

Extraction de Connaissances B.E. 3

Manipulation des méthodes d'apprentissage avec WEKA

Classification par Clustering

ECL - Option Informatique - 2024/2025

Déc 2024

Table des matières

1	Classification Non supervisée (Clustering)	3
2	Clustering sous Weka	3
2.1	Pratique de la méthode K-means et la BD météo	3
2.2	Remarques	6
2.3	A propos des paramètres de K-means	7
2.3.1	Initialisation des centroïdes dans K-means	7
3	Aperçu de quelques méthodes de Clustering sous Weka	9
4	Classification et Clustering	10
5	Les résultats du Clustering	11
5.1	Utilisation de SciKitLearn sous Weka	12
6	Travail en séance et à rendre	13
7	Travail A	13
7.1	K-means et Farthest First sur la BD. Weather	13
7.2	Comparaison de clusters	15
7.3	Évaluation par rapport à un attribut de classe	15
7.4	Choisir un modèle obtenu par K-means	18
7.5	FilteredClusterer	18
8	Travail B	19
8.1	La méthode DBSCAN et la BD. Vote	19
8.2	Méthode probabiliste EM et la BD. Weather	20
8.3	Application de CobWeb et la BD. Weather	22
9	Travail C	25
9.1	Clustering sur la BD. Bank-data	25
9.1.1	Application de la méthode K-means à Bank-data	25
9.1.2	Application de la méthode EM à Bank-data	28
10	Travail D	30
10.1	Clustering K-means sur la BD iris	30
10.2	Clustering EM sur la BD iris	30
10.3	Test avec BD. séparée	31
11	Consignes sur le Travail à rendre	31
12	Compléments sur les méthodes du Clustering sous Weka	32
12.1	SimpleKMeans	32
12.1.1	Phase d'évaluation en SimpleKmeans	33
12.2	Farthest First	34
12.3	EM	35
12.3.1	Exemple d'application de la méthode EM	36
12.4	Remarque	37
12.4.1	A propos de la Log-vraisemblance	37
12.5	Filtres Weka , clustering et EM	37
12.5.1	Un autre filtre pour obtenir des probabilités	38
12.5.2	Similarités entre K-means et EM	38
12.6	Cobweb	39
12.7	DBSCAN	41
12.8	A propos de la méthode OPTICS	42
12.9	A propos de la méthode CLOPE	42

13 Les Bases de données (pour ce BE)	43
13.1 Exemple météo (ou Golf)	43
13.2 Exemple Banque	43

☞ **Suivant les versions de Weka**, certaines copies d'écrans ne sont peut être pas tout à fait au top !

Déroulement de la séance :

- Lire et pratiquer la première partie (K-means). Cela vous permettra de vous familiariser avec une méthode de clustering (de base) sous Weka.
- Ensuite, la section 6 page 13 abordent le travail propre à ce 3e BE.
- Le reste de ce document contient des détails sur les méthodes de clustering de Weka. Il pourra vous être utile pour y trouver des réponses à vos questions éventuelles.

Consignes sur le Travail à rendre :

Certaines BDs nécessitent une conversion du format .csv au format .arff

→ Voir passgae .csv à .arff en BE1 et également dans ce document. Vous pouvez également charger un fichier .csv et le sauvegarder au format .arff.

Consignes :

Il vous est demandé un compte-rendu détaillé de votre méthodologie ainsi que de vos résultats.

Vous réaliserez un rapport d'une quinzaine de pages environ qui détaillera :

- votre démarche générale,
- votre compréhension des données (informations, types, quantité...)
- les informations que vous pensez pouvoir en tirer,
- les différentes méthodes de fouille utilisées : pour être crédible, il est nécessaire d'appliquer *au moins deux méthodes* aux données.
- les résultats obtenus pour chacune d'elle,
- une synthèse de ces résultats, ainsi que leur interprétation.
- Une **comparaison (statistique ?) des résultats** (cf. BE1, BE2).

Vous devriez maintenant être capable de retenir des critères de comparaison.

1 Classification Non supervisée (Clustering)

Soit une base de données $BD = (X, y)$ où pour chaque instance (X_i, y_i) , y_i représente la *classe* de X_i .¹

Par exemple, pour le service des prêts d'une banque, la BD peut contenir les X_i = les dossiers de prêt contenant diverses infos (age, revenus, situation fam., etc.) et y_i la **décision** d'accorder ou non le prêt.

Dans un processus d'apprentissage, une partie de la BD $((X_1, y_1), \dots, (X_m, y_m) \subseteq (X, y))$ sert à apprendre le modèle et le reste $((X_{m+1}, y_{m+1}), \dots, (X_n, y_n) \subset (X, y))$ pour tester/évaluer la qualité du modèle appris en confrontant pour chaque instance (X_i, y_i) , la sortie du modèle (la prédiction \hat{y}_i) avec la sortie connue y_i et en minimisant l'écart cumulé sur l'ensemble de la BD.

On dira que tout processus d'apprentissage apprend une fonction de prédiction $g(\cdot)$.

Dans le cas d'un apprentissage supervisé (BE1 et BE2), la fonction $g(X_i) = \hat{y}_i$ et le modèle appris tente de prédire la décision \hat{y}_i que l'on voudrait la plus proche de "la véritable" y_i .

Cas d'apprentissage non supervisé (ce BE) : le **Clustering** (ou Segmentation) est une méthode *non supervisée* dans le sens où aucun des attributs ne joue a priori un rôle particulier. On se contente de "grouper" les données (tout $X_i \cup \{y_i\}$) dans des groupes qui se *ressemblent* (notion de *similarité*) en **maximisant la similitude** entre les instances regroupées dans un même *cluster* (segment, grappe, groupe homogène, ...) et en **maximisant la distance** entre deux clusters séparés. Ici, la fonction apprise $g(X_i \cup \{y_i\})$ associe un cluster k_j à chaque instance $X_i \cup \{y_i\}$ de la BD : $k_j = g(X_i \cup \{y_i\})$, $j \in \{1, \dots, k\}$. k représente le nombre de clusters ($k = nb_clusters$).

² ← Lire cette note du bas de page.

Dans l'exemple de prêt bancaire, on cherchera à assembler les dossiers similaires (sans que y_i joue un rôle particulier).

On dira qu'un **bon clustering** **maximise** la cohésion intra-cluster (les clusters les plus compactes en leur sein) ainsi que la dispersion inter-cluster (les clusters les plus "éloignés" les uns des autres).

Cependant, **il n'est pas rare que pour X_i , la valeur y_i soit un marqueur de groupe.**

Dans l'exemple du banquier ci-dessus, on peut regrouper les dossiers en deux *clusters* et ensuite interpréter les deux groupes obtenus comme *{bon-payeurs, mauvais-payeurs}* en exploitant dans les dossiers des clients par exemple la régularité des remboursements.

Si on regroupait les dossiers en **excluant** y_i (ils ne participent pas à l'apprentissage de la fonction $g(\cdot)$), les y_i pourront être utilisés *a posteriori* pour savoir si les clusters produits correspondent (ou non) la valeur de y_i : la valeur des y_i d'un même cluster sont identiques et les y_i du premier cluster sont différents des y_i du second cluster.

Ainsi, si les deux groupes interprétés comme *{bon-payeurs, mauvais-payeurs}* sont mis en relation avec les y_i (*"prêt accordé", "prêt non accordé"*), cela pourra être considéré comme un indicateur de la qualité du clustering.

Si les y_i sont **exclus** de la tâche du clustering, on peut les utiliser *a posteriori* pour constater si les y_i correspondent aux deux clusters obtenus et ainsi disposer d'un indicateur supplémentaire de la qualité du clustering (comme dans une tâche de classification). **Weka permet de procéder au clustering de cette manière.**

2 Clustering sous Weka

2.1 Pratique de la méthode K-means et la BD météo

Commencer par charger le fichier de données *weather-numeric.arff* (données Météo/Golf).

☞ Cette BD contient deux attributs numériques, à ne pas confondre avec *weather.nominal.arff*.

Le contenu de ce fichier (en français) est rappelé en annexes (section 13.1, page 43).

On peut trouver les détails du chargement d'un fichier de données sous Weka

- Dans le fichier (fourni avec BE1) : "Data-Preparation-WEKA.pdf".

1. X_i est un vecteur de taille $p \geq 1$ et en général, on suppose par défaut que y_i est de taille une.

2. On voit dans la littérature : pour k clusters, au lieu d'associer un (numéro de) cluster $k_j = g(X_i \cup \{y_i\})$, $j \in \{1, \dots, k\}$, on envisage que chaque numéro de cluster soit codé par un vecteur de bits $\{0, 1\}^k$ où le cluster numéro j aura un 1 à l'indice j de ce vecteur. Pour une instance $E_i = (X_i \cup \{y_i\})$ de la BD et k clusters où $encoder(E_i) = k_j$ ($encoder(E_i)$ désigne le cluster de E_i), on peut écrire : $encoder(E_i) \in \{0, 1\}^k$ et $decoder(\{0, 1\}^k)$ désigne le centroïde du cluster k_j représenté par ce vecteur de bits. L'objectif de l'**auto-encodeur** idéal est donné par **$decoder(encoder(E_i))$ = le centroïde plus proche de E_i** (à défaut de $decoder(encoder(E_i)) = E_i$). Voir cours 7 pour $encoder(\cdot)/decoder(\cdot)$.

Le format d'un fichier de données Weka (format arff) est également rappelé dans BE1. Rappelons que Weka sait également charger un fichier au format csv.

Les algorithmes de clustering (ou segmentation) regroupent les instances en clusters (segments) selon leur similarité. Dans Weka, les algorithmes de clustering sont accessibles par l'onglet **Cluster**.

La figure ci-dessous montre l'exécution d'un exemple de clustering sous Weka appliquée au fichier *weather-numeric.arff* (méthode *K-means* vue en Cours).

Les options de la méthode *K-means* employée sont celles proposées par défaut.

Dans la figure ci-contre, on peut noter (en haut, sous les onglets) :

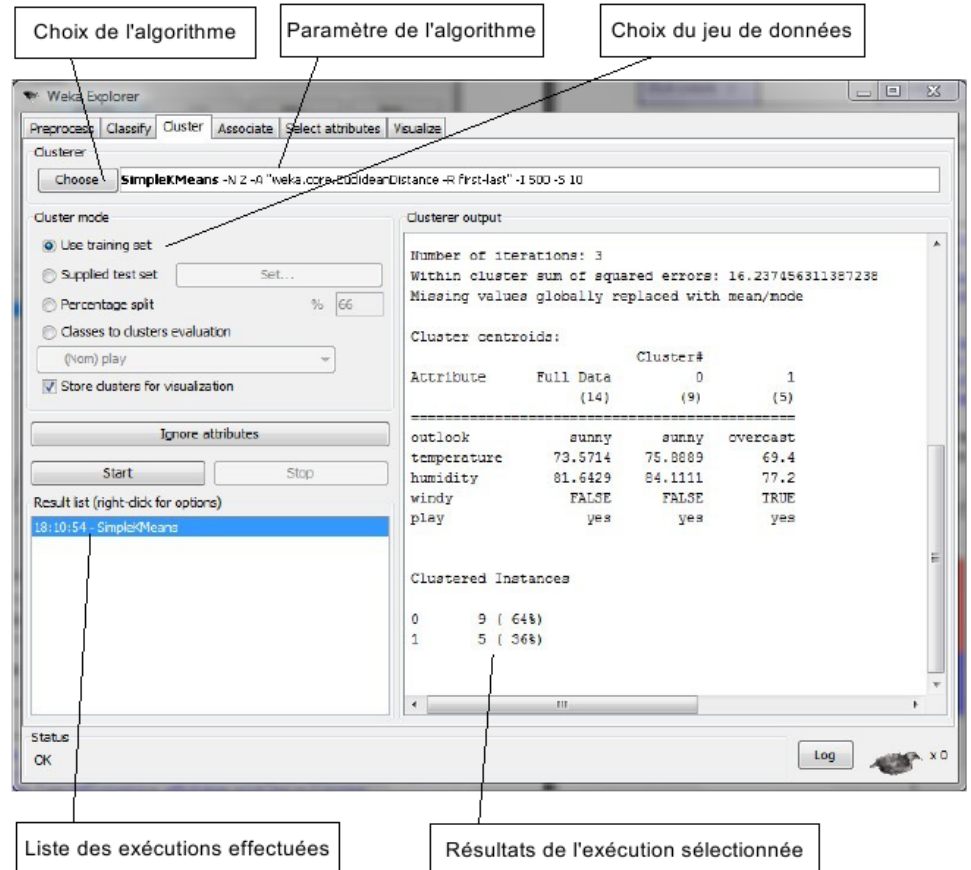
- Le cadre *Clusterer : Choose* permet de choisir l'algorithme du Clustering.

Un clic sur *Choose* permet de faire cette sélection dans une liste (liste dynamique en fonction du contenu du fichier de données chargé).

Une fois la méthode choisie (ici *SimpleKmeans* avec ses paramètres par défaut affichés en face de *Choose*), on peut cliquer sur les paramètres pour les modifier.

- La zone *Cluster mode* permet de choisir le jeu de données traité et les options d'évaluation des clusters.

Il n'y a plus de zone d'évaluation (cf. **classification** vue en BE1 et BE2).



Les options du clustering :

Dans le cadre **cluster Mode** (à gauche) :

- *Use training set* : le clustering est exécuté sur tout le jeu de données (risque de *over-fitting*).
- *Supplied test set* : choisir un autre jeu de données (cliquez sur le bouton Set) pour tester le modèle produit.
- *Percentage split* : le pourcentage est la partie des données utilisée pour définir un modèle des clusters (ensemble d'apprentissage). Le reste est utilisé comme jeu de test du modèle (dit ensemble de test).

Important : la méthode de validation 10-XV (que vous connaissez) a disparue des choix ! Elle ne s'applique pas à cette méthode mais certaines méthodes de Clustering comme EM peuvent encore s'en servir.

Notez également que "use training set" est devenu le choix par défaut.³

Par rapport aux BEs précédents, un nouveau choix a fait son apparition : *Classes to clusters evaluation*.

Classes to clusters evaluation : un attribut (par défaut la classe y_i) est choisi pour comparer les clusters obtenus et les valeurs (classes) définies par y_i . L'évaluation est faite en affichant la matrice de confusion correspondant.

→ Notez que ceci est un choix d'évaluation. Par conséquent, on ne peut pas le sélectionner en même temps qu'un autre choix d'évaluation.

Pourtant, un tel choix ne serait pas aberrant : par exemple en même temps que "supplied test set" !.

- Les deux autres options (qui suivent) :

3. Selon le principe Essai-Erreur (cf. X-V), pour atteindre le nombre optimal k de clusters, (en particulier si un k_{min} et un k_{max} sont précisés en paramètre), Weka commence avec k_{min} et en le variant, cherche à atteindre la $k_{optimal} \leq k_{max}$. Néanmoins, certaines méthodes (cf. EM, voir plus loin) utilisent XV pour obtenir le meilleur k (nombre de clusters).

- *Store clusters for vizualisation* : les clusters sont mémorisés pour visualisation.
- Le bouton *Ignore attributes* : on peut ignorer certains (un à plusieurs) attributs pendant le clustering.
- Rappel : le cadre *Result list* (à gauche) contient la liste des exécutions effectuées et permet de choisir les résultats d'une exécution dans le cadre *Clusterer output* (à droite).
- Rappelons que nous utilisons l'exemple *weather-numeric.arff* de **format mixte** contenant deux attributs numériques (*Humidity* et *Temperature*) (section 13.1, page 43).

La figure suivante montre les résultats détaillés (sans "Debug" ni "DisplayStdDev") de l'application de *K-means* à la même BD avec quelques commentaires sur les résultats.

Activez l'option "**Classes to Cluster evaluation**" (et notez que la "class" est "play").

Détails des résultats :

ci-contre, on a les résultats de l'application de l'algorithme de clustering *SimpleKMeans* au fichier *weather-numeric.arff*.

🔊 Les informations de cette zone sont importantes !

- Les **Centroïdes** des clusters obtenus et celui de toute la BD sont présentés.

Rappel du cours :

centroïde = une sorte de moyenne du cluster.

Le centroïde d'un cluster ne correspond pas forcément à une instance.

- Remarquez également :
 - K-means a choisi les centres initiaux **random**,
 - les statistiques de chaque cluster,
 - l'erreur moindre carrée (**l'erreur SSE**) : ("Within cluster sum of squared errors") s'affiche en haut de la fenêtre des résultats visible dans la figure ci-contre après la ligne "Number of iterations : 3".

- On peut également voir le traitement prévu pour les **valeurs manquantes**.

🔊 Notez bien que l'attribut "**play**" est absent des centroïdes (car on a activé "Classes to cluster evaluation"). Grâce à ce choix, nous disposons de la **matrice de confusion**.

Voir plus loin la section 2.3 en page 7 à propos de "Classes to cluster evaluation".

- Notez bien en bas de cette figure les 3 dernières lignes (cluster 0 correspond à "play=yes", cluster 1 à "no"). Sous cette hypothèse, le nombre d'instances mal classées = 6 (équivalent à une erreur de 42.8%)

```
Clusterer output
=====
Number of iterations: 3
Within cluster sum of squared errors: 11.237456311387234

Initial starting points (random):
Cluster 0: rainy,75,80,FALSE
Cluster 1: overcast,64,65,TRUE

Missing values globally replaced with mean/mode

Final cluster centroids:
Attribute      Full Data      Cluster#
              (14.0)      0          1
              (9.0)      (5.0)
=====
outlook        sunny        sunny        overcast
temperature    73.5714      75.8889      69.4
humidity       81.6429      84.1111      77.2
windy          FALSE        FALSE        TRUE

Time taken to build model (full training data) : 0 seconds

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

Clustered Instances
0      9 ( 64%)
1      5 ( 36%)

Class attribute: play
Classes to Clusters:
0 1 <-- assigned to cluster
6 3 | yes
3 2 | no

Cluster 0 <- yes
Cluster 1 <- no

Incorrectly clustered instances : 6.0      42.8571
```

Evaluation : l'erreur SSE de K-means est calculée de la manière suivante (différence de vecteurs) : pour toute instance d'un cluster, la somme de l'erreur au carrée entre chaque attribut de l'instance et l'attribut correspondant du centroïde est obtenue. La somme de ces erreurs donnera l'erreur pour cette instance. La somme de toutes les erreurs des instances constitue l'erreur SSE affichée (ici, cela donne 11.237).

