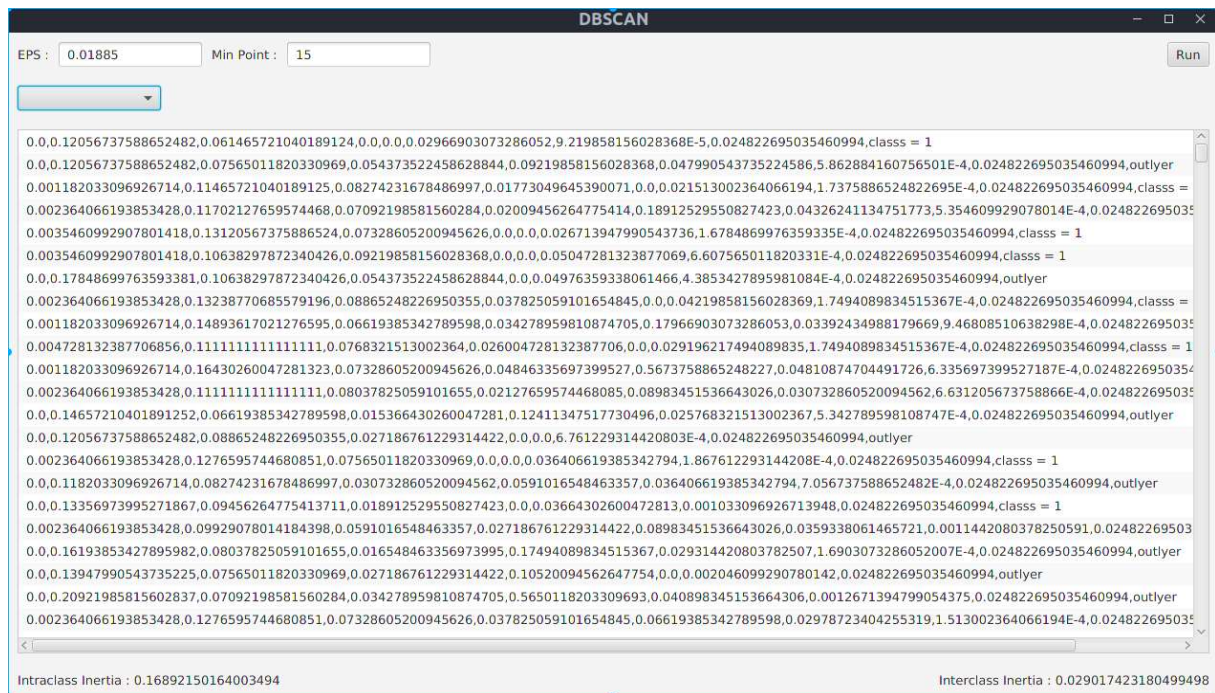
Id	Rule
0	[bereavement-assistance__yes]->[class__good]
1	[class__good]->[bereavement-assistance__yes]
2	[bereavement-assistance__yes]->[contribution-to-health-plan__full]
3	[contribution-to-health-plan__full]->[bereavement-assistance__yes]

Runing Time:3 seconds

## 2) Exécution de l'algorithme KNN :

The screenshot shows the KNN Algorithm interface. At the top, there are input fields for 'divide by (%)' (set to 70) and 'K nearest neighbours' (set to 10), with 'split' and 'Run' buttons. Below these, the 'model set' is displayed as a list of 10 data points, each with 14 numerical features and a categorical label (tested\_negative or tested\_positive). The 'training set' is also displayed as a list of 10 data points, each with 14 numerical features and a categorical label. To the right of the training set, the 'predicted classes' are listed, showing the predicted label for each training set point. At the bottom, the 'Predictions correctes' are 84.84848484848484% and the 'Predictions faux' are 15.1515151515152%.

### 3) Exécution de l'algorithme DBSCAN :



### B) Cas nominal :

Pour un dataset nominal on a utilisé « contact-lenses.arff » .

Algerian Weka									
Preprocessing Mining patterns									
Open /home/imad/M2SII/DM/contact-lenses.arff Preview									
Attributes : 5									
Instances : 24									
Save preprocessed Data									
Boxplot									
Id	Name/Type	Mean(Mode)	Min	Max	Q1	Q3	Median	Midrange	ModeNum
0	age {young,pre-presbyopic,presbyopic}	young	---	---	---	---	---	---	0.0
1	spectacle-prescrip {myope,hypermetrope}	myope	---	---	---	---	---	---	0.0
2	astigmatism {no,yes}	no	---	---	---	---	---	---	0.0
3	tear-prod-rate {reduced,normal}	reduced	---	---	---	---	---	---	0.0
4	contact-lenses {soft,hard,none}	none	---	---	---	---	---	---	0.0
Num	Instance								
0	young,myope,no,reduced,none								
1	young,myope,no,normal,soft								
2	young,myope,yes,reduced,none								
3	young,myope,yes,normal,hard								
4	young,hypermetrope,no,reduced,none								
5	young,hypermetrope,no,normal,soft								
6	young,hypermetrope,yes,reduced,none								
7	young,hypermetrope,yes,normal,hard								
8	pre-presbyopic,myope,no,reduced,none								