Application avec une méthode d'évaluation différente :

Cette fois, on désactive "classes to clusters evaluation" et on choisit "percentage split" à 66%.

Détails des résultats : on remarque que

- **K-means est appliquée une première fois à toute la BD** mais ses résultats sont absents de la figure ci-contre (par manque de place!). Vous pouvez cependant les constater sur vos machines, au dessus de la fenêtre ci-contre.

- Ensuite, K-means est appliquée à 66% de la BD. (9 instances) pour apprendre les clusters,

- Puis ces résultats sont testés sur le 33% de la BD. (5 instances, voir le tableau tout en bas de la figure).

☞ Notez bien que l'attribut "play" est cette fois présent dans les centroïdes.

→ Comparez l'erreur de ces résultats avec celle obtenue sur toute la BD. (la partie absente de la figure ci-contre) : elle s'est améliorée.

Quelques uns des algorithmes implantés sous Weka sont énumérés ci-dessous.

Lors du choix de la méthode, Weka adapte la liste des méthodes au contenu des données selon les types des attributs : nominal, numérique, mixte,

Voir aussi (en section 12 en page 32) les détails de certaines méthodes ; voir aussi le dernier cours.

```
kMeans
=====

Number of iterations: 2
Within cluster sum of squared errors: 7.9443946412391675

Initial starting points (random):

Cluster 0: rainy,65,70,TRUE,no
Cluster 1: overcast,81,75,FALSE,yes

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute      Full Data      Cluster#
                (9.0)         0           1
=====
outlook         rainy         rainy      overcast
temperature     73.7778       72         74.6667
humidity        80.3333      83.6667    78.6667
windy           TRUE         TRUE       FALSE
play            yes          no         yes

Time taken to build model (percentage split) : 0 seconds

Clustered Instances
0      1 ( 20%)
1      4 ( 80%)
```

Attention : les numéros des clusters sont ici en première colonne (confondable la première ligne).

Petit exercice : cliquez dans la zone de la méthode "SimpleKMeans" et dans la fenêtre des options qui s'affiche, activez "Debug" et "DisplayStdDev" (écart type) (**Ne pas choisir "fastDistanceCalc"** qui empêchera d'obtenir l'**erreur SSE** en haut de la fenêtre des résultats.

→ Constatez les informations supplémentaires affichées dans la Fenêtre des résultats. En particulier, les détails pour chaque attribut dans chacun des clusters (compte tenu d choix "Percentage split=66%).

2.2 Remarques

☞ On reprend les deux figures précédentes.

- Si on compare les caractéristiques des clusters dans ces résultats pour le cas "66% split" vs. le cas "Use Training Set", on constate que les clusters ne sont pas les mêmes.

- **Plus important :** on constate également que l'erreur pour le cas "66% split" est **supérieure** à l'erreur lors du clustering avec 100% de la BD. A priori, il devrait constater l'inverse (avec du sur-apprentissage pour le cas 100% de la BD.).

En fait, les deux erreurs **ne sont pas calculées** de la même façon : dans le cas du "66% split", l'erreur est calculée sur 33% des données du test tandis que pour l'autre cas, elle est calculée sur toute la BD. Il est donc normale que l'erreur soit plus élevée car elle représente une somme des carrées des distances et le cas "66% split" ne somme que sur environ 5 instances de test.

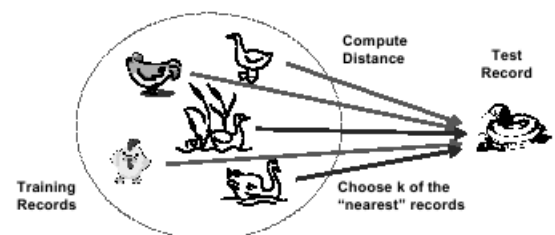
☞ Constatez également la différence entre les deux SSE (erreurs) calculées pour les deux cas.

- Voir aussi plus loin la section 7.4 en page 18.

Suite à un clustering, une information recherchée est de connaître le cluster des instances.

Plus encore, c'est le cluster de **toute nouvelle instance** (= la prédiction) qui peut nous intéresser.

Pour cela, il nous faut préparer un fichier de test et choisir l'option "Supplied test set". Voir plus loin.



Les exercices de ce BE font souvent référence à quelques éléments importants **signalés** ci-après.

Prenez le temps d'y prêter attention sur cette BD simple, ils vous serviront ensuite dans les exercices prévus dans ce BE.

2.3 A propos des paramètres de K-means

Visualisation des résultats :

On revient sur la méthode *K-means* de l'exemple précédent.

- Notez (veiller à) l'option "*store cluster for visualisation*" dans la zone "*Cluster mode*",
- Cliquer droit sur "*SimpleKmeans*" dans le cadre "*Result List (right clic for options)*" puis "Visualize Cluster Assignments" ouvre une fenêtre de visualisation.
- Fixer X à *cluster* et l'axe Y à *instance_number* pour visualiser les distributions des instances dans les clusters (≥ 2). Placer le curseur "Jitter" à droite pour agrandir.
- D'autres combinaisons de visualisation (p.ex. cluster vs. l'attribut "Windy") permettent de repérer les attributs qui discriminent (plus ou moins) les clusters.

Recommencer mais cette fois avec l'axe X pour l'attribut "play" et l'axe Y pour les "clusters". Vous devriez repérer l'erreur (une croix bleu) : une instance avec *play=yes* est placée dans le cluster 1

Option "Classes to clusters evaluation" :

- Lorsque qu'on active l'option *Classes to clusters evaluation*, quelques différences apparaissent dans les résultats. On remarque cette fois qu'une **matrice de confusion** et un taux d'erreur peuvent être calculés.

Dans ce mode, **Weka ignore d'abord les classes et procède à un clustering**. Puis pendant le test, il attribue les classes aux clusters en se basant sur les valeurs des attributs, les classes et les clusters. Ensuite, l'erreur du clustering est calculée en fonction cette affectation et une matrice de confusion correspondante est présentée.

☞ Ayant choisi cette méthode d'évaluation ("Classes to clusters evaluation"), on ne peut naturellement plus fournir un fichier de test, ni d'autres méthodes de test et d'évaluation.

Options de la méthode (clique dans la zone "SimpleKmeans") :

- L'option "**Debug**" dans les paramètres ("Debug=True") donne des informations supplémentaires. Si vous lancez Weka en console (cf. MacOS ou Linux), vous pouvez y voir quelques autres informations de débogage ("debug").
- Demandez l'option "**displayStdDevs**" pour avoir des statistiques sur les attributs. Vous obtenez quelques résultats supplémentaires dans la zone des résultats à droite.
- L'option "**numExecutionSlots**" permettant d'utiliser plusieurs *threads* (calcul parallèle) pour les calculs.

Rappelons que dans la méthode de clustering **SimpleKMeans**, on a une approche par centroïde. Chaque cluster final est décrit par son centre de gravité (qui n'est pas forcément une instance de la BD.).

Dans les clusters obtenus, Weka affiche les moyennes des valeurs des attributs numériques (ou les valeurs caractéristique/majoritaire des attributs catégoriels) de chaque cluster.

N.B. : au besoin, on peut réclamer plus de mémoire pour Weka (si grosses BDs. ou méthodes compliquées)

☞ Si vous lancez weka dans une fenetre de commande, en cas de problèmes de mémoire, lancer Weka plutôt avec l'option qui fixe une taille de mémoire :

`java -Xmx2048m -jar weka.jar`

→ On demande 2048m (2048 mega octets=2GB). Fixer cette valeur par exemple à 256MB.

Pour en savoir plus sur les méthodes de clustering sous Weka il est conseillé de lire la section 12 page 32 pour en savoir davantage.

2.3.1 Initialisation des centroïdes dans K-means

On détaille ci-après le choix des centroïdes initiaux de la méthode *SimpleKMeans* de Weka avant d'aborder d'autres méthodes de clustering de Weka.

Dans *SimpleKmeans*, cliquer dans la ligne de commande pour obtenir différentes stratégies de sélection des clusters initiaux. La stratégie de sélection par défaut est un choix **aléatoire** des centroïdes.

☞ Certaines de ces stratégies sont au coeur de méthodes de clustering indépendantes (de K-means).

La stratégie de sélection des centres peut varier dans SimpleKmeans :

- *Random* : aléatoire pure.
- *FarthestFirst* : choix des médoïdes initiaux les plus éloignés.

Les résultats affichés correspondent cependant aux centroïdes (à ne pas confondre avec la méthode *Kmedoïde*). Voir la section 12.2, page 34 pour les détails.

- *K-means++* : similaire à Farthest-First. Limite le risque du choix des outliers comme médoïdes initiaux (voir ci-dessous).
- *Canopy* : initialisation des clusters initiaux (voir ci-dessous).

• **Stratégie d'initialisation Farthest First** dans *Simplekmeans* :

Dans cette stratégie d'initialisation (voir cours chapitre 5), le premier médoïde est choisi aléatoirement⁴. Le médoïde suivant est choisi le plus éloigné du premier, ... Le i ème médoïde ($i \in 3..k$ pour k clusters) est le plus éloigné de l'ensemble des $(1..i-1)$ médoïdes précédents. Notons que ces choix peuvent favoriser des outliers (qui pourraient être des anomalies ou pas !).

• **Stratégie d'initialisation K-means++** dans *Simplekmeans*

C'est une stratégie d'initialisation des médoïdes initiaux (l'élément central qui correspond à une instance existante vs. un centroïde = une moyenne qui ne correspond pas forcément à une instance).

Appelée "K-means++" (paramètre à fixer dans la ligne de commande de la méthode), elle donne de meilleurs résultats (que la méthode aléatoire de choix initiaux, voire même mieux que la stratégie *farthest-First*) en terme de la somme des carrés des erreurs (erreur SSE) ainsi que d'autres mesures d'évaluation.

Cette méthode d'initialisation (voir les détails immédiatement après) choisit des médoïdes éloignés les uns des autres tout en évitant de favoriser les outliers (un défaut de la méthode d'initialisation "farthest first").

Détails de la méthode K-means++ :

Soit K clusters et un ensemble d'instances X de taille $\geq K$. K-means++ choisit d'abord le premier centroïde c_1 aléatoirement puis les autres médoïdes $x = c_k \in X, k = 2..K$ sont choisis comme suit :

Choisir le prochain $c_k = x$ avec une probabilité proportionnelle à $D(x)^2$ où $D(x)$ est le minimum de distance à un médoïde existant : $D(x) = \min_{k' < K} \text{dist}(x, c_{k'})$. Les distances sont normalisées par $\frac{D(x)^2}{\sum_{x \neq c_k} D(x)^2}$.

Ainsi, les médoïdes sont les plus éloignés les uns des autres. Les outliers ont par définition une valeur élevée de $D(x)$ mais sachant qu'ils ne sont pas proches d'autres points, un cluster avec un outlier seul est possible (repris en post-traitement).

La stratégie de K-means++ est plus rapide (stabilisation plus rapide) et utilise une matrice des distances. Elle est assez proche de la stratégie "farthest first" ci-dessous.

• **Stratégie d'initialisation Canopy** dans *Simplekmeans* :

on choisit une première instance pour constituer un nouveau *canopy* (*canopée*, *baldaquin*, cluster primaire) et l'instance en question est retirée de l'ensemble d'apprentissage.

Pour les autres, si une instance est à une distance inférieure à un certain seuil s_1 , on l'intègre dans le baldaquin. Si en plus cette instance est à une distance inférieure à un certain seuil (plus sévère) s_2 , on la supprime de l'ensemble d'apprentissage (ne participe plus aux autres tests). Cette séquence est répétée créant différents clusters primaires.

L'intérêt de cette initialisation est de réduire le nombre de comparaisons entre les instances. La pratique montre que les clusters initiaux ainsi créés permettent un meilleur résultat pour *K-means*.

4. Notons que ce premier médoïde peut être choisi comme une "moyenne" / "mode" / "médiane" des attributs.

3 Aperçu de quelques méthodes de Clustering sous Weka

☞ Ces méthodes sont détaillées en section 12 page 32.

1. Méthode de clustering **K-means** et ses stratégies de choix initiaux (voir section 12.1, page 32).

2. Méthode de clustering **Farthest First** (voir section 12.2, page 34) :

Outre la stratégie du même nom dans SimpleKmeans, cette méthode est une méthode de clustering qui choisit ses médoïdes les plus éloignés les uns des autres.

☞ Dans sa version originale (Hochbaum and Shmoys 1985), Farthest-First diffère de la méthode K-means en deux points :

- Une fois les médoïdes choisis, l'algorithme Farthest-First **termine en une seule itération** et affecte les autres instances aux centres les plus proches.
- Elle **ne recalcule pas de centroïde** et les clusters obtenus ont pour centre les médoïdes de départ.

3. Méthode de clustering **EM** (*Expectation Maximization*, voir section 12.3, page 35, et 12.5, page 37) :

génère une description probabiliste des clusters en terme de moyenne et écart-type pour les attributs numériques et en terme de nombre et pourcentage pour les attributs nominaux.

Chaque cluster est décrit par sa probabilité a priori et une distribution de probabilité pour chaque attribut. L'affichage des résultats comprend le nombre d'exemples pour un attribut nominal et les caractéristiques de la distribution normale pour un attribut numérique.

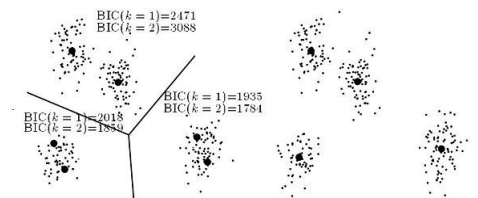
4. Méthode de clustering **Cobweb** : clustering hiérarchique par division ascendant à base de la mesure *Category utility*. Le résultat est un arbre hiérarchique probabiliste des divisions effectuées (l'arbre qui peut être visualisé graphiquement). voir section 12.6, page 39.

5. Méthode de clustering **DBScan** : approches basées sur la *densité* (voir ci-dessous, et aussi le support du cours pour *DBScan*). voir également la section 12.7, page 41.

6. La méthode "**X-means**" traite des **BDs numériques**. Elle exécute puis améliore (l'erreur SSE de) K-means.

Lorsque les clusters de K-means sont établis, on essaie de diviser chaque cluster en deux en se basant sur le critère BIC (*Bayesian Information Criterion*, voir ci-dessous) calculé pour le cluster qui a été divisé vs. les deux sous-clusters obtenus.

Si BIC est supérieur pour les deux sous-clusters, on retient la division (qui augmente le nombre de clusters k).



L'expérience montre que le nombre de clusters obtenus est très proche du k initial mais avec une erreur SSE moindre.

Dans la figure ci-dessus, X-means a produit 4 clusters (à droite) à partir des 3 clusters (de gauche) obtenus par K-means. Si $BIC(k=2)$ est plus élevée, X-means retient les deux sous-clusters ; dans le cas contraire, on conserve le cluster initial.

$BIC(k) = \ln(n).k - 2\ln(\ell)$ où n est le nombre d'instance dans le/les k clusters (1 ou 2),

$\ell = Pr(x|\theta)$ est la vraisemblance de la BD (Voir cours pour le calcul de la vraisemblance) et θ les paramètres du clustering obtenu.

☞ Une des utilisations pratiques de *X-means* est de fournir le nombre (optimal) de cluster k qui peut servir à d'autres méthodes.

7. Méthode de clustering **OPTICS** (voir section 12.8, page 42) : une autre approche basée sur la densité.

Méfiez-vous de la méthode OPTICS qui a souvent tendance à bugger et nous obliger à redémarrer Weka (en particulier si l'option "ShowGUI" =True). Utiliser plutôt DBScan !

8. **CLOPE** : méthode par construction de modèle basée sur le comptage des co-occurrences de valeurs des attributs dans les clusters.

9. La méthode "**Cascade simple k-means**" établit k clusters selon un certain critère de choix de k , (cf. *calinski-harabasz*). L'utilisateur donnera un min et un max pour k .

10. La méta-méthode "**MakeDensityBasedClusterer**" où on peut préciser la méthode de clustering qui devrait être utilisée (par défaut *SimpleKmeans*). Voir l'encadré sur EM ci-après.

Voir aussi la section 12 page 32 pour plus de détails sur les méthode de clustering sous Weka.

Remarques plus générale sur le clustering sous Weka :

Sous Weka, les paramètres des méthodes varient selon l'algorithme utilisé.

Les paramètres les plus communs en clustering sont :

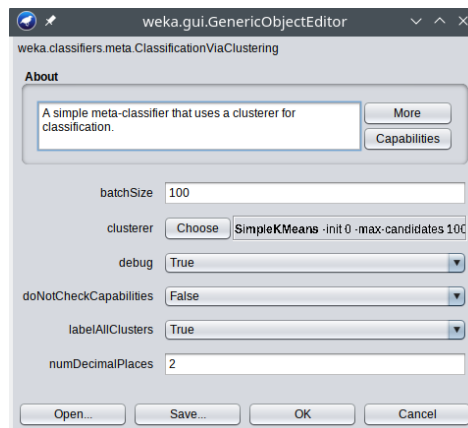
- *numClusters* : nombre de clusters à former.
- *seed* : pour modifier la séquence de tirages aléatoires utilisée par certains algorithmes.
- *MaxIterations* : nombre maximal d'itérations à exécuter.
- *DistanceFunction* : fonction de distance pour le calcul de similarité.

4 Classification et Clustering

Il est possible, sous Weka, d'utiliser une méthode de clustering pour procéder à une classification.

Dans l'onglet "classify, choisir "meta" puis "classification via clustering" (il faudra installer le package "classification via clustering" puis relancer Weka et rechargez le même fichier "weather-numeric.arff").

La fenêtre correspondant à "classification via clustering" est donné ci-dessous.



Par défaut, l'algorithme de clustering proposé est *Kmeans*. Vous pouvez modifier la méthode de clustering choisie en cliquant dans "Choose".

Dans cette fenêtre, mettre "labelAllClusters" à Vrai.

Si vous cliquez en face de "chosse" sur "SimpleKMeans....", vous aurez accès aux paramètre de K-mens (comme ci-dessus dans ce document).

Dans les paramètres de KMeans, mettez "Debug" à Vrai et mettre "DisplayStdDevs" à Vrai.

Procédé comme dans BE1 : "More Options" puis demander "output prediction" dans "PlainTex"....

Fermer ces fenêtre et cliquer sur "start" pour voir les résultats.

Vous aurez ainsi procédé à une "classification" à l'aide du Clustering à l'aide de KMeans.

Vous constaterez que KMeans a exclu la classe "play" du clustering.

Vous avez dans le fenêtre des résultats des informations concernant le clustering mais aussi celles concernant la classification. Vous avez les centroïdes, l'erreur SSE du clustering,

Plus bas dans la fenêtre des résultats, vous constaterez par exemple un taux de justesse de 64.2%

Répétez l'expérience avec une autre méthode de clustering.

Choisir par exemple la méthode EM (à la place de Kmeans, voir la section 12.3.1, page 36). Relancez et comparer les deux résultats.

5 Les résultats du Clustering

Une fois le clustering effectué, on peut par un clique droit, visualiser les résultats ("visualize cluster assignments"); choisir pour l'axe X les instances et pour Y les clusters. Ensuite, dans cette même fenêtre ("visualize cluster assignments"), sauvegarder les résultats dans un fichier au format arff. Le fichier résultat donnera le cluster de chaque instance.

Un exemple de résultat est donné ci-dessous. On note que les numéros des clusters sont ajoutés aux données de la BD "weather-numeri.arff".

```
@relation weather_clustered

@attribute Instance_number numeric
@attribute outlook sunny,overcast,rainy
@attribute temperature numeric
@attribute humidity numeric
@attribute windy TRUE,FALSE
@attribute play yes,no
@attribute Cluster cluster0,cluster1

@data
0,sunny,85,85,FALSE,no,cluster0
1,sunny,80,90,TRUE,no,cluster0
2,overcast,83,86,FALSE,yes,cluster0
3,rainy,70,96,FALSE,yes,cluster0
4,rainy,68,80,FALSE,yes,cluster0
5,rainy,65,70,TRUE,no,cluster1
6,overcast,64,65,TRUE,yes,cluster1
7,sunny,72,95,FALSE,no,cluster0
8,sunny,69,70,FALSE,yes,cluster0
9,rainy,75,80,FALSE,yes,cluster0
10,sunny,75,70,TRUE,yes,cluster1
11,overcast,72,90,TRUE,yes,cluster1
12,overcast,81,75,FALSE,yes,cluster0
13,rainy,71,91,TRUE,no,cluster1
```

Un autre exemple de résultats :

Les résultats de la même expérience avec la méthode de clustering Hiérarchique (*CobWeb*) est donné ci-dessous.

On note que les numéros des clusters ne sont pas contiguës : il s'agit d'un clustering Hiérarchique (*CobWeb* et *Agglomératif* détaillés plus loin) où un cluster peut encore être scindé en d'autres clusters. Les numéros qui restent sont ceux des clusters au dernier niveau du dendrogramme.

Le fichier et la structure ci-dessous correspondent aux résultats de l'application de la méthode *CobWeb* à l'exemple "météo" mais ce n'est qu'une méthode parmi d'autres..

```
@relation weather_clustered

@attribute Instance_number numeric
@attribute outlook sunny,overcast,rainy
@attribute temperature numeric
@attribute humidity numeric
@attribute windy TRUE,FALSE
@attribute play yes,no
@attribute Cluster cluster0,cluster1,cluster2,cluster3,cluster4,cluster5

@data
0,sunny,85,85,FALSE,no,cluster3
1,sunny,80,90,TRUE,no,cluster5
2,overcast,83,86,FALSE,yes,cluster2
3,rainy,70,96,FALSE,yes,cluster4
...
11,overcast,72,90,TRUE,yes,cluster5
12,overcast,81,75,FALSE,yes,cluster2
13,rainy,71,91,TRUE,no,cluster5
```

Ce qui correspond à l'arbre dessiné dans les résultats CobWeb (visualisez l'arbre sous Weka).

☞ L'encadré ci-dessus est obtenu avec un clique droit sur la méthode choisi (*CobWeb*) puis "Visualisation Cluster Assignements" puis "save" dans un fichier arff.

5.1 Utilisation de SciKitLearn sous Weka

On peut choisir la méthode du clustering "ScikitLearnCluster" (package Python scikitlean pour weka à installer).

Par exemple, pour appliquer K-means par cette voie :

- charger une BD. (météo p. ex.),
 - dans les paramètres de "ScikitLearnCluster", choisir K-means (par défaut)
 - ajouter dans la zone suivante ("LearnerParameters", c-à-d. pour K-means)
"n_clusters=2, init='random', verbose=1"
 - exécuter et observer ses résultats.
- ➔ Dans le cas présent, cela revient à appliquer K-means ; il ne représente donc pas d'intérêt particulier si ce n'est de signaler / montrer la disponibilité de la librairie "ScikitLearnCluster" (et "ScikitLearn" d'une manière générale).

Aller plus loin dans l'expérience d'utiliser ScikitLearn sous Weka. Tester d'autres méthodes,

6 Travail en séance et à rendre

Le travail à réaliser est réparti en 4 parties : A, B, C et D. Il vous permet de vous familiariser avec quelques méthodes (les plus utilisées) de clustering. L'ensemble est noté sur 24.

Informations :

- voir la section 12.5 en page 37 pour voir comment affecter des clusters aux instances.
- Pour visualiser les erreurs (et les instances concernées) dans un clustering, n'oubliez pas d'utiliser "Visualize Cluster Assignment" (clique droit sur une méthode), placer "color" = la classe, axes X="Instance Number" et Y="Cluster"; augmenter "Jitter" et repérer les instances "mal placées" (désaccord classe / cluster).

7 Travail A

En plus de la section 3 (page 9), il est conseillé de lire la section 12 (page 32) pour en savoir plus sur les méthode de clustering sous Weka.

7.1 K-means et Farthest First sur la BD. Weather

Consignez les résultats du clustering sur l'exemple *Météo* (*weather-nominal.arff*). Voir autre consignes en section 11 en page 31. Au besoin, agrandissez les encadrés des réponses avec Excel/Word (ou équivalents) mais rendez un .pdf !

L'objectif est de regrouper les instances du jeu de données en clusters correspondant chacun soit à *Play=yes* soit à *Play=no*. C'est à dire que l'on souhaite que les instances d'un cluster soient majoritairement associées à *Play=yes* ou bien à *Play=no*.

- **EXERCICE I** : Appliquez l'algorithme *FarthestFirst* au fichier *weather-nominal.arff* en fixant le nombre de clusters à 2 par le paramètre *numClusters*. Cochez l'option "Use Training Set".

Indiquez dans le tableau ci-dessous les valeurs des attributs des centroïdes (médoïdes) par cette méthode pour chacun des deux clusters. Ces valeurs représentent les valeurs moyennes pour les instances du cluster.

Cluster	Outlook	Temperature	Humidity	Windy	Play
Cluster0 :	overcast	mild	high	TRUE	yes
Cluster1 :	sunny	hot	high	FALSE	no

- **EXERCICE II** : Appliquez l'algorithme *SimpleKMeans* en fixant le nombre de clusters à 2 et indiquez dans le tableau ci-dessous la description des centroïdes des deux clusters. Cochez l'option "Use Training Set".

Cluster	Outlook	Temperature	Humidity	Windy	Play
Cluster0 :	sunny	mild	high	FALSE	yes
Cluster1 :	overcast	cool	normal	TRUE	yes

- **EXERCICE III** : Comparez la description des centroïdes des clusters obtenus par les deux algorithmes. Indiquez ci-dessous le nom de l'algorithme qui génère le mieux (erreur minimale) un cluster correspondant à *Play=yes* et un autre à *Play=no*. Justifiez !

Le premier évidemment

Les deux algorithmes fournissent des résultats notablement différents. Nous allons dans la suite paramétrer l'exécution de *SimpleKMeans* afin d'obtenir 2 clusters correspondant le mieux à chaque valeur de l'attribut *Play*.

Cliquer avec le bouton droit dans le cadre *Result list* sur les 2 exécutions précédentes et supprimez-les en choisissant l'option *Delete result buffer*.

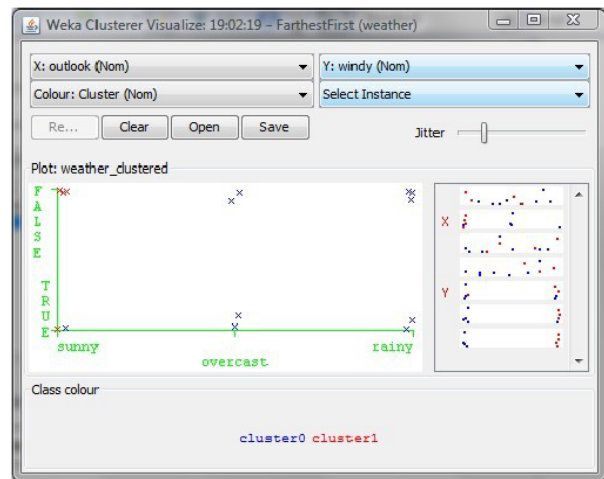
Affichage des clusters : Afin de mieux appréhender le contenu de chaque cluster, nous allons observer la répartition des instances des clusters selon les valeurs des attributs, c-à-d la composition des clusters.

● **EXERCICE IV** Exécutez l'algorithme *FarthestFirst* et *SimpleKMeans* en cochant l'option *Store clusters for visualization*.

- Cliquez avec le bouton droit dans le cadre *Result list* sur l'exécution de *FarthestFirst* et choisissez l'option *Vizualise cluster assignement*.

- Sélectionnez l'attribut *outlook* pour la dimension X et l'attribut *windy* pour la dimension Y.

- Sélectionnez l'attribut cluster pour la couleur dans la zone *Colour* afin que la couleur de chaque croix (instance) représente le cluster auquel elle appartient. Les croix bleues représentent les instances du cluster 0 et les croix rouges les instances du cluster 1.



Notez dans un cadre similaire à ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles toutes les instances appartiennent au cluster 0, c'est à dire pour lesquelles les croix sont toutes bleues.

→ Vous pouvez lire les valeurs sur le repère ou bien cliquer sur les croix pour voir les instances concernées.

Outlook :		Outlook :	
Windy :		Windy :	
Outlook :		Outlook :	
Windy :		Windy :	

Notez dans un cadre similaire à ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles toutes les instances appartiennent au cluster 1, c'est à dire pour lesquelles les croix sont toutes rouges.

Outlook :		Outlook :	
Windy :		Windy :	

● **EXERCICE V** : Exécutez l'algorithme *SimpleKMeans* en cochant l'option *Store clusters for visualization* et en donnant la valeur 20 au paramètre *seed* dans la fenêtre des paramètres qui s'affiche lorsque vous cliquez sur le nom de l'algorithme.

Indiquez dans le tableau ci-dessous les valeurs des attributs des centroïdes pour chacun des deux clusters obtenus.

Cluster	Outlook	Temperature	Humidity	Windy	Play
Cluster0 :					
Cluster1 :					

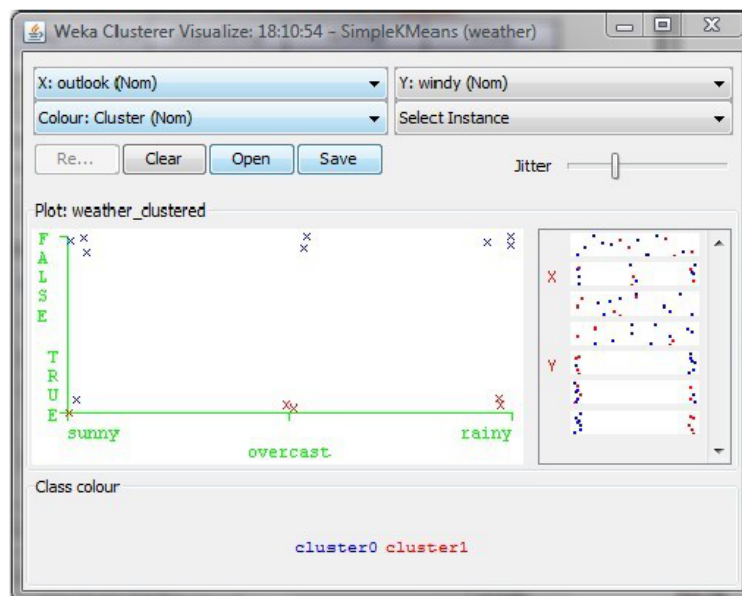
EXERCICE VI : Cliquez avec le bouton droit dans le cadre *Result list* sur l'exécution de *SimpleKMeans* et choisissez l'option *Vizualise cluster assignement* (le curseur "Jitter" = Loupe).

Notez dans le cadre ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles toutes les instances appartiennent au cluster 0.

Outlook :		Outlook :	
Windy :		Windy :	
Outlook :		Outlook :	
Windy :		Windy :	

Notez dans le cadre ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles toutes les instances appartiennent au cluster 1.

Outlook :		Outlook :	
Windy :		Windy :	



7.2 Comparaison de clusters

Nous voulons comparer les clusters obtenus avec les valeurs *Play=yes* et *Play=no*.

- **EXERCICE VII** : Cliquez à nouveau avec le bouton droit dans le cadre *Result list* sur l'exécution de *FarthestFirst* et choisissez l'option *Vizualise cluster assignement*. Dans la fenêtre qui s'affiche, sélectionnez l'attribut *play* pour la couleur dans la zone *Colour*.

Chaque couleur correspond maintenant à une classe : les croix bleues aux instances *play=yes* et les croix rouges aux instances *play=no*.

Affichez côte-à-côte (ou visualisez à tour de rôle) ce graphique et celui de l'affichage de l'attribut cluster en couleur pour l'exécution de *FarthestFirst*.

Notez dans le cadre ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles on peut observer des différences dans la couleur des croix entre les 2 graphiques. Ces différences sont les instances qui ont été placées dans un cluster alors qu'elle auraient du l'être dans l'autre pour augmenter la cohérence du clustering par rapport à l'attribut *Play*. **Si aucune différences n'apparaît, toutes les instances de chaque cluster correspondent à la même valeur de l'attribut *Play*, c-à-d que le clustering est parfait.**

Outlook :		Outlook :	
Windy :		Windy :	

- **EXERCICE VIII** : Cliquez avec le bouton droit dans le cadre *Result list* sur l'exécution de *SimpleKMeans* et choisissez l'option *Vizualise cluster assignement*.

Dans la fenêtre qui s'affiche, sélectionnez l'attribut cluster pour la couleur dans la zone *Colour* afin que chaque couleur corresponde à une classe (croix bleues *play=yes* et croix rouges **play=no**).

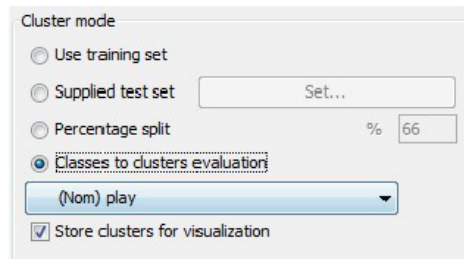
Affichez côte-à-côte ce graphique et celui de l'affichage des clusters en couleur pour l'exécution de *SimpleKMeans* et notez dans le cadre ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles on peut observer des différences dans la couleur des croix entre les 2 graphiques.

Outlook :		Outlook :	
Windy :		Windy :	
Outlook :		Outlook :	
Windy :		Windy :	

7.3 Évaluation par rapport à un attribut de classe

Il est possible d'évaluer automatiquement si chaque cluster correspond ou non à une classe distincte, c-à-d une valeur distincte d'un attribut de classe tel que *Play*.

Ceci se fait à l'aide de l'option *Classes to clusters evaluation* et en sélectionnant l'attribut de classe dans la zone (Nom) en dessous.



Lorsque cette option est choisie, à chaque cluster généré est associée une valeur de l'attribut de classe, par exemple un cluster pour Play=yes et un cluster pour Play=no.

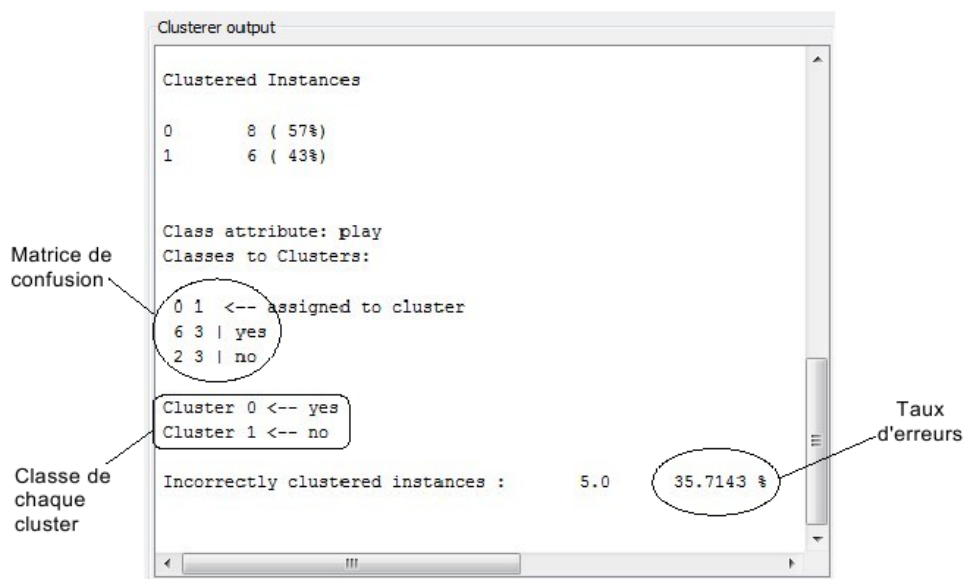
Dans ce mode, Weka ignore d'abord l'attribut *class* (décision) et calcule les clusters. Ensuite, pendant la phase de test, Weka attribue des classes aux clusters en fonction de la valeur majoritaire de l'attribut de classe dans chaque cluster. Puis Weka calcule l'erreur de cette classification et propose la matrice de confusion correspondante.

Dans la figure ci-dessous le Cluster 0 correspond à la valeur Play=yes pour et le cluster 1 à la valeur *Play=no*.