### 1) Apriori :

1050 x 659

Load:

Confidence:

Minimum support:

Id	Itemset
0	duration__0.025 wage-increase-first-year__0.07 wage-increase-second-year__0.099293 wage-increase-third-year__0.097833 cost-of-living-adjustment__none
1	duration__0.025 wage-increase-first-year__0.0525 wage-increase-second-year__0.099293 wage-increase-third-year__0.097833 cost-of-living-adjustment__none
2	duration__0.075 wage-increase-first-year__0.15 wage-increase-second-year__0.15 wage-increase-third-year__0.1 cost-of-living-adjustment__none
3	duration__0.025 wage-increase-first-year__0.15 wage-increase-second-year__0.099293 wage-increase-third-year__0.097833 cost-of-living-adjustment__none

Id	Patterns
0	[bereavement-assistance__yes, class__good, contribution-to-health-plan__full, cost-of-living-adjustment__none, education-allowance__no, longterm-care__no]
1	[bereavement-assistance__yes, class__good, contribution-to-health-plan__full, cost-of-living-adjustment__none, education-allowance__no, longterm-care__no]

Id	Rule
0	[bereavement-assistance__yes]->[class__good]
1	[class__good]->[bereavement-assistance__yes]
2	[bereavement-assistance__yes]->[contribution-to-health-plan__full]
3	[contribution-to-health-plan__full]->[bereavement-assistance__yes]

Running Time:4 seconds

### 2) KNN :

KNN Algorithm

divide by (%) :

K nearest neighbours :

model set :

presbyopic,hypermetrope,yes,normal,none
presbyopic,hypermetrope,yes,reduced,none
presbyopic,hypermetrope,no,normal,soft
presbyopic,hypermetrope,no,reduced,none
presbyopic,myope,yes,normal,hard
presbyopic,myope,yes,reduced,none
presbyopic,myope,no,normal,none
presbyopic,myope,no,reduced,none

training set :

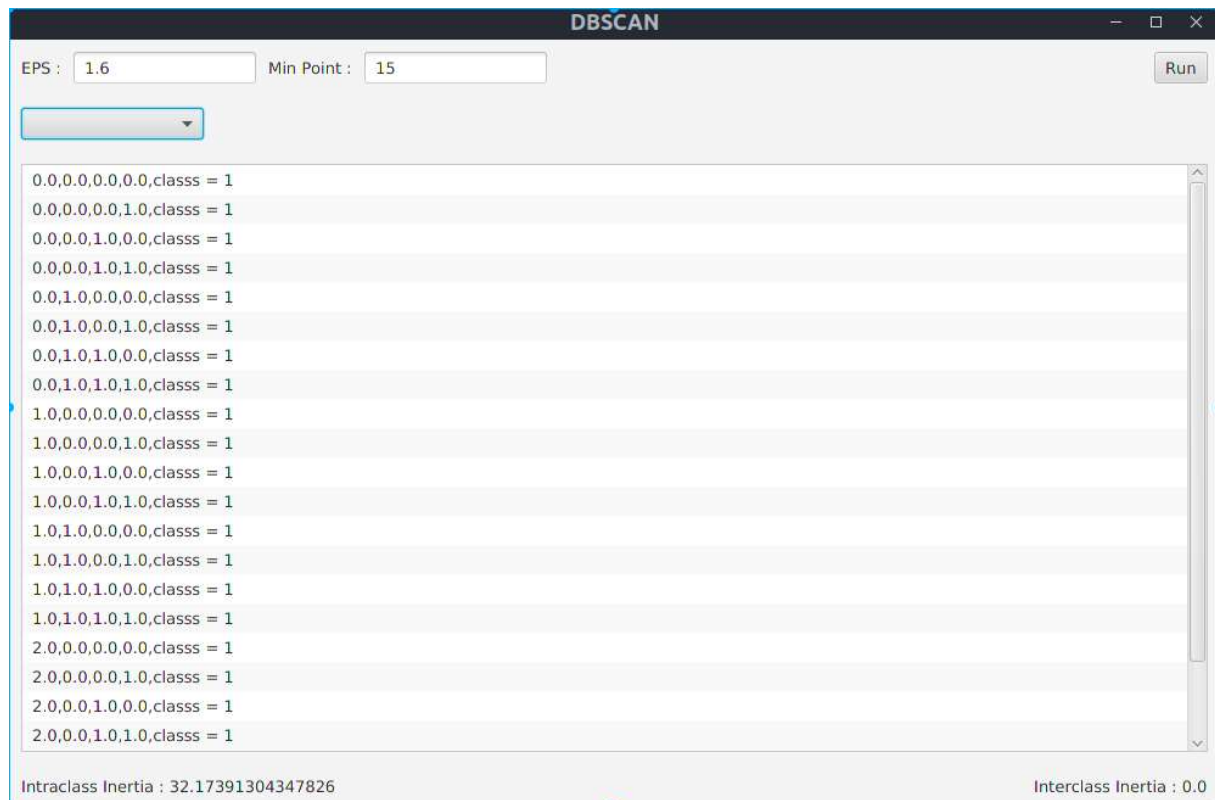
young,myope,no,reduced,none
young,myope,no,normal,soft
young,myope,yes,reduced,none
young,myope,yes,normal,hard
young,hypermetrope,no,reduced,none
young,hypermetrope,no,normal,soft
young,hypermetrope,yes,reduced,none
young,hypermetrope,yes,normal,hard