Si cette option est choisie, l'algorithme détermine alors pour chaque cluster le nombre d'instances bien et mal placées, c'est à dire le nombre de réussites et d'erreurs vis-à-vis de l'attribut de classe. Les 4 nombres calculés sont affichés dans la **matrice de confusion** :

- Vrais positifs : nombre d'instances yes placées dans le cluster correspondant à yes
- Faux positifs : nombre d'instances no placées dans le cluster correspondant à yes
- Vrais négatifs : nombre d'instances yes placées dans le cluster correspondant à no
- Faux négatifs : nombre d'instances no placées dans le cluster correspondant à yes

Dans cette figure, les lignes représentent les valeurs de Play (yes/no) et les colonnes les clusters (0 ou 1).



Dans cet exemple, nous avons :

- Vrais positifs : 6 instances Play=yes placées dans le cluster 0 (correspondant à Play=yes)
- Faux positifs : 2 instances Play=no placées dans le cluster 0
- Vrais négatifs : 3 instances Play=no placées dans le cluster 1 (correspondant à Play=no)
- Faux négatifs : 3 instances Play=yes placées dans le cluster 1

Le nombre total et le taux d'erreurs sont également affichés.

• **EXERCICE IX** Ré-exécutez les algorithmes *FarthestFirst* (paramètre numClusters=2) et *SimpleKMeans* (paramètres numClusters=2 et seed =20) en utilisant l'option *Classes to clusters evaluation* et en sélectionnant l'attribut play comme attribut de classe.

Remplissez le tableau ci-dessous.

Valeur / Méthode	<i>FarthestFirst</i>	<i>SimpleKMeans</i>
Nb. faux Positifs		
Nb. faux Négatifs		
Taux d'erreur		

• **EXERCICE X** Nous allons comparer les résultats obtenus avec différentes valeurs des paramètres des algorithmes afin d'optimiser leur valeur. Utilisez l'option "Classes to clusters evaluation" afin d'évaluer chaque exécution.

Appliquez les algorithmes *SimpleKMeans* et *FarthestFirst* en fixant le nombre de clusters à 2.

Réalisez pour chaque algorithme une expérimentation pour des valeurs du paramètre *seed* de 1, 10, 20, 50, 100 et 1000.

Notez le meilleur résultat obtenu pour chaque algorithme dans le tableau ci-dessous à l'aide des matrices de confusion. Supprimez ensuite les autres exécutions.

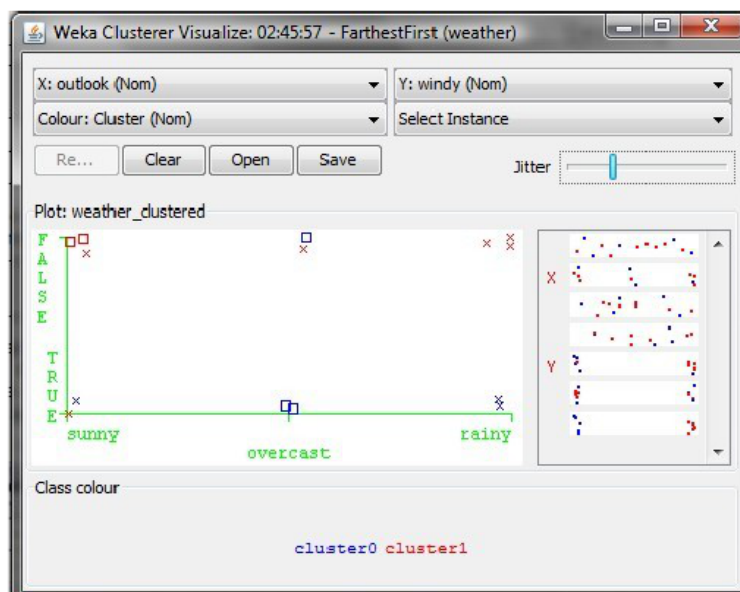
Algorithme	Seed	Taux d'err	Play=yes mal placées	Play=no mal placées
<i>SimpleKMeans</i>				
<i>FarthestFirst</i>				

Notez dans un cadre similaire à ci-dessous l'algorithme qui génère deux clusters correspondant le mieux aux classes *play=yes* et *play=no*, c-à-d avec le moins grand nombre d'erreurs.

• **EXERCICE XI** Cliquez avec le bouton droit dans le cadre *Result list* sur l'exécution de l'algorithme qui a donné le meilleur résultat et choisissez l'option *Vizualise cluster assignment*.

→ Lorsque l'option *Classes to clusters evaluation* a été utilisée, les instances bien et mal classées sont affichées dans la visualisation bi-dimensionnelle. Les croix représentent les instances placées dans le bon cluster (vrais positifs et vrais négatifs). Les carrés représentent les instances placées dans le mauvais cluster (faux positifs et faux négatifs).

Dans l'exemple ci-dessous, les 3 carrés bleus sont des instances *play=yes* mal placées; elles correspondent toutes à la valeur *outlook=overcast*. Les 2 carrés rouges sont des instances *play=no* mal placées; elles correspondent toutes aux valeurs *windy=false* et *outlook=sunny*.



Sélectionnez l'attribut *outlook* pour la dimension X, l'attribut *windy* pour la dimension Y et l'attribut *play* pour la couleur. Notez dans le cadre ci-dessous les combinaisons de valeurs de *outlook* et *windy* pour lesquelles aucune erreur n'est commise.

Outlook :		Outlook :	
Windy :		Windy :	
Outlook :		Outlook :	
Windy :		Windy :	

7.4 Choisir un modèle obtenu par K-means

Nous avons vu (section 2.2 page 6) les différences des erreurs de K-means selon la méthode de validation du clustering utilisée. La question à poser serait : **Quel modèle choisir ?**

Il n'y a pas de meilleur choix que de tester les deux modèles sur un (autre) ensemble de test avec assez d'instances.

Pour ce faire :

1. Charger p. ex la BD. *vote.arff* (fourni avec Weka)
2. Echantillonner 10% (appliquer Preprocess/Filter/Unsupervised/Instance/Resample) avec l'option "sans remise"
3. Sauvegarder les 10% (p. ex. sous le nom *vote-10-pourcent.arff*)
4. Refaire les étapes 1,2,3 mais cette fois avec 90% et sauvegarder sous le nom *vote-90-pourcent.arff*
5. Appliquer K-means (4 fois) de la manière suivante :
 - (a) Charger la BD originale *vote.arff* et appliquer *K-means* à toute la BD ; noter l'erreur (erreur SSE = 1510)
 - (b) Sans changer de BD., appliquer *K-means* avec la validation "pourcentage split / 66%" (erreur = 970.0)
 - La différence entre les deux erreurs a été relevée plus haut (en section 2.2 page 6) ;
 - On va voir les résultats dans le cas où on fournit une BD. de test séparée ;
 - (c) Charger la BD *vote-90-pourcent.arff*, sélectionner la méthode de validation "Supplied test set", Cliquer sur le bouton "set" en face et donner comme fichier de test *vote-10-pourcent.arff* ; Appliquer *K-means* ; noter l'erreur (erreur SSE = 1354)

Ce que l'on vient de faire est équivalent à ceci :

6. Appliquer (enfin) *K-means* à toute la BD. originale *vote.arff* avec la validation "pourcentage split / 90%". Cela nous rapproche du cas de "supplied test" ci-dessus et on obtient un erreur SSE=1322.
7. **Comparer** les clusters de ces 2 dernières applications. Ils sont assez proches !

- **EXERCICE XII** Reporter les éléments important de ces 4 essais. Puis décidez laquelle vous conservez

Méthode 1 :		Méthode 2 :	
Méthode 3 :		Méthode 4 :	

☞ Au delà des erreurs, quel modèle est mieux testé ?

En cas de résultats très proches, on fait confiance aux modèles mieux testés.

7.5 FilteredClusterer

On peut réaliser le même effet de la manière suivante : choisir la méthode "FilteredClusterer" ; cliquer dans les paramètres de la méthode ; choisir *K-means* et cliquer dans la zone de la méthode "K-means" et donner les même paramètres que plus haut.

Cela ne nous apporte rien de plus mais nous permet en fait de donner la méthode du clustering en paramètre. On l'utilisera plus loin pour la méthode EM.

Les algorithmes *SimpleKMeans* et *FarthestFirst* que nous avons utilisés jusque-là sont basés sur une approche par centroïde/médoïde. Nous allons maintenant tester les méthodes basées sur la densité et par construction de modèles.

8 Travail B

8.1 La méthode DBSCAN et la BD. Vote

Du package `optics-dbscan` (de Weka), DBSCAN est basé sur la densité et ne permet pas à l'utilisateur de définir le nombre de clusters à générer.

Il a pour principal paramètre le nombre minimal de points dans le voisinage *minPoints*. Sa valeur détermine le nombre de clusters générés en fonction de la distance de voisinage définie par le paramètre *epsilon* (V. Cours Chap-4-III).

Charger la BD. *Vote.arff* et conservez l'option "classes to cluster evaluation" (dans le cadre gauche).

- **EXERCICE XIII** Exécutez un clustering avec l'algorithme *DBScan* en sélectionnant l'option *Classes to clusters évaluation*.

Réalisez une expérimentation pour des valeurs du paramètre *minPoints* de 2, 3, 4 et 6 en laissant les autres paramètres à leur valeur par défaut.

☞ Pour certaines valeurs du paramètre *minPoints* l'algorithme ne peut s'appliquer sur ce jeu de données et un message d'erreur apparaît.
En cas de bug ou délai de réponse important, fermer cette fenêtre d'expérimentation et recommencer !

Notez le meilleur résultat obtenu dans le tableau ci-dessous à l'aide de la matrice de confusion. Supprimez les autres résultats dans la zone *Result list* (voir l'indication ci-dessous).

Algo	MinPoints	Taux d'err	Class= <i>Democrat</i> mal placées	Class = <i>Republican</i> mal placées
<i>DBScan</i>				

☞ **Indication** : essayer DBScan avec *minPoints*=3 et *Eps*=1.01 pour obtenir un clustering. Notez néanmoins le nombre d'instance qui n'ont pas trouvé de cluster.

- **EXERCICE XIV** Décider d'une valeur pour "MinPoints" (par exemple 2) et faites cette fois varier le paramètre "Epsilon" (le rayon du cercle d'une zone *dense*) en faisant des toute petites variations (par exemple de 1.00 à 1.02 en passant par 1.01, 1.015, ...) et notez les meilleurs résultats ci-dessous. Remarquons que si l'option "Class to Cluster" est cochée, la valeur 1.2 devrait vous donner 2 clusters).

Algo	Epsilon	Taux d'err	Class= <i>Democrat</i> mal placées	Class = <i>Republican</i> mal placées
<i>DBScan</i>				

- **EXERCICE XV** On décide d'utiliser la log-vraisemblance (voir section 12.4.1 page 37) pour comparer 3 méthodes. Choisissez la méthode "MakeDensityBasedClusterer". Cette (méta) méthode a besoin d'une (vraie) méthode de clustering (on dit qu'il s'agit d'un *wrapper*).

Cliquez dans la zone des paramètres de cette méthode, et devant "clusterer", cliquez sur "choose" et choisissez DBSCAN. Ici même, cliquez dans la ligne "DBSCAN" et donner lui "minPoints=2" et "epsilon=1.01". Cliquer sur OK, puis encore OK.

→ On vient de décider d'appliquer DBScan sous le contrôle des variances fait par MakeDensityBasedClusterer.

Cliquer "Start" et observez le contenu du cadre de droite.

Remplacer "DBScan" par "SimpleKMeans" et refaire l'expérience. Refaire avec "FarthestFirst".

Si DBSCAN affiche une erreur (pb. cluster vide), mentionner l'erreur et écarter DBSCAN de cette question.

Remplir la table suivante en faisant varier 3 fois les paramètres de vos méthodes et retenez les meilleurs résultats (LV = Log-Vraisemblance), noter vos divers paramètres (selon la méthode) et vos commentaires :

Algorithme	Paramètre	Meilleure LV	Nb Clusters	Taux erreur	Autres paramètres divers
<i>DBScan</i>					
<i>SimpleKMeans</i>					
<i>FarthestFirst</i>					

8.2 Méthode probabiliste EM et la BD. Weather

Il est fortement conseillé de lire la section 12.3.1 page 36 pour comprendre les résultats.

l'algorithme probabiliste **EM** (voir cours Chap4-II) permet de définir le nombre de clusters générés. Il a pour principal paramètre le *seed* et la valeur minimale de l'écart type *minStdDev*.

Appliquée à la BD. *weather-numeric.arff*, le résultat de l'application de la méthode EM avec *seed=20* et *nb. clusters=2* est donné ci-dessous.

→ Option *Classes to cluster evaluation* activée pour la classe *play*.

Parmi ces résultats, on remarque les estimateurs pour les attributs nominaux (*Outlook* signalé).

Considérons par exemple le cluster-1 et l'attribut *outlook* :

Attribute : outlook

Discrete Estimator. Counts = 2.96 2.99 1 (Total = 6.94)

Ces valeurs sont issues de l'application de l'estimateur de Laplace (véto) :

→ Utilisant *visualize cluster assignments* pour consulter les instances du cluster 1, on constate que parmi les 4 instances dans le cluster 1, pas de Outlook=rainy, la valeurs de sa fréquence (proba) est initialisée (comme pour les autres) à 1 et n'en change pas.

→ Pour outlook=sunny, deux des instances parmi les 4 ont cette valeur : la valeur initiale de la fréquence devrait passer de 1 (init) à 3.

Dans l'estimateur de Laplace appliqué, chaque occurrence rencontrée n'ajoute pas tout à fait 1 et tient compte du nombre totale des occurrences de ce couple attribut-valeur dans la BD. On a 5/14 occurrences de *outlook=sunny* et 4/14 pour *outlook=overcast*; une occurrence de *sunny* est davantage pénalisés (de très peu par rapport à 1) que *overcast* du à cette différence du nbr total d'occurrences. La valeur finale pour cette valeur de *Outlook=sunny* est 2.96 (pas tout à fait 3).

→ De même pour outlook=overcast avec une valeur finale 2.99 (pas tout à fait 3). Voir la page suivante ../.

```

=== Run information ===

Scheme:      weka.clusterers.EM -I 100 -N 2 -M 1.0E-6 -O -S 20
Relation:    weather
Instances:   14
Attributes:  5
              outlook
              temperature
              humidity
              windy
Ignored:     play
Test mode:   Classes to clusters evaluation on training data
=== Model and evaluation on training set ===

EM
==

Number of clusters: 2

Cluster: 0 Prior probability: 0.7183

  Attribute: outlook
  Discrete Estimator. Counts =  4.04 3.01 6  (Total = 13.06)
Attribute: temperature
Normal Distribution. Mean = 70.1574 StdDev = 3.6061
Attribute: humidity
Normal Distribution. Mean = 80.7353 StdDev = 11.043
Attribute: windy
Discrete Estimator. Counts =  6.04 6.01  (Total = 12.06)

Cluster: 1 Prior probability: 0.2817

  Attribute: outlook
  Discrete Estimator. Counts =  2.96 2.99 1  (Total = 6.94)
Attribute: temperature
Normal Distribution. Mean = 82.2771 StdDev = 1.9212
Attribute: humidity
Normal Distribution. Mean = 83.9571 StdDev = 5.5038
Attribute: windy
Discrete Estimator. Counts =  1.96 3.99  (Total = 5.94)
Clustered Instances

0      10 ( 71%)
1       4 ( 29%)

Log likelihood: -8.36599

Class attribute: play
Classes to Clusters:

0 1 <-- assigned to cluster
7 2 | yes
3 2 | no

Cluster 0 <-- yes
Cluster 1 <-- no

Incorrectly clustered instances : 5.0  35.7143 %

```

- **EXERCICE XVI** Appliquez l'algorithme EM en sélectionnant l'option *Classes to clusters evaluation* et en fixant le nombre de clusters à 2.

Réalisez une expérimentation pour des valeurs du paramètre *seed* de deux valeurs différentes en laissant les autres paramètres à leur valeur par défaut. Y a-t-il une différence dans les résultats ?

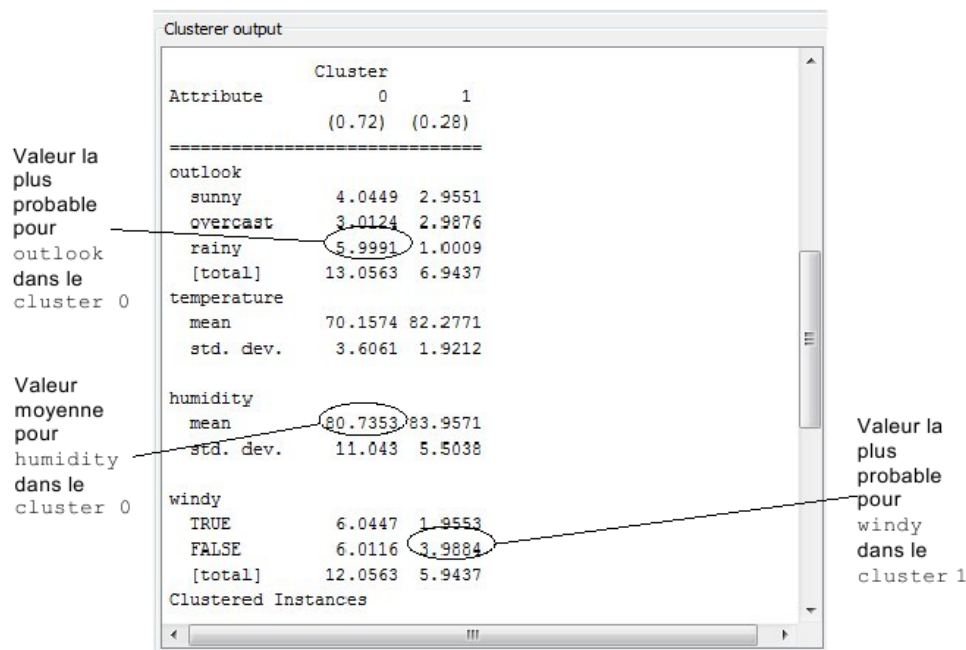
Si les résultats sont les mêmes, remplissez une ligne dans le tableau suivant. S'ils sont différents, remplissez deux lignes de ce tableau. A faire à l'aide de la matrice de confusion.