predicted classes :

none
soft
none
hard
none
soft
none
none

Predictions correctes: 87.5% Predictions faux: 12.5%

### 3) DBSCAN :



## VI. Conclusion :

Afin d'analyser et de comparer les différents algorithmes avec leurs paramètres empiriques, nous allons tout d'abord dresser un tableau montrant les paramètres choisis et les différents résultats obtenus par chaque algorithme.

1) Apriori :

Pour un ensemble **d** d'itemset fréquent extrait avec Apriori nous obtenant grâce à la formule suivante, le nombre total possible de règles d'associations :

$$R = \sum_{k=1}^{d-1} \left[ \binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right] = 3^d - 2^{d+1} + 1$$

Où **d** : est le nombre d'itemset fréquent extrait par Apriori. Par exemple, si nous avons extrait 6 itemset fréquent, le nombre de règles sera = 602.

Cette formule nous a permis de savoir et de vérifier au préalable si notre génération de règles d'association était correcte à partir d'un certain nombre d'itemset **d**, ce qui est le cas.

Tableau des différents résultats obtenus en appliquant Apriori sur le dataset **car.Arff** en faisant varier les paramètres Min\_sup et Min\_confidence:

Min_sup	Min_confidence	Nbr_freq_pat	Nbr_règles	Temps execution
50%	50%	1	0	1s
25%	50%	3	3	3s
12%	50%	21	20	7s
50%	25%	1	0	1s
25%	25%	3	6	3s
12%	25%	21	34	10s

On remarque qu'Apriori a un temps d'exécution assez court pour ce dataset car il contient que 1728 instances, il trouve des règles d'associations quand Min\_Sup est < 50% car avec ce Min\_Sup il obtient un seul item fréquent et on ne peut pas former une règle avec, donc on a décidé de réduire Min\_Sup à 25% et une confiance = 50% nous avons obtenu 3 règles d'association en un laps de temps = 3s, ensuite on a encore diminué Min\_Sup et laissé la Min\_conf à 50% ce qui a généré plus d'item fréquent et donc plus de règles d'association. par la suite nous avons décidé de fixer Min\_confiance et de faire varier Min\_Sup on remarque que quand Min\_Sup est petit (exemple Min\_Sup=12% et Min\_confiance =25%) on génère plus d'itemset fréquent et plus de règles d'association donc le temps d'exécution augmente plus, et même chose lorsqu'on a fixé Min\_confiance et réduit Min\_Sup.

Ce dataset reste relativement petit comparé aux bases de données qui contiennent des millions et des millions de transaction donc on peut déduire que la complexité temporelle d'Apriori est très grande pour des bases de données volumineuses.

- Inconvénient d'Apriori : le nombre considérable d'accès à la base de données, et un temps d'exécution exponentiel pour de grande base de données.
- Avantage d'Apriori : est qu'il est simple, et facile à implémenter.

## 2) Knn :

Afin de mesurer l'efficacité de l'algorithme KNN implémenté, nous avons utilisé une formule pour calculer le taux d'erreur qui est comme suit :

$$Erreur = \frac{\text{nombre de prédiction fausse}}{\text{nombre total d'instance}}$$

Tableau des différents résultats obtenus en appliquant Knn sur le dataset **Labor.Arff** et en faisant varier le paramètre K :

K	% instance correctement classé	%instance mal classé
10	44.44%	55.55%
20	27.77%	72.22%
5	61.11%	38.88%
3	72.22%	27.77%
2	72.22%	27.77%
1	72.22%	27.77%

On remarque que pour KNN les résultats s'améliorent quand vraiment on a décidé de prendre les K les plus proches voisins, car à partir de K=3 jusqu'à K=1 les résultats sont restés les mêmes et sont relativement meilleurs que les autres instanciations de K prise auparavant.