Algo	Seed	Taux d'err	Play=yes mal placées	Play=no mal placées
EM				

☞ Sauf dans le cas de peu de données et un nombre petit de clusters (comme ici), il doit y avoir normalement des changements quand on change la valeur du "seed". Le "seed" (*graine*) intervient dans le tirage des nombres aléatoires (cf. méthode EM, voir Cours, chapitre 5).

- **EXERCICE XVII** Observez la description du modèle généré dans la zone *Clusterer output*. Pour chaque cluster, en colonne, et pour chaque attribut est indiqué pour chacune de ses valeurs :
 - Une probabilité s'il s'agit d'un attribut nominal. La valeur la plus fréquente est celle dont le nombre est le plus élevé.
 - La moyenne (mean) et l'écart type (std. dev.) sil s'agit d'un attribut numérique.

Dans l'exemple ci-dessous, 2 clusters ont été générés. La valeur la plus fréquente pour outlook dans le cluster 0 est rainy. La valeur la plus fréquente pour windy dans le cluster 1 est FALSE. La valeur moyenne pour Humidity dans le cluster 0 est 80.7353.



Notez dans le tableau ci-dessous les valeurs les plus probables de chaque attribut pour chacun des 2 clusters.

Cluster	Outlook	Temperature	Humidity	Windy	Play
Cluster0 :					
Cluster1 :					

8.3 Application de CobWeb et la BD. Weather

L'algorithme par construction de modèles *Cobweb* ne permet pas de définir d'avance le nombre de clusters à obtenir. Ce sont les valeurs des paramètres qui détermineront le nombre de clusters. Le principal paramètre pour cela est le *CutOff* qui détermine le seuil de division des noeuds lors de la construction du modèle arborescent.

- **EXERCICE XVIII** Exécutez l'algorithme **Cobweb** à la BD. *weather-nominal.arff* en sélectionnant l'option *Classes to clusters evaluation* pour les valeurs du paramètre *CutOff* de 0.01, 0.1, 0.24 et 0.3. Identifiez la valeur du *CutOff* permettant d'obtenir 2 clusters correspondant respectivement à *Play=yes* et *Play=no* et complétez le tableau ci-dessous à l'aide de la matrice de confusion. Supprimez les autres résultats dans la zone *Result list*.

Algo	Cutoff	Taux d'err	Play=yes mal placées	Play=no mal placées
CobWeb				

- **EXERCICE XIX** Observez la description du modèle généré dans la zone *Clusterer output*. Celle-ci consiste en une arborescence qui n'est pas interprétable mais permet d'estimer la complexité du modèle (profondeur et largeur de l'arbre).
- **EXERCICE XX** (Bonus) Choisir au moins une parmi les Bases de données fournies (voir section 13, page 43), appliquez deux des méthodes vues ci-dessus et consignez le "meilleur" clustering (en terme de matrice de confusion).
- **EXERCICE XXI** : Lire aussi l'encadré d'explication ci-dessous.

Appliquez la méthode du clustering "Hierarchical Clustering" à la BD. "météo", modifier différents paramètres et consignez vos conclusions. Ces méthodes (il s'agit d'une méthode *hiérarchique agglomératif*) donnent-elles de meilleurs résultats que les précédentes ?

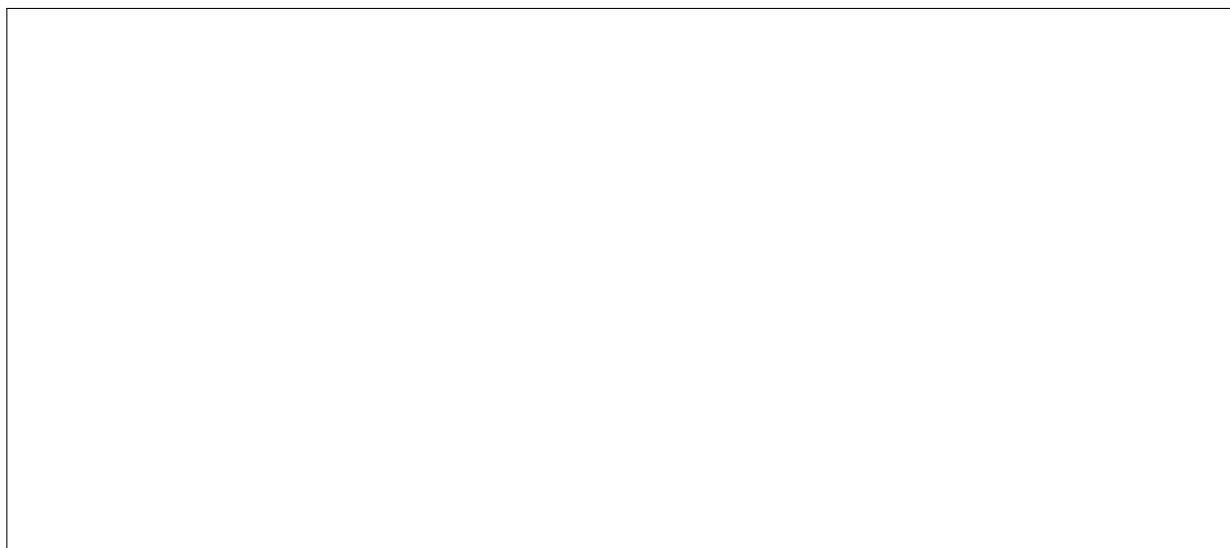
☞ **Pour connaître le rôle des paramètres de la méthode** : cliquez dans le nom de la méthode (zone sous les onglets), afficher les paramètres de la méthode et cliquez sur "more". Vous saurez le rôle de ces paramètres.

En particulier, appliquer différentes valeurs du paramètre "linkType" et donner l'effet de ce dernier sur les résultats.

➔ Le paramètre "**linkType**" (vu en cours) permet de préciser comment calculer la distance entre deux clusters à fusionner : entre les centroïdes, entre les 2 éléments les plus proches, les plus loin, toute paires 2-à-2, ...

- Relancer la méthode en choisissant 1 cluster pour obtenir toute l'arborescence du clustering (une sorte de peigne appelée *dendrogramme*). A reporter ci-dessous.

N.B. : Par un clique droit, on peut visualiser (sommairement) cette hiérarchie des clusters.



Que veulent dire les résultats de cette méthode ?

- Rappel du cours sur la méthode appliquée : on considère au départ autant de clusters que d'instances. On construit une matrice de distances puis on choisit les deux clusters les plus proches et on les fusionne. Cela modifie la matrice de distances.

On cherche deux autres clusters (les plus) "proches", on les fusionne, ;

Cette itération continue jusqu'à obtenir k clusters. Si $k = 1$, on obtient toute la hiérarchie des clusters.

- Dans la fenêtre "Result List" à gauche, un clique droit sur la méthode puis "visualize Tree" nous donne le dendrogramme de cette méthode hiérarchique agglomérative.

C'est cette même hiérarchie qui est donnée dans la fenêtre des résultats sous forme d'une expression très parentésée telle que :

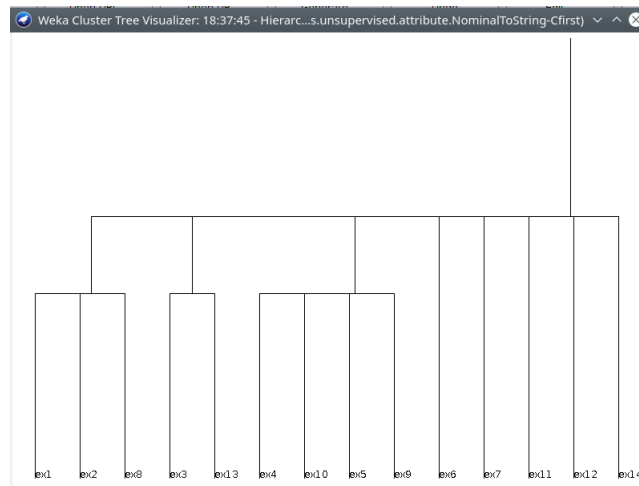
```
Cluster 0
(((((((1.0:1,1.0:1):0,1.0:1):0.41421,(0.0:1,0.0:1):0.41421):0,(((0.0:1,0.0:1):0,0.0:1):0.41421):0,
1.0:1.41421):0,0.0:1.41421):0,0.0:1.41421):0,0.0:1.41421)
```

On y comprend peu !

Voici la même expression si on "nomme" les observations dans la BD. On aurait alors :

```
Cluster 0
(((((((ex1:1,ex2:1):0,ex8:1):0.41421,(ex3:1,ex13:1):0.41421):0,(((ex4:1,ex10:1):0,ex5:1):0.41421):0,
ex6:1.41421):0,ex7:1.41421):0,ex11:1.41421):0,ex12:1.41421):0,ex14:1.41421)
```

Voir la figure suivante (+ explications dans l'encadré suivant).



- Si vous voulez y voir plus clair, utilisez le fichier fourni "weather-nominal-avec-num-ex.arff". Si non disponible, créer la BD comme suit.

Le contenu de ce fichier est le même que celui du "weather-nominal.arff" (fourni par Weka) auquel on a ajouté une première colonne contenant les valeurs "ex1" à "ex14". La première colonne n'a donc aucun autre intérêt que de rendre le dendrogramme plus lisible.

Lorsqu'on applique la même méthode à cette BD, on obtient dans la fenêtre des résultats :

```
Cluster 0
(((((((ex1:1,ex2:1):0,ex8:1):0.41421,(ex3:1,ex13:1):0.41421):0,(((ex4:1,ex10:1):0,ex5:1):0,ex9:1):0.41421):0
,ex6:1.41421):0,ex7:1.41421):0,ex11:1.41421):0,ex12:1.41421):0,ex14:1.41421)
```

Ce qui correspond au dendrogramme de la figure.

• Comment créer cette BD : ?

Sous un éditeur / Excel/ etc., ajouter une colonne de No à "weather-nominal.arff", le charger sous Weka puis utiliser *Preprocessing/Filters/Unsupervised/Attribute/NominalToString* et préciser *first* pour appliquer le filtre à la première colonne qui doit être de type String.

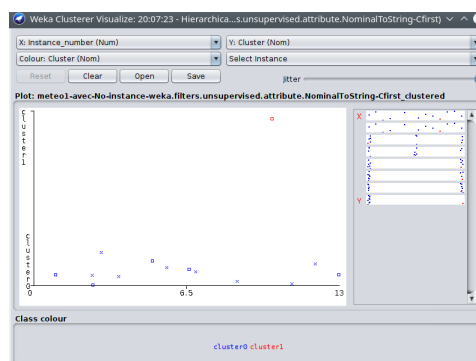
→ Vérifier que la commande dans la zone "Filter" soit : *NominalToString -C first*.

Appliquer le filtre et revenez à l'onglet "Cluster".

- Appliquez la méthode "Hierarchical Clustering" avec l'option "Debug=True" en précisant "numClusters = 2". Dans la zone "cluster mode" (à gauche), choisir les validations "Class to Cluster" et "Store clusters for visualization", appliquer la méthode puis visualisez son arbre (cliquez droit sur la méthode). Le dendrogramme ressemble à la figure précédente. Cette fois, la première colonne permet d'identifier les instances.

- Toujours dans la fenêtre "Result List", cliquez droit et sélectionnez "Visualize cluster assignments". Fixez l'axe X sur "instance number" et l'axe Y sur "Cluster". Pousser "Jitter" à droite pour zoomer (v. Fig. svte). Comme d'habitude, les petits carrés sont des erreurs et les croix les bons résultats : on a
cluster 0 \iff "play=yes" et cluster 1 \iff "play=no".

N'hésitez pas à cliquer sur les petites croix et carrés bleus et rouges pour visualiser les instances.



9 Travail C

9.1 Clustering sur la BD. Bank-data

La BD. *bank-data.csv* est fournie.

9.1.1 Application de la méthode K-means à Bank-data

Données :

Fichier de données bancaires "Bank-data.csv" (contenant un attribut "Id")

- L'attribut "save_act" : compte épargne
- L'attribut "current_act" : compte courant
- L'attribut "mortgage" : le client paie des agios (donc endetté)
- La classe PEP = le client ouvrirait un plan d'épargne populaire

- Charger le fichier "Bank-data.csv" (séparation des colonnes par une virgule).

Éliminer la colonne "Id" en cochant la case "Id" puis "remove" et sauvegarder le résultat dans "Bank-data-all-sans-Id.arff".

☞ La BD actuellement en mémoire sera le résultat de l'application de ces filtres.

Préparation des données

Généralement, les outils de clustering ne traitent pas les données numériques (il faut les discrétiser) et nécessitent la binarisation des attributs catégoriels (autant de colonnes 0/1 que de valeur pour un attribut dont une seule sera =1) WEKA dispose de tous les filtres pour réaliser ces changements mais il ne les impose pas (donc pas besoin de les appliquer ici).

L'algorithme *K-means* de Weka peut traiter des BDs avec une mixture de données catégorielles (valeurs énumérées comme {"homme", "femme"}) et des attributs numériques.

De plus, WEKA normalise automatiquement les attributs numériques pour le calcul des distances. Notons que sans cette normalisation, des données avec des échelles différentes (par exemple l'âge entre 0 et 100 cohabitant avec les revenus entre 1000 et 100000) peuvent souvent fausser les résultats.

- L'algorithme *K-means* (appelé *SimpleKmeans* sous Weka) n'accepte que la distance Euclidienne ou Manhattanne (somme des valeurs absolues des différences linéaires) pour calculer la distance entre les instances et les clusters.

Application du clustering :

- Recharger "Bank-data-all-sans-Id.arff".
- Dans l'onglet Clustering, sélectionner *SimpleKmeans*.
- Fixer nb clusters à 6 (la valeur par défaut = 2).
- Laisser la valeur "seed" inchangée. Cette valeur est utilisée pour initialiser les séquences random (par exemple pour le choix des Centroides initiaux). Une modification de "seed" peut donner un clustering différent.
- Choisir "Use Training Set" comme "Cluster Mode".
- Choisir "Store Cluster for Visualisation"
- Appuyer sur "start".

- **EXERCICE XXII** : Étudier les résultats et décrire ce qui vous permet empiriquement de distinguer / regrouper / résumer DEUX ou TROIS groupes distincts parmi les 6 clusters.

Groupe 1 :

Groupe 2 :

Groupe 3 (éventuellement) :

Mettre le paramètre "DisplayStdDevs" pour obtenir des informations sur les clusters.

On obtient des détails sur les vecteurs des moyennes et écarts types (pour les attributs numérique) ou des nombres et pourcentages qui caractérisent les centres des clusters.

- **EXERCICE XXIII** : A la lumière de ces détails, dans quelle mesure ces informations confirment ou non la réponse précédente ? (les mêmes ou d'autres ?).

- **EXERCICE XXIV** : relancer la méthode en choisissant le nombre de cluster (Deux ou Trois) suivant vos réponses précédents.

Les résultats du clustering confirment-ils vos réponses ? Commenter.

Partition des données en 2 ensembles d'apprentissage et de test

- On souhaite séparer les données en 90% pour apprendre et 10% pour test :
 - Preprocess/Filter/Resample puis cliquer dans la ligne de commande :
 - Mettre "noReplacement" (sans remise) = True pour permettre ce découpage (sinon, pas possible).
 - SampleSizePercent=90.0 pour choisir 90%
 - InvertSelection = True pour ne conserver que 100-90=10% des données.
 - "Apply" pour obtenir 10%. Remarquer que désormais, il y a 60 instances en mémoire.
 - Save dans le fichier "Bank-Data-test.arff"
 - La BD actuelle = 10% des données = 60 instances.
- Pour obtenir les 90% d'apprentissage, il faut recharger le fichier de départ
 - Recharger la totalité des données "bank-data-all-sans-Id.arff"
 - Refaire la même manipulation mais "InvertSelection=False" pour obtenir effectivement 90%.
 - Cliquer sur "Apply" pour obtenir 90%.
 - Il y a maintenant 540 instances en mémoire.

Stratégie d'initialisation K-means++ (rappelée ici et utilisée ci-dessous)

C'est une stratégie d'initialisation des médoïdes initiaux (l'élément central qui correspond à une instance existante vs. un centroïde = une moyenne qui ne correspond pas forcément à une instance).

Appelée "K-means++" (paramètre à fixer dans les paramètres de la méthode), elle donne de meilleurs résultats (que la méthode aléatoire de choix initiaux) en terme de la somme des carrées des erreurs (erreur SSE) ainsi que d'autres mesures d'évaluation.

La stratégie de cette méthode d'initialisation est de choisir des médoïdes éloignés les uns des autres tout en évitant de favoriser les outliers (un défaut de la méthode d'initialisation "farthest first").

Les Détails de la méthode K-means++ sont donnés plus haut. On va l'utiliser ci-dessous.

Application de SimpleKmenas :

- Charger le fichier d'apprentissage "Bank-Data-app.arff"
- Choisir SimpleKmenas avec la méthode "K-means++"
- Demander l'affichage de "StdDev"
- Choisir la distance "Manhattanne"
- Choisir nombre de clusters = 2
- Dans "Cluster Mode", choisir "supplied test set" et donner le nom du fichier de test "Data-bank-test.arff"

- **EXERCICE XXV** : appliquez la méthode et commenter les résultats sur les 60 instances de test. Faire varier (au moins 5 fois) le paramètre "seed" et reporter la meilleure valeur ci-dessous.

Comme ci-dessus, la mesure SSE permet d'évaluer la méthode.

Sauvegarde des résultats

Avec un clic droit dans la zone "Result list (...)", demander la visualisation des affectation des clusters.

Dans la fenêtre de visualisation fixer X=cluster et Y=instance_number. "Jitter" permet de zoomer sur la zone.

Sauvegardez les résultats dans le fichier "Bank-data-K-means-2clusters.arff". Ce fichier contiendra les 60 instances du fichier test et pour chaque instance, le cluster d'affectation est donné.

- **EXERCICE XXVI** : appliquez une méthode de classification (arbre de décision j48, cf. BE1) au fichier "Bank-data-K-means-2clusters.arff" en fixant le "cluster" comme classe. Commenter les résultats sur les 60 instances de test et reporter les règles obtenues.

Option "Classes to clusters evaluation"

Rappelons que dans ce mode, Weka ignore d'abord les classes et procède à un clustering. Puis en phase de test, il attribue les classes aux clusters suivant la majorité de la valeur de la classe. L'erreur de clustering est ensuite calculée en fonction cette affectation et la matrice de confusion correspondante est présentée.