- Limitations de K-NN :

K-NN est un algorithme assez simple à appréhender. Principalement, grâce au fait qu'il n'a pas besoin de modèle pour pouvoir effectuer une prédiction. Le contre coût est qu'il doit **garder en mémoire l'ensemble des observations** pour pouvoir effectuer sa prédiction. Ainsi il faut faire attention à **la taille du jeu d'entraînement**. Également, le choix de la méthode de calcul de la distance ainsi que le nombre de voisins peuvent ne pas être évident. Il faut essayer plusieurs combinaisons et faire du tuning de l'algorithme pour avoir un résultat satisfaisant.

### 3) DBSCAN :

Pour DBSCAN nous avons décidé de prendre un **Eps** assez petit car les instances sont normalisées et leurs valeurs sont comprises entre 0 et 1.

Tableau des différents résultats obtenus en appliquant DBSCAN sur le dataset **CPU.Arff** et en faisant varier les paramètres EPS et Min\_pts:

EPS	Min_pts	Interclass	Intraclass	Nbr de cluster
<b>0.01</b>	<b>20</b>	<b>0.0032</b>	<b>0.0012</b>	<b>1</b>
<b>0.01</b>	<b>10</b>	<b>0.2386</b>	<b>0.0038</b>	<b>8</b>
<b>0.02</b>	<b>10</b>	<b>0.2159</b>	<b>0.0597</b>	<b>5</b>
<b>0.03</b>	<b>10</b>	<b>0.2092</b>	<b>0.2288</b>	<b>4</b>
<b>0.04</b>	<b>10</b>	<b>0.1753</b>	<b>0.3741</b>	<b>3</b>
<b>0.05</b>	<b>10</b>	<b>0.1753</b>	<b>0.3741</b>	<b>3</b>
<b>0.05</b>	<b>10</b>	<b>0.1753</b>	<b>0.3741</b>	<b>3</b>
<b>0.001</b>	<b>10</b>	<b>1.2561</b>	<b>0.1036</b>	<b>9</b>
<b>0.002</b>	<b>10</b>	<b>0.2623</b>	<b>4.2410</b>	<b>9</b>
<b>0.003</b>	<b>10</b>	<b>8.1617</b>	<b>0.2565</b>	<b>9</b>
<b>0.0001</b>	<b>10</b>	<b>-</b>	<b>-</b>	<b>No cluster</b>

Nous avons remarqué que pour Min\_pts=10 et EPS= 0.003 on a obtenu la plus grande Interclass entre les clusters car EPS est le plus petit choisi, et le second meilleur résultat est pour EPS=0.001 et Min\_pts=10 qui est relativement bon comparé au reste des exécutions faites avec d'autres instanciations de Min\_pts et Eps.

- Avantages de DBSCAN:

DBScan n'exige pas que vous connaissiez a priori le nombre de clusters dans les données, par opposition à k-means. DBScan peut trouver des grappes de forme arbitraire. Il peut même trouver des grappes complètement entourées par (mais non connectées à) une grappe différente. En raison du paramètre MinPts, l'effet appelé lien unique (les différentes grappes étant connectées par une mince ligne de points) est réduit. DBScan a une notion de bruit. DBScan nécessite seulement deux paramètres et est généralement insensible à l'ordre des points dans la base de données. (Seuls les points situés au bord de deux clusters différents peuvent échanger l'appartenance à un cluster si leur ordre est modifié et que l'affectation de cluster n'est unique que jusqu'à l'isomorphisme.)

- Désavantages de DBSCAN:

DBSCAN ne peut donner lieu à un bon regroupement que si sa mesure de distance est dans la fonction getNeothers (P, epsilon). La métrique de distance la plus couramment utilisée est la mesure de distance euclidienne. Surtout pour les données de grande dimension, cette métrique de distance peut être rendue presque inutile. DBScan ne répond pas bien aux ensembles de données de densités variables (appelés ensembles de données hiérarchiques).



## **Bibliographie et références :**

1. Livre de Han : Data Mining \_ Concepts and Techniques
2. Dr S. BABA-ALI cours CLUSTERING PDF
3. <http://37steps.com/4370/nn-rule-invention/>
4. [https://fr.wikipedia.org/wiki/M%C3%A9thode\\_des\\_k\\_plus\\_proches\\_voisins](https://fr.wikipedia.org/wiki/M%C3%A9thode_des_k_plus_proches_voisins)
5. <https://fr.wikipedia.org/wiki/DBSCAN>
6. [https://fr.wikipedia.org/wiki/Algorithme\\_APriori](https://fr.wikipedia.org/wiki/Algorithme_APriori)
7. [https://people.revoledu.com/kardi/tutorial/KNN/KNN\\_Numerical-example.html](https://people.revoledu.com/kardi/tutorial/KNN/KNN_Numerical-example.html)