☞ Si on active ce mode d'évaluation, on ne peut plus fournir un fichier de test, ni d'autres méthodes de test et d'évaluation.

On procède à un clustering des données Bank-data avec cette option.

- Charger le fichier d'apprentissage non scindé "Bank-Data-all.arff"
- Choisir SimpleKmenas avec la méthode "K-means++"
- Demander l'affichage de "StdDev"
- Fixer le nombre de clusters à 2.
- Dans le mode d'évaluation, activer "Classes to clusters evaluation"
- Appliquer la méthode
- On observe en base de la fenêtre des résultats la matrice de confusion.
- L'erreur du clustering est calculée à partir de cette matrice.
- **EXERCICE XXVII** : Faire varier les paramètres (seed) plusieurs fois et reporter ci-dessous le meilleur clustering.

Pour sauvegarder les résultats, demander la visualisation des résultats et dans la fenêtre de visualisation (cf. ci-dessus), sauvegardez les résultats dans un fichier "Bank-data-K-means-all.arff".

Ce fichier contiendra toutes les instances du fichier d'apprentissage et pour chaque instance, le cluster d'affectation est donné (en dernière colonne).

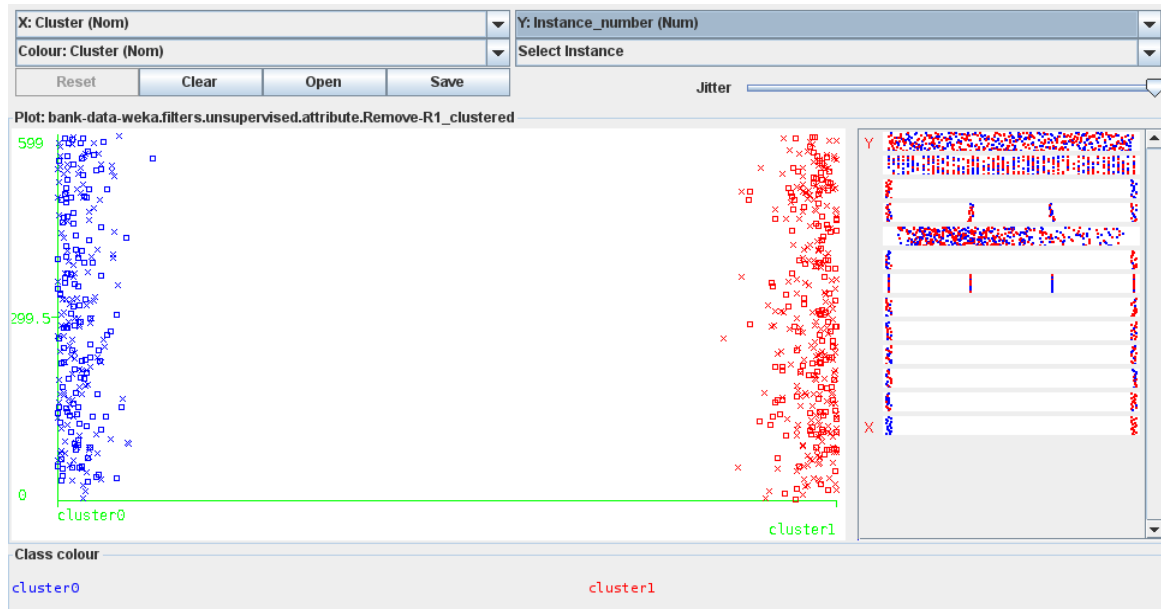
Important :

- Dans la figure de visualisation ci-dessous, une croix (quelque soit sa couleur) représente l'accord entre la classe du cluster et la classe de l'instance. Un carré pointe donc une erreur (désaccord classe-cluster).

Par exemple, dans la zone cluster0 (bleu) dont la classe est YES, les croix correspondent à PeP=Yes (accord classe-cluster = instance bien classée dans le bon cluster) et les carrés dénotent à PeP=No (donc une mauvaise classification).

De même, dans la zone cluster1 (rouge) dont la classe est NO, les croix correspondent à PeP=No (accord classe-cluster = instance bien classée dans le bon cluster) et les carrés dénotent PeP=YES (donc une mauvaise classification).

- On peut cliquer sur une croix/un carré pour accéder aux attributs de l'instance correspondante (voir la figure ci-après).



- **EXERCICE XXVIII** : appliquez une méthode de classification (arbre de décision, cf. BE1) au fichier "Bank-data-K-means-all.arff" en fixant le "cluster" comme classe. Commenter les résultats sur les 600 instances de données (all) et reporter les règles obtenues.

9.1.2 Application de la méthode EM à Bank-data

Sous Weka, la méthode EM attribue à chaque instance une distribution de probabilités d'appartenir à un cluster à chaque cluster.

EM décide du nombre de clusters (par une validation croisée) mais on peut spécifier ce nombre.

Pour décider du nombre de clusters, EM procède comme ci-dessous :

- 1- Le nombre de clusters est fixé à 1
- 2- L'ensemble d'apprentissage divisé en 10 plis.
- 3- EM procède à 10 clustering sur ces 10 plis (chaque clustering est évalué par la méthode habituelle XV)
- 4- On établit la moyenne des log-de-vraisemblances (loglikelihood). Voir section 12.4.1 en page 37.
- 5- Si cette valeur a été augmentée (d'au moins 10^{-6} par défaut), on ajoute 1 au nombre de clusters et on recommence en 2.

Application à l'exemple Bank-data :

- **EXERCICE XXIX** : Appliquer EM avec les paramètres par défaut. Noter le nombre de clusters que la méthode EM

propose et noter la log-vraisemblance. Faire varier les paramètres (nombre de clusters ≤ 6 , seed, nombre de plis) et donner le nombre de cluster et la log-vraisemblance optimaux.

• **EXERCICE XXX :**

Demander "DisplayModeInOldFormat" pour avoir plus d'information sur les clusters. En particulier, on obtient la probabilité a priori de chaque cluster.

Notons que le paramètre "nombre de slots" (par défaut = 1) permet d'utiliser les capacités de parallélisme de la machine.

Activer "Classes to clusters evaluation" dans "Cluster mode" et faire varier le nombre de clusters et appliquer la méthode. Donner vos conclusions.

• **EXERCICE XXXI : Qui dit mieux :** la meilleure log-vraisemblance (Voir section 12.4.1 page 37)

On travaille sur les données d'apprentissage "Bank-Data-app.arff" et de test "Bank-Data-test.arff".

Choisir la méta méthode "MakeDensityBasedClusterer", cliquer dans la ligne de commande et lui fournir à tour de rôle 5 méthodes applicables différentes. Choisir parmi Canopy, Cobweb, FilteredClusterer (avec EM), Hierarchical, LVQ, EM, Simplekmeans.

→ Si vous ne les voyez pas, il faut installer les packages correspondants.

Pour chaque cas, fixer (si possible) le nombre de cluster à 2.

Noter le nom de la méthode et la valeur de la log-vraisemblance produite.

→ Pour chaque méthode utilisée, cliquer sur le bouton "More" pour savoir un peu sur la méthode.

• **EXERCICE XXXII (Bonus) :** Appliquer la méthode du **clustering Hiérarchique** (méthode Weka "Hierarchical-Clusterer") sur cette BD et consignez brièvement vos résultats (texte libre).

10 Travail D

10.1 Clustering K-means sur la BD iris

- **EXERCICE XXXIII** : Charger la BD. *iris-2D.arff* (avec 2 attributs, fournie avec Weka) et procédez à un clustering via K-means comme ci-dessus. Fixer le nombre de cluster à 3.

Choisissez "Class to Cluster evaluation" et l'option "Store cluster for Visualisation".

Avec un clique droit sur la méthode exécutée, visualisez les clusters ("visualize Cluster Assignments").

1. Placer l'axe X sur "petallength" et l'axe Y sur "Instance Number". **Commentez.**
2. Placer l'axe X sur "petalwidth" et l'axe Y sur "Instance Number". **Commentez.**
3. Placer l'axe X sur "Cluster" et l'axe Y sur "Classer". Augmentez "Jitter" et **Commentez.**
4. Quel attribut "sépare" mieux les clusters (et discriminant en clustering) ?
5. **Faire de même** avec la BD. *iris.arff* fournie avec Weka. Dans cette BD., vous avez plusieurs autres attributs.
6. Quel attribut "sépare" mieux les clusters (quel attribut est plus discriminant en clustering) ?

10.2 Clustering EM sur la BD iris

- **EXERCICE XXXIV** : avec la BD *iris.arf*, procédez à un clustering EM (probabiliste, voir cours 7 et ci-après). Laisser le nombre de cluster à "-1" (la méthode les trouvera) et procédez au clustering avec les autres options comme ci-dessus.

1. Notez la valeur **Log Likelihood**, puis visualisez les clusters.
2. Ensuite, fixer le nombre de cluster à 3 et procédez de la même manière.
Notez la valeur **Log Likelihood**, puis visualisez les clusters.
3. Compte tenu des valeurs des *Log-vraisemblance* et les visualisations, **Quel** résultat du clustering est préférable ?

10.3 Test avec BD. séparée

- **EXERCICE XXXV** : la BD. *iris.arff* chargée, procédez (comme en BE2 et la BD. *spam.arff*) pour créer le fichier *iris-10.arff* de test contenant 10% des données *iris.arff*.

Rappel : utilisez le filtre / unsupervised / instance et fixer "SampleSizePercent" à 10. Appuyer sur "Apply" puis "Save".

Rechargez la BD. *iris.arff* et procéder de la même manière mais cette fois fixer "SampleSizePercent" à 90. Ces 90% serviront à l'apprentissage dans le fichier *iris-90.arff*. Après l'application du filtre, ces 90% sont en mémoire.

On va appliquer la méthode EM.

Dans la zone "Cluster mode", choisissez "Supplied TEst Set" et ouvrir *iris-10.arff*.

Procédez au clustering EM avec l'option "Debug" activée et "NumClusters"=3.

Notez le Log Vraisemblance et visualisez les graphiques avec X="Class".

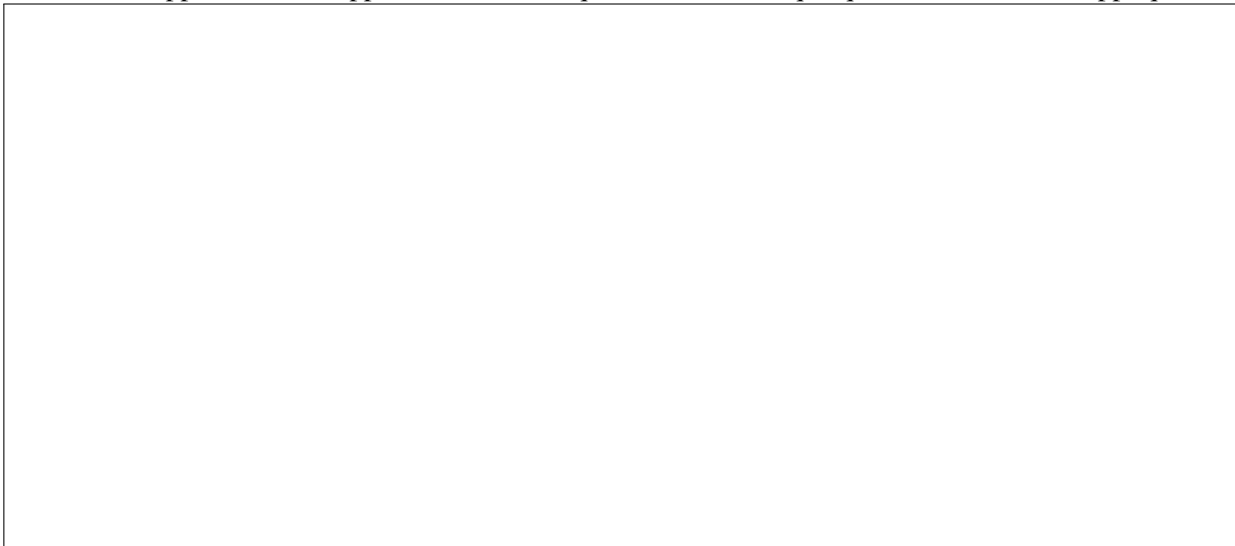
1. Suivant les visualisations, **Combien** d'instances sont mal classées (confondez les classes et les 3 couleurs des 3 clusters).

Choisissez maintenant l'option "Class to Cluster Evaluation" et appliquez EM sur la BD. des 90% (qui est toujours en mémoire). Notez le taux de l'erreur (la matrice de confusion) et le Log likelihood.

2. Comparer avec la question (1) ci-dessus. Faites-vous confiance à ce clustering ? Est-ce que les résultats du point (1) sont meilleurs que ceux du point (2) ?

Bien faire attention à l'interprétation de la matrice de confusion.

☞ Si vous lancez Weka sous Linux / Mac, la console de lancement de Weka (si Weka lancé via java) affiche des informations supplémentaires appréciables. Notez que ceci est le cas quelque soit la méthodes appliquée.



11 Consignes sur le Travail à rendre

Certaines BDs nécessitent une conversion au format *.arff* (voir passgae *.csv* à *.arff* ci-dessus).

Consignes :

Il vous est demandé un compte-rendu détaillé de votre méthodologie ainsi que de vos résultats.

Vous réaliserez un rapport d'une quinzaine de pages environ qui détaillera :

- votre démarche générale,
- votre compréhension des données (informations, types, quantité...)
- les informations que vous pensez pouvoir en tirer,
- les différentes méthodes de fouille utilisées : pour être crédible, il est nécessaire d'appliquer *au moins deux méthodes* aux données.
- les résultats obtenus pour chacune d'elle,
- une synthèse de ces résultats, ainsi que leur interprétation.
- Une **comparaison (statistique ?) des résultats** (cf. BE1, BE2).

Vous devriez maintenant être capable de retenir des critères de comparaison.

12 Compléments sur les méthodes du Clustering sous Weka

On applique différentes méthodes de clustering présentes dans Weka sur la BD *météo*, avec attributs mixtes (weather.arff) ou tous *nominal* (weather-nominal.arff).

Cependant, pour certaines méthodes, la BD. a pu être différentes. Vous disposez de toutes ces BDs.

12.1 SimpleKMeans

SimpleKMeans sur weather.arff (mixte) :

```
=== Run information ===
Scheme:          weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance
                  -R first-last" -I 500 -S 10
Relation:        weather.symbolic
Instances:       14
Attributes:      5
                  outlook
                  temperature
                  humidity
                  windy
                  play
Test mode:       evaluate on training data

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

K-means
=====

Number of iterations: 4
Within cluster sum of squared errors: 26.0
Missing values globally replaced with mean/mode

Cluster centroids:
Attribute      Full Data      Cluster#
                (14)        0          1
                (14)        (10)        (4)
=====
outlook         sunny      sunny      overcast
temperature     mild       mild       cool
humidity        high       high       normal
windy           FALSE      FALSE      TRUE
play            yes        yes        yes

Clustered Instances

0      10 ( 71%)
1       4 ( 29%)
```

N.B. : Si vous cochez *DisplayStdDev* (ligne Choose), la fenêtre *Results* contiendra des informations supplémentaires :

```
Cluster centroids:
Attribute      Full Data      Cluster#
                (14)        0          1
                (14)        (10)        (4)
=====
outlook         sunny      sunny      overcast
sunny           5 ( 35%)   5 ( 50%)   0 (  0%)
overcast        4 ( 28%)   2 ( 20%)   2 ( 50%)
rainy           5 ( 35%)   3 ( 30%)   2 ( 50%)

temperature     mild       mild       cool
hot             4 ( 28%)   3 ( 30%)   1 ( 25%)
mild            6 ( 42%)   6 ( 60%)   0 (  0%)
cool            4 ( 28%)   1 ( 10%)   3 ( 75%)

humidity        high       high       normal
high            7 ( 50%)   7 ( 70%)   0 (  0%)
normal          7 ( 50%)   3 ( 30%)   4 (100%)

windy           FALSE      FALSE      TRUE
TRUE            6 ( 42%)   4 ( 40%)   2 ( 50%)
FALSE           8 ( 57%)   6 ( 60%)   2 ( 50%)

play            yes        yes        yes
yes             9 ( 64%)   6 ( 60%)   3 ( 75%)
no              5 ( 35%)   4 ( 40%)   1 ( 25%)
```

12.1.1 Phase d'évaluation en SimpleKmeans

Le choix de *Supplied test set* ou *Percentage split* évalue les résultats du clustering sur un ensemble de test si modèle probabiliste (eg. EM ou CobWeb).

Avec le choix de *Classes to Cluster evaluation*, Weka ignore d'abord la classe précisée et crée le modèle (les clusters) puis en phase de test, assigne la classe (l'attribut désigné, ici *Play*) au clusters par un vote majoritaire des membres du cluster. L'erreur du clustering est alors calculée sur la base de cette assignation.

Un exemple est donné ci-dessus pour *SimpleKmeans*.

```
=== Run information ===

Scheme:      weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last"
-I 500 -S 10
Relation:     weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy

Ignored:      play

Test mode:     Classes to clusters evaluation on training data
=== Model and evaluation on training set ===

K-means
=====

Number of iterations: 4
Within cluster sum of squared errors: 21.000000000000004
Missing values globally replaced with mean/mode

Cluster centroids:

Attribute      Full Data      Cluster#
              (14)        (10)        (4)
=====
outlook        sunny        sunny        overcast
temperature    mild         mild         cool
humidity       high        high         normal
windy          FALSE       FALSE       TRUE

Clustered Instances

0      10 ( 71%)
1       4 ( 29%)

Class attribute: play
Classes to Clusters:

0 1 <-- assigned to cluster
6 3 | yes
4 1 | no

Cluster 0 <-- yes
Cluster 1 <-- no

Incorrectly clustered instances : 7.0 50 %
```

12.2 Farthest First

Kmédioïde clustering : la stratégie Farthest First

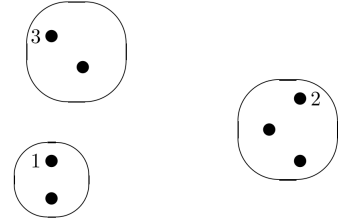
→ Stratégie de choix des centroïde / médioïdes.

• Pour un objet o et un ensemble S d'objets, $\text{dist}(o, S)$ est une métrique (une relation ordonnée, par exemple Euclidienne) permettant de mesurer "l'éloignement" entre o et tout point de S .

• Dans sa version triviale, la stratégie Farthest-First sélectionne m_1 au hasard, puis m_2 est choisi le plus éloigné de m_1 , etc... avant d'itérer ensuite sur (par exemple) K-means.

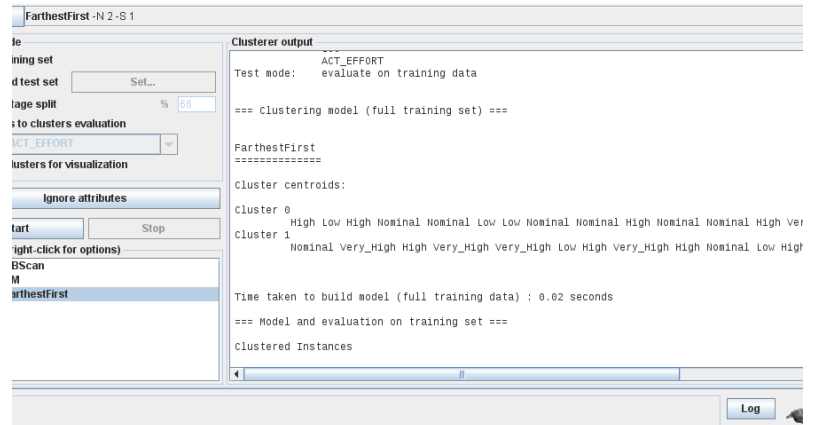
→ Exemple pour $k=3$:

Ici, la distance peut être une métrique euclidienne triviale.



Cette stratégie ne donne pas forcément de bons résultats car elle privilégie les outliers (supposés être des bruits / erreurs). Voir également la stratégie K-means++.

→ Pour l'exemple du graphe ci-contre et pour 2 clusters, si $m_1 = e$, alors $m_2 = a$. Une itération sur K-means devrait nous donner les clusters $\{e\}$ et $\{c, a, b, d\}$ (ou $\{a\}$ et $\{c, e, b, d\}$).



• Des variantes de Farthest First existent lorsque la métrique distance n'est pas triviale.

Par exemple, on peut appliquer la méthode suivante :

- On choisit m_1 comme ci-dessus.

- Ensuite, une fois le médioïde m_i choisi, on trouve le point $p \in S$ le plus proche de $\{m_1, m_2, \dots, m_i\}$ puis on choisit le point m_{i+1} le plus éloigné de p .

→ Pour l'exemple du graphe ci-dessus et pour 2 clusters, si $m_1 = e$, alors $p = d$ le plus proche de m_1 puis $m_2 = a$ est le plus éloigné de d . Les clusters obtenus sont les mêmes que ci-dessus.

○ Ici, la distance aura été :

pour un objet o et un ensemble S d'objets, soit $\text{dist}(o, S) := \min \{ \text{dist}(o, p) \mid p \in S \}$.

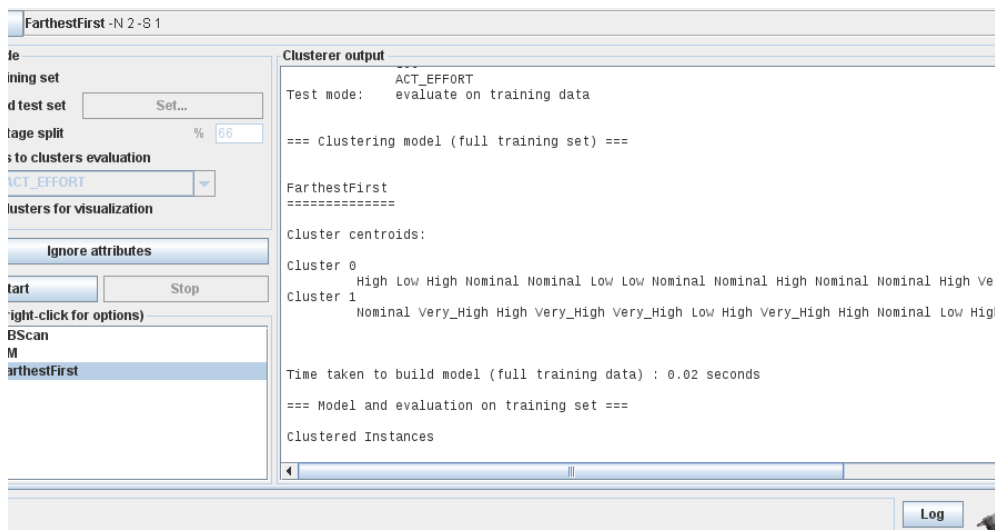
→ Le choix du prochain médioïde m_i : $\text{argmax}_{m_i} \text{dist}(m_i, p)$

• Vois aussi **K-means++** (traité ci-dessus).

• L'algorithme de *Farthest First* est de complexité $O(nk)$ (nb données \times nb clusters).

Farthest first est également une étape qui permet d'avoir une idée de k pour p. ex K-means.

C'est un algorithme assez rapide en particulier pour de **BD de grande taille**.



12.3 EM

Le schéma génère un clustering probabiliste en termes de la moyenne et de l'écart type pour les attributs numériques et un comptage pour les attributs nominaux.

Le problème de veto (probabilité nulle) est géré par l'ajout de la constante 1 et modifié par une petite valeur (voir cours, méthode Bayésienne, l'estimateur de Laplace).

En cas de l'évaluation "Classes to clusters", le modèle assigne une classe aux clusters et on obtient des informations supplémentaires telles que la *log vraisemblance* (section 12.4.1 page 37), la *matrice de confusion* et le *taux d'erreur*.

```
=== Run information ===

Scheme:      weka.clusterers.EM -I 100 -N -1 -M 1.0E-6 -O -S 100
Relation:    weather
Instances:   14
Attributes:  5
              outlook
              temperature
              humidity
              windy

Ignored:     play

Test mode:   Classes to clusters evaluation on training data
=== Model and evaluation on training set ===

EM
==

Number of clusters selected by cross validation: 1

Cluster: 0 Prior probability: 1

Attribute: outlook
Discrete Estimator. Counts =  6 5 6  (Total = 17)
Attribute: temperature
Normal Distribution. Mean = 73.5714 StdDev = 6.3326
Attribute: humidity
Normal Distribution. Mean = 81.6429 StdDev = 9.9111
Attribute: windy
Discrete Estimator. Counts =  7 9  (Total = 16)
Clustered Instances

0      14 (100%)

Log likelihood: -8.75386

Class attribute: play
Classes to Clusters:

0 <-- assigned to cluster
9 | yes
5 | no

Cluster 0 <-- yes

Incorrectly clustered instances : 5.0 35.7143 %
```

12.3.1 Exemple d'application de la méthode EM

Petite application de la méthode EM à la BD météo. Lire l'encadré suivant.

A propos des résultats de la méthode EM :

Pour la BD "météo" chargée, la méthode EM a tendance à ne conserver qu'un seul cluster. Aller dans les paramètres de la méthode EM et fixer "numCluster" à 2 (pour la BD "météo").

☞ Un autre point : si dans les paramètres, vous demandez "Debug=True", quelques résultats complémentaires seront affichés dans la console qui a lancé Weka (Mac et Linux vérifié, Windows?). En particulier, la probabilité pour chaque instance d'être dans un des clusters.

☞ Il est également possible d'obtenir ces probabilités d'appartenance aux clusters via le filtre *weka.filters.unsupervised.attribute.ClusterMembership* dans l'onglet "preprocessing", en choisissant EM comme méthode. Voir aussi le '*' en bas de cet encadré.

La (log) Vraisemblance (voir aussi la page suivante) :

Un exécution de la méthode de clustering EM sur les données Météo (avec attributs mixtes) produit parmi les résultats **Log likelihood : -9.4063** (tout en bas).

On a $e^{-9.4063} = 0.0001 \rightarrow$ la vraisemblance = 0.0001.

Que représente la valeur *log likelihood* affichée ? Elle semble très petite.

Cette valeur doit être considérée en comparaison avec d'autres exécutions de la méthode EM (par exemple en variant la valeur du "Seed" ou le nombre de clusters...).

La plus grande valeur de la vraisemblance est meilleure.

(*) On peut également utiliser la méta-méthode "**MakeDensityBasedClusterer**" (onglet "cluster") et y préciser la méthode de clustering qui devrait être utilisée (par défaut *SimpleKmeans*), récupérer la valeur de *Log likelihood* affichée pour la méthode en question afin de comparer celle-ci avec le *Log likelihood* (section 12.4.1) d'autres méthodes (par exemple EM).

- EM utilise un algorithme itératif pour trouver le maximum de vraisemblance ou une estimation du maximum a posteriori (MAP) des paramètres du modèle statistique où le modèle dépend de *variables latentes non observées* (cf. les 5 paramètres, voir cours).
- EM itère en calculant la vraisemblance évaluée suivant l'actuelle estimation des paramètres (phase E) puis maximise (étape M) les dits paramètres qui ont donné le log vraisemblance de l'étape E. Ces estimations sont ensuite utilisées pour déterminer la distribution des variables latentes à l'étape E suivante.... etc.
- Les clusters trouvés sont donnés (à partir du numéro 0) suivant la probabilité *a priori* des clusters (le cluster '0' a donc le maximum de probabilité).
- **Les avantages de la méthode EM :**
 - Donne des résultats très utiles pour les DB avec des données réelles (vraie données)
 - On utilise EM quand par ex. K-means ne donne pas de résultats intéressants pour des petit cas, voire, des zones d'intérêt dans une BD.

The screenshot shows the Weka EM algorithm configuration window. The 'Cluster mode' section has 'Use training set' selected. The 'Cluster output' section shows the results of the EM algorithm. The output is a table with columns for 'Attribute' and 'cluster' (0, 1, 2, 3, 4, 5). The table shows the distribution of data points across clusters for different attributes.

Attribute	cluster 0 (0.18)	cluster 1 (0.02)	cluster 2 (0.22)	cluster 3 (0.27)	cluster 4 (0.08)	cluster 5 (0.05)
RELY						
Nominal	8	1	12	1	1	4
Very_High	1	2	1	1	2	1
High	5	1	3	17	5	1
Low	1	1	1	1	1	1
[total]	15	5	17	20	9	7
DATA						
High	3	2	2	1	1	3
Low	1	1	13	17	1	1
Nominal	10	1	1	1	6	2
Very_High	1	1	1	1	1	1
[total]	15	5	17	20	9	7
CPLX						
Very_High	1	1	1	1	2	2
High	8	2	13	17	4	2
Nominal	3	1	2	1	1	2
Extra_High	1	1	1	1	2	1
Low	3	1	1	1	1	1
[total]	15	5	17	20	9	7

- L'inconvénient de EM est d'être une méthode (algorithme) un peu compliqué !

12.4 Remarque

☞ La méthode EM utilise la validation X-V et la *log-likelihood* sur une BD pour déterminer le meilleur nombre de clusters pour cette BD.

CobWeb ainsi que DbScan déterminent également un nombre optimal de clusters.

Ce nombre optimal est trouvé par une interaction entre le complexité (cutoff) / erreur et la justesse des modèles.

Mais sous Weka, ces éléments ne sont pas toujours présentés comme paramètres des méthodes (seules certaines comme CobWeb le demandent). Dans ce cas, on a recours à d'autre environnement (comme *SciKitLearn*).

12.4.1 A propos de la Log-vraisemblance

Pour comprendre les détails de ce calcul, voir le cours Chapitre 5.

Rappelons que cette valeur n'a pas de signification particulière en soi. Par contre, elle est utilisée pour une comparaison relative des performances de différentes application de la méthode EM.

→ En appliquant plusieurs fois la méthode EM (en variant les paramètres, en particulier le "seed"), **plus la valeur de loglikelihood est élevée, meilleur est la modèle obtenu.**

☞ La méthode EM n'est pas la seule à produire une log-vraisemblance.

On peut par exemple choisir la méthode de clustering "MakeDensityBasedClusterer" qui est un "wrapper" (une méta méthode paramétrée par une vraie méthode de clustering) puis choisir par exemple "SimpleK-means" comme méthode à appliquer et observer les résultats, en particulier la **log-vraisemblance**, à comparer avec la log-vraisemblance obtenue par d'autres méthodes telles que EM sur les mêmes données.

→ On choisira le modèle avec la meilleure log-vraisemblance.

12.5 Filtres Weka , clustering et EM

Il est possible d'appliquer un filtre aux données (dans l'onglet "Preprocess").

Par contre, dans ces filtres, on ne peut pas disposer de l'option "Classes to clusters evaluation" mais la classe d'origine des instances est prise en compte si elle est présente dans les données.

① Obtenir le cluster de chaque instance :

Par exemple, pour réaliser l'équivalent des manipulations ci-dessus et sauvegarder le cluster de chaque instance obtenu via EM (à l'exclusion de l'option "Classes to clusters evaluation"), on peut procéder par :

- Charger le fichier "Bank-data-all.arff"
 - Dans l'onglet "Preprocess", choisir "Filters" puis "Filter/unsupervised/attribut" et choisir "AddCluster".
 - Cliquer dans la ligne de la commande et choisir la méthode de clustering EM. Weka propose alors les options de la méthode. On peut d'ailleurs choisir toute autre méthode de clustering.
 - Fixer le nombre de clusters à 2.
 - Cliquer sur "Apply".
 - On constate que la BD en mémoire est augmentée d'un nouvel attribut qui est le cluster de chaque instance.
- On peut Editer / sauvegarder les résultats dans ce même onglet "Preprocess". Notons qu'on n'a plus accès aux détails des résultats de la méthode EM appliquée.

☞ Attention : la BD en mémoire vient d'être modifiée. Si on souhaite revenir en arrière (après avoir sauvegardé les résultats), on élimine le nouvel attribut "cluster" et on continue.

② Obtenir la probabilité de cluster de chaque instance :

Il est également possible d'obtenir la probabilité pour chaque instance d'appartenir à chaque cluster. Pour cela :

- Charger le fichier "Bank-data-all.arff"
- Dans l'onglet "Preprocess", choisir "Filters" puis "Filter/unsupervised/attribut" et choisir "ClusterMembership".
- Cliquer dans la ligne de la commande et choisir la méthode de clustering EM. Weka propose alors les options de la méthode. On peut d'ailleurs choisir toute autre méthode de clustering.
- Fixer le nombre de clusters à 2.
- Cliquer sur "Apply". On constate que la BD en mémoire est totalement modifiée.

Maintenant ../.

Maintenant, on dispose des attributs de la forme "pCluster_X_Y" représentant la probabilité d'appartenir aux clusters. Ici, X représente la classe d'origine de l'instance (si elle est dans la BD initiale) et Y représente un des clusters.

→ Par exemple, l'attribut "pCluster_0_1" donne la probabilité pour l'instance étant de la classe 0 (rapelons qu'on avait 2 classes dans les données d'apprentissage : YES/NO) d'appartenir au cluster No 1 (2e cluster = le cluster YES). La première instances des données a une probabilité maximale dans cette colonne et donc cette instance étant de classe PeP=YES est dans le 2e cluster (dont l'étiquette =YES).

→ De même, pour la 2e instance, la probabilité maximale est dans la colonne "pCluster_1_0" : l'instance est de la classe 1 (PeP=No) et son cluster est 0 (1er cluster dont le label est No).

On peut éditer / sauvegarder les résultats. Notons qu'on n'a plus accès aux détails des résultats de la méthode EM appliquée.

☞ Attention : la BD en mémoire vient d'être totalement modifiée.

12.5.1 Un autre filtre pour obtenir des probabilités

Egalement, Sous Weka, le filtre *weka.filters.unsupervised.attribute.ClusterMembership* permet d'obtenir, pour EM et DBscan les probabilités dans les clusters.

- Dans l'onglet **Preprocess**, choisir *filters > unsupervised > attribute > ClusterMembership*.

Par défaut, la méthode proposée est EM.

Si on applique cette méthode à la BD météo-numérique, les données en mémoire deviennent les clusters (en 2 colonnes pour 2 clusters) et la classe *play* est préservée.

Il s'agit donc d'une **transformation des données en probabilités** d'appartenance à l'un des deux clusters.

On peut ensuite appliquer la méthode de clustering EM à ces données et obtenir un taux de 14% de mal-classification, bien meilleur que toutes les autres clustering qu'on a directement appliquées à la même BD. d'origine.

☞ Revenons au filtre appliqué. Cliquer dans la ligne de commande du filtre ci-dessus puis sur "choose". On propose cette fois la méta-méthode "MakeDensityBasedClusterer". Choisir ce dernier puis cliquer dans sa ligne de commande. Dans la nouvelle fenêtre qui s'ouvre, on propose par défaut d'appliquer *K-means* mais dans le bouton "Choose", on nous propose l'ensemble de méthodes de clustering de Weka.

Bien entendu, pour chacune de ces méthodes, on peut intervenir dans les paramètres de la commande comme si on appliquait un clustering.

Si on choisit par exemple EM, ce sera comme si on procédait à un clustering EM des données issues du filtre EM ci-dessus. la différence est qu'on a peu de résultats affichées.

☞ Si on lance WEKA en ligne de commande (ou en CLI), a-t-on les probas de membership ? Pour certaine méthodes, on a des infos supplémentaires dans la console de lancement de Weka mais pas pour toutes.

☞ Notez bien que cette démarche peut s'appliquer à toute autre méthode de clustering. La méthode EM est ici un exemple parmi d'autres, choisie pour son aspects probabiliste.

12.5.2 Similarités entre K-means et EM

- Pour K-means, il y a des heuristiques pour améliorer différents choix initiaux (choix des centroïdes, de k, etc). Ces heuristiques sont très similaires à EM pour mixture Gaussienne et les deux algorithmes (K-means et EM) utilisent la même démarche de raffinement des paramètres et des clusters.

☞ K-means cherche à trouver des clusters avec un contexte similaire alors que EM peut trouver des clusters de différentes formes.

- N.B. : pour de données avec beaucoup d'attributs, K-means semble plus efficace que le clustering hiérarchique.

12.6 Cobweb

Cobweb génère un clustering hiérarchique où les clusters sont décrits par probabilités.

On peut choisir d'ignorer l'attribut *play* (bouton *ignore attributes*) étant donné que le nombre important de clusters minimisera l'effet du choix "Classes to clusters".

🔊 Il se peut que selon les différentes versions de Weka, les résultats obtenus soient plus ou moins différents ! Dans ce cas, ce sont les résultats de vos exécutions qui comptent.

🔊 Essayer avec **CutOff=0.23** pour voir 2 clusters (essayez 0.24 aussi !).

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

```
Scheme:      weka.clusterers.Cobweb -A 1.0 -C 0.23 -S 42
```

```
Relation:    weather
```

```
Instances:   14
```

```
Attributes:  5
```

```
outlook
```

```
temperature
```

```
humidity
```

```
windy
```

```
Ignored:
```

```
play
```

```
Test mode:   Classes to clusters evaluation on training data
```

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

```
Number of merges: 2
```

```
Number of splits: 1
```

```
Number of clusters: 6
```

```
node 0 [14]
```

```
| node 1 [8]
```

```
| | leaf 2 [2]
```

```
| node 1 [8]
```

```
| | leaf 3 [3]
```

```
| node 1 [8]
```

```
| | leaf 4 [3]
```

```
node 0 [14]
```

```
| leaf 5 [6]
```

```
Clustered Instances
```

```
2      2 ( 14%)
```

```
3      3 ( 21%)
```

```
4      3 ( 21%)
```

```
5      6 ( 43%)
```

```
Class attribute: play
```

```
Classes to Clusters:
```

```
2 3 4 5  <-- assigned to cluster
```

```
2 1 3 3 | yes
```

```
0 2 0 3 | no
```

```
Cluster 2 <-- No class
```

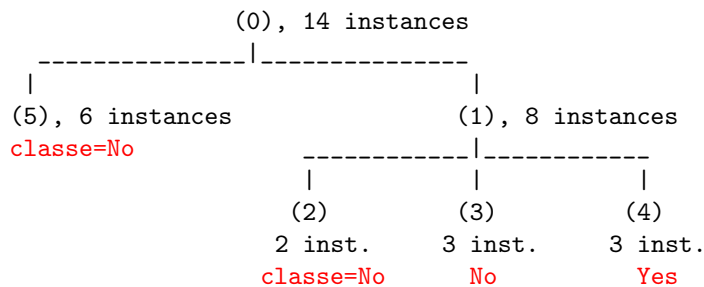
```
Cluster 3 <-- No class
```

```
Cluster 4 <-- yes
```

```
Cluster 5 <-- no
```

```
Incorrectly clustered instances : 8.0  57.1429 %
```

<<=== Faites afficher l'arbre (clic droit dans *Results* et choisir "visualize tree")



Les clusters entre parenthèses (xx)
Ou sous la forme "leaf xx"

🔊 L'arbre dessiné dans l'encadré ci-dessus peut être visualisé sous Weka.

La méthode Cobweb crée un dendrogramme (dit *classification tree*) où chaque cluster est caractérisé par une description de probabilités.

☞ Elle fait du *hiérarchical clustering* avec des probabilités. Pour tenir compte de la classe (dans les erreurs), il faut une classe nominale (pas numérique).

Cobweb utilise une mesure d'évaluation heuristique appelée "category utility" qui guide la construction de l'arbre. Elle procède ainsi en intégrant les instances dans l'arbre pour maximiser "category utility".

Une classe (cluster) peut être créée à la volée ; ce qui représente une grande différence entre Cobweb et K-means.

Par ailleurs, Cobweb fait du *merge* et *split* de classes sur la base de "category utility" ; ce qui permet de faire une recherche bi-directionnelle. Par ex, un *merge* peut défaire un *split* précédent.

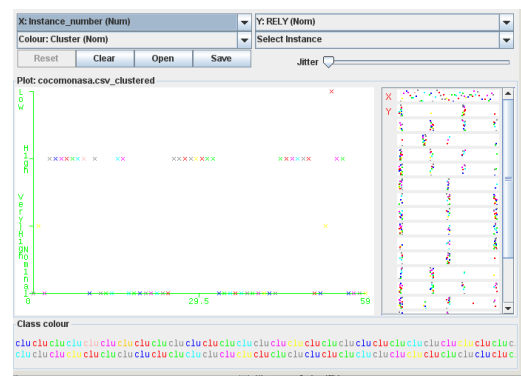
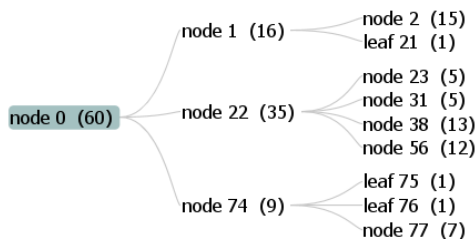


Par comparaison, K-means fait seulement un traitement unidirectionnelle : le cluster d'une instance est défini par sa distance au centre du cluster. Ce qui peut être sensible aux *outliers*.

Cependant, Cobweb a des limitations à cause des hypothèses initiales qu'elle fait :

- Les distributions de probabilité sur les différents attributs sont statistiquement indépendants, ce qui n'est pas tjs le cas car il existe souvent des corrélations entre les attributs.
- La représentation de ces distributions de probabilités des clusters rend le stockage des clusters couteux. En particulier pour les attributs avec beaucoup de valeurs différentes car la complexité dépend non seulement du nombre des attributs mais aussi des domaines de ces attributs.
- Si l'arbre de classification n'est pas balancé p. ex. à cause de l'asymétrie statistique (*skewness*) des données, il causera une complexité bien plus grande. *K-means* n'a pas ce problème (car pas de probabilité dans *K-means* !) mais seulement la distance, ce qui à son tour semble impropre pour des données de grande taille.

Résultats de CobWeb :



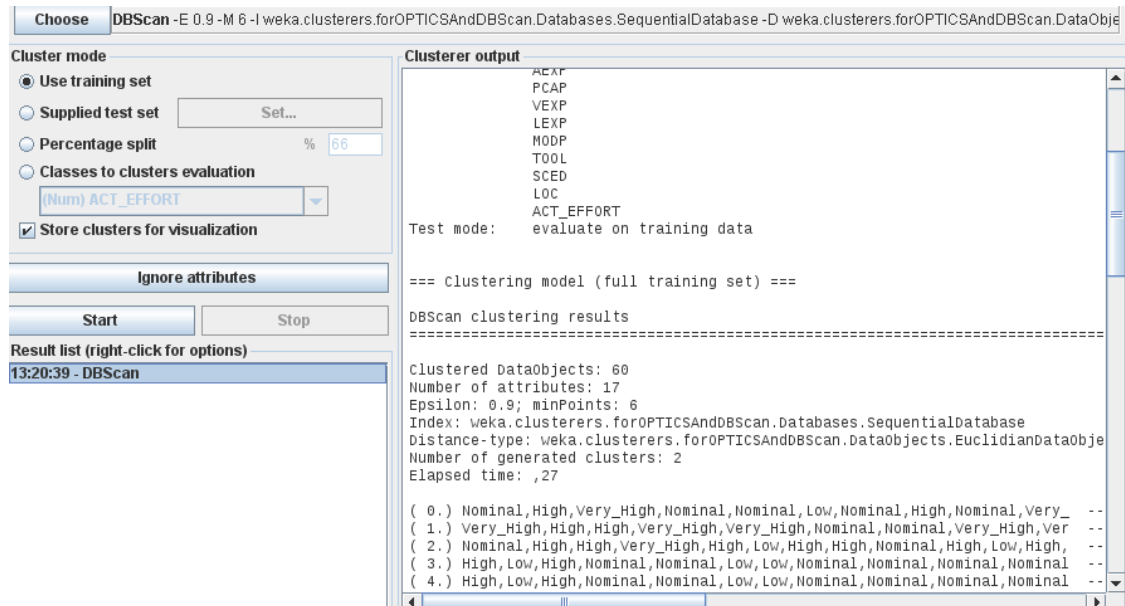
☞ Une fois dans la fenêtre de visualisation des clusters, on peut sauvegarder les résultats et avoir un fic arff où on a les clusters.

validation : training set car la classe dans cette BD est numérique (NASA.arff).

CobWeb a trouvé un grand nbr de clusters. j'ai appliqué K-means pour 2 clusters.

12.7 DBSCAN

- Une méthode très utilisée.
- Trouve des clusters en commençant par une estimation de la distribution de la densité des noeuds.
- La méthode de clustering OPTICS est une généralisation de DBSCAN remplaçant le param ϵ avec un maximum de diamètre (radius).



Les avantages de DBSCAN :

- Pas besoin de connaître d'avance le nbr de clusters
- Avec le paramètre *MinPts*, l'effet de la méthode *single-link* qui relie des clusters par une fine ligne (eg. dans "SimpleKmeans") est réduit.
- DBscan peut trouver des clustres entourés par d'autre clusters (grâce à *MinPts*).
- DBscan gère les données bruitées
- Il lui faut juste 2 paramètres (ϵ et *MinPts*);
- Elle est insensible à l'ordre des instances dans la BD. Seuls les points à la frontière de 2 clusters peuvent changer de cluster si l'ordre des instances change.

Inconvénients de DBSCAN :

- Les résultats de DBscan sont dépendants de la mesure de distance. En général, c'est une distance Euclidienne mais pour de grande dimensions, une telle distance risque d'être inutilisable. Cependant, on note que cet inconvénient est présent dans toutes les autres méthodes.
- DBScan ne peut pas créer de bons clusters si les données représentent de grande différences dans leurs densités car les deux paramètres de la méthode (ci-dessus) ne peuvent pas être choisis pour répondre à ces différentes régions (de différentes densités) de données.

12.8 A propos de la méthode OPTICS

OPTICS (*Ordering Points To Identify Clustering Structure*) est une méthode très proche de DBSCAN. Elle construit un clustering **sur la base de DBSCAN** pour minimiser le nombre de paramètres à donner en entrée.

On donne (cf. DBSCAN) le paramètre *Core-Distance*, un plus petite ϵ permettant de considérer une instance comme *core-object* ainsi que la distance d'**atteignabilité** qui est la distance entre tout couple d'instances d'un cluster.

L'atteignabilité est calculée comme le maximum entre la *core-distance* et la distance Euclidienne entre 2 instances. Ces distances permettent d'ordonner les instances de la BD. Les clusters sont définis suivant la distance d'atteignabilité et la core-distance associées à chaque instance (on stocke ces valeurs avec les données de la BD). Cela permet d'extraire plus d'informations des clusters obtenus.

☞ Weka a quelques problèmes avec cette méthode ! Cela dépend en fait de la BD utilisée.

Clusterer output

OPTICS clustering results

Clustered DataObjects: 60
 Number of attributes: 17
 Epsilon: 0.9; minPoints: 6
 Write results to file: no
 Index: weka.clusterers.forOPTICSAndDBScan.Databases.SequentialDatabase
 Distance-type: weka.clusterers.forOPTICSAndDBScan.DataObjects.EuclidianDataObject
 Number of generated clusters: 0
 Elapsed time: ,06

(0.) Nominal,High,Very_High,Nominal,Nominal,L --> c_dist: UNDEFINED r_dist: UNDEFI
 (1.) Very_High,High,High,Very_High,Very_High, --> c_dist: UNDEFINED r_dist: UNDEFI
 (10.) Nominal,Nominal,High,High,Nominal,Nomina --> c_dist: UNDEFINED r_dist: UNDEFI
 (11.) High,Nominal,High,Nominal,Nominal,Nomina --> c_dist: UNDEFINED r_dist: UNDEFI
 (12.) Nominal,Nominal,High,Nominal,Nominal,Nom --> c_dist: UNDEFINED r_dist: UNDEFI
 (13.) Nominal,Nominal,High,Nominal,Nominal,Nom --> c_dist: UNDEFINED r_dist: UNDEFI

OPTICS Visualizer - Main Window

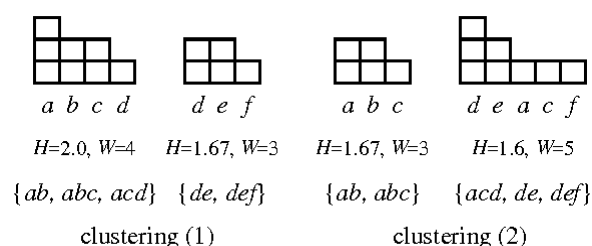
File View Help

Table Graph - Epsilon: 0.9, MinPoints: 6

Key	DataObject	Core-Distance	Reachability-Distance
0	Nominal,High,Very_High,Nominal,Nominal,Low,Nominal,High,Nomina...	UNDEFINED	UNDEFINED
1	Very_High,High,High,Very_High,Very_High,Nominal,Nominal,Very_Hig...	UNDEFINED	UNDEFINED
10	Nominal,Nominal,High,High,Nominal,Nominal,Nominal,Nominal,Nomi...	UNDEFINED	UNDEFINED
11	High,Nominal,High,Nominal,Nominal,Nominal,Nominal,Nominal,Nomi...	UNDEFINED	UNDEFINED
12	Nominal,Nominal,High,Nominal,Nominal,Nominal,Nominal,Nominal,N...	UNDEFINED	UNDEFINED
13	Nominal,Nominal,High,Nominal,Nominal,Nominal,Nominal,Nominal,N...	UNDEFINED	UNDEFINED
14	Nominal,Nominal,High,Nominal,Nominal,Nominal,Nominal,Nominal,N...	UNDEFINED	UNDEFINED
15	High,Nominal,Nominal,High,Nominal,Nominal,Nominal,Nominal,High...	UNDEFINED	UNDEFINED
16	High,Nominal,High,Nominal,Nominal,Nominal,Nominal,Nominal,High...	UNDEFINED	UNDEFINED
17	Nominal,Nominal,Nominal,Nominal,Nominal,Nominal,Nominal,Nominal...	UNDEFINED	UNDEFINED
18	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.41597	UNDEFINED
19	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.10098	0.41597
20	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.05771	0.10098
21	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.10098	0.05771
23	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.09064	0.05771
24	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.06729	0.05771
22	Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,Hi...	0.09446	0.05771
2	Nominal,High,High,Very_High,High,Low,High,High,Nominal,High,Low...	UNDEFINED	UNDEFINED
25	High,Low,High,Nominal,Nominal,Low,Low,Nominal,Nominal,Nominal...	0.04236	UNDEFINED
26	High,Low,High,Nominal,Nominal,Low,Low,Nominal,Nominal,Nominal...	0.0397	0.04236
4	High,Low,High,Nominal,Nominal,Low,Low,Nominal,Nominal,Nominal...	0.05029	0.0397

12.9 A propos de la méthode CLOPE

Cette méthode cherche à améliorer le ratio entre la hauteur (H) et la largeur (W) de l'histogramme des clusters obtenus. On retiendra le clustering avec le plus grand ratio (voir fig. ci-dessous).



13 Les Bases de données (pour ce BE)

☞ Voir annexes BE2 pour les explications sur les BDs.

13.1 Exemple météo (ou Golf)

Sur le jeu de données "météo", lancer l'extraction des règles d'association (méthode A Priori). Essayer diverses méthodes de test.

Num	Temps(S)	Temperature(T)	Humidité(H)	Vent(V)	Classe
1	ensoleillé	Elevée	Elevée	non	N
2	ensoleillé	Elevée	Elevée	oui	N
3	nuageux	Elevée	Elevée	non	P
4	pluvieux	Moyenne	Elevée	non	P
5	pluvieux	Faible	Normale	non	P
6	pluvieux	Faible	Normale	oui	N
7	nuageux	Faible	Normale	oui	P
8	ensoleillé	Moyenne	Elevée	non	N
9	ensoleillé	Faible	Normale	non	P
10	pluvieux	Moyenne	Normale	non	P
11	ensoleillé	Moyenne	Normale	oui	P
12	nuageux	Moyenne	Elevée	oui	P
13	nuageux	Elevée	Normale	non	P
14	pluvieux	Moyenne	Elevée	oui	N

La même base d'instance avec deux attributs numériques :

Num	Temps(S)	Temperature(T)	Humidité(H)	Vent(V)	Classe
1	ensoleillé	81	78	non	N
2	ensoleillé	80	90	oui	N
3	nuageux	83	80	non	P
4	pluvieux	75	96	non	P
5	pluvieux	69	75	non	P
6	pluvieux	64	70	oui	N
7	nuageux	65	65	oui	P
8	ensoleillé	72	83	non	N
9	ensoleillé	68	72	non	P
10	pluvieux	71	74	non	P
11	ensoleillé	75	69	oui	P
12	nuageux	70	77	oui	P
13	nuageux	85	70	non	P
14	pluvieux	73	82	oui	N

13.2 Exemple Banque

Une banque dispose des informations suivantes sur un ensemble de clients (20 transactions).

Client	Montant	Classe-Age	Résidence	Etudes	Internet
1	moyen	moyen	village	oui	oui
2	élevé	moyen	bourg	non	non
3	faible	âgé	banlieue	non	non
4	faible	moyen	bourg	oui	oui
5	moyen	jeune	ville	oui	oui
6	élevé	âgé	ville	oui	non
7	moyen	âgé	banlieue	oui	oui
8	faible	moyen	village	non	non
9	moyen	moyen	bourg	oui	oui
10	moyen	moyen	village	non	non
11	faible	âgé	banlieue	oui	oui
12	faible	jeune	ville	non	non
13	élevé	âgé	bourg	non	non
14	faible	moyen	banlieue	non	non
15	faible	moyen	village	non	oui
16	faible	jeune	banlieue	oui	oui
17	faible	ag�	village	non	non
18	�lev�	�g�	bourg	oui	non
19	faible	moyen	banlieue	non	oui
20	moyen	jeune	village	oui	oui

L'attribut ternaire *Montant* d crit la moyenne des montants sur le compte client. Le second attribut ternaire *Classe-Age* donne la tranche d' ge du client. Le troisi me attribut ternaire *R sidence* d crit la localit  de r sidence du client. Le dernier attribut binaire *Etudes* a la valeur oui si le client a un niveau d' tudes sup rieures.

La **classe associée** à chacun de ces clients correspond au contenu de la colonne *Internet*. La classe oui correspond à un client qui effectue une consultation de ses comptes bancaires en utilisant Internet.

A faire à l'aide d'une calculatrice ou d'un tableur : Classer ces exemples dans leur Clusters.

Client	Montant	Classe-Age	Résidence	Etudes	Internet
21	moyen	âgé	village	oui	oui
22	élevé	jeune	ville	non	oui
23	faible	âgé	banlieue	non	non
24	moyen	moyen	bourg	oui	non