Extraction de Connaissances
Document utile aux BEs

Préparation / Manipulation / Échantillonnage
des données
Comparaison des modèles
avec

WEKA

ECL - Option Informatique - 2022-23

Octobre 2023

Table des matières

1 Manipulation d'extracteur WEKA

1.1 installation de Weka

WEKA est un utilitaire Open Source d'apprentissage automatique écrit en Java et développé à l'université de Waïkato en Nouvelle Zélande (Site Web : `http://www.cs.waikato.ac.nz/ml/weka/`).

Sur la page `http://www.cs.waikato.ac.nz/ml/weka/downloading.html`, téléchargez la version de développement (*Developer version*) pour votre machine.

Weka est réalisé en Java (archive jar) et existe pour les plate-formes Windows, Linux et Mac.

Pour utiliser Weka, vous aurez besoin de la machine abstraite Java. Sous Windows ou MacOS, le mieux serait de récupérer la version contenant la machine virtuelle Java qui convient.
☞ Les essais et tests de ce documents sont réalisés avec différentes version de Weka (depuis 3.5 .. 3.9).

WEKA implante un grand nombre d'algorithmes d'apprentissage automatique.

Pour lancer Weka

- **Sous Linux / MacOS :** placer vous dans le répertoire d'installation puis tapez :

  `java -jar weka.jar`

  ou bien cliquer sur l'archive `weka.jar`.

  ☞ Pour demander plus de mémoire (BD volumineuse) : `java -Xmx1000M -jar weka.jar`

- **Sous Windows :** récupérer sur le même site la version AVEC la machine virtuelle Java.
  Installer puis lancer l'exécutable Weka dans le répertoire d'installation (ou sur *Weka* dans les "programmes")

☞ Il se peut que la façon de lancer Weka se modifie d'une version à une autre. Vérifier la présence d'un fichier exécutable qui vous dispensera de lancer java à la main (voir avec l'enseignant).

1.2 Mémoire pour Weka

Réclamation de la mémoire pour Weka (si grosses BDs. ou méthodes compliquées) :

☞ Pour lancer Weka, si problèmes de mémoire, lancer Weka plutôt avec l'option

java -Xmx2048m -jar weka.jar

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

2 Utilisation de Weka

Le principe d'utilisation basique de Weka est le suivant :

1. Charger un fichier de données en mémoire (les données préparées, mises au format accepté par *weka*).
2. Répéter Jusqu'au chargement d'un autre fichier de données
   (a) Éventuellement procéder à un pré-traitement (par exemple, discrétisation)
   (b) Appliquer une méthode d'Extraction de Connaissances
   (c) Visualiser (sous différentes formes) les résultats, les sauvegarder éventuellement ...

2.1 Chargement d'un fichier de données

Après le lancement de Weka, dans la première fenêtre de WEKA, cliquez sur ***Explorer*** (ou *Application* → *Explorer*) : cela fait apparaitre une interface composée d'onglets.
Pour ouvrir (et charger en mémoire) un fichier de données, aller sur l'onglet ***Preprocess*** et cliquez sur ***Open file...***, et choisissez votre fichier de données "météo" `weather.nominal.arff` (situé éventuellement dans le répertoire `data`). Le contenu de cette base de données est rappelé dans la section 10.2 en page 20.

3 Format de fichiers

3.1 Format arff

Le format de fichier utilisé par Weka (`.arff`) est constitué comme suit :

- Une entête (*header*) qui contient :
  - le nom de la relation (au sens "Base de Données" relationnelle) : `@RELATION <relation_name>`
  - une liste d'attributs et leurs types : `@ATTRIBUTE <attribute_name> <datatype>` où `datatype` peut prendre les valeurs `numeric`, `string` où un ensemble énuméré de valeurs de la forme {`val1, val2, val3`}.

  ➙ Un exemple de header pourrait être :

```
% Commentaire : ici, on écrit du blabla
@RELATION ventes
@ATTRIBUTE num_vente numeric
@ATTRIBUTE nom_vendeur string
@ATTRIBUTE detail_facture numeric
@ATTRIBUTE total numeric
@ATTRIBUTE paiement {especes, CB, cheque}
```

- Un corps qui contient les données. Le début du corps est marqué par le mot clé `@DATA`. Les différentes valeurs d'un tuple (une instance) sont séparés par une virgule. Les valeurs inconnues sont spécifiées par un " ?".

  Un exemple d'une section "DATA" pourrait être celui-ci :

```
@DATA
1,'Corine',120,25.34,especes
2,'Jean-Luc',124,32,CB
3,'Michel',126,58.25,CB
....
```

3.2 Format csv

☞ Nous pouvons également charger un fichier *csv* (séparateur = virgule).

3.3 Format data pour Weka (pour C4.5)

Vous pouvez placer vos données dans un fichier .data, par exemple *exemple.data* (sans fournir les noms des attriburs) et placer les informations sur ces données dans *exemple.names*.
Le format du fichier exemple.names est :
énumération des classes, séparés par ','
nom 1er attribut : type
...
nom dernir attribut : type

• Voir l'exemple "glass". Vous pouvez
- soit charger sous weka le fichier (du format C4.5 = J48) *glass.data* pour lequel la présence de *glass.names* (contenant les noms et types des attributs) est requise ;
- soit charger *glass.arff* (qui a été produit en enregistrant la BD *glass.data* chargée sous Weka).
- soit charger *glass.csv* (qui a été produit en enregistrant la BD glass.data chargée sous Weka).

• Il est possible de demander à Weka de sauvegarder une BD chargée en mémoire sous différents formats (arff, csv, date, ...)

Attention : lorsque l'on crée soit même un fichier *.csv* à partir d'un fichier *.data*, faire attention aux types des données. Dans l'exemple glass, la classe n'est pas numérique mais énuméré (de 1..7). Une erreur sur ce type supprime un ensemble de méthodes applicables.

Par exemple, si dans l'exemple *glass*, on déclare la classe de type numérique, la méthode *j48* n'est plus applicable.

4 Préparation des données sous Weka

Weka dispose d'un certain nombre d'outils d'exploration, filtrage et préparation de données. Voir onglets *"preprocess"* et *"select attributes"*. Quelques filtres seront présentés ci-dessous.

> **Nota Bene :** Après l'application de tout filtre expliqué dans cette partie (discrétisation, échantillonnage, anonymisation, normalisation, etc.), les données sont modifiées en mémoire et peuvent être utilisées par exemple pour une classification. Les données d'origine ne sont plus en mémoire.
> Si vous voulez par exemple essayer une autre discrétisation, **il faudra recharger** le fichier initial.

4.1 Discrétisation sous Weka

Dans WEKA, certaines méthodes comme ID3 (mais pas C4.5) n'acceptent que des données de type nominal. Dans ce cas, lorsque l'on charge un fichier "arff" avec des données numériques (ou mixtes = mélange numérique-nominal), on peut procéder à une discrétisation des attributs numériques.

Manipulation : Dans l'onglet **Preprocess**, charger les données de l'exemple "météo" (fichier *weather.arff*) qui contient des attributs mixtes (cf. 2e tableau ci-dessous).

Cliquez ensuite sur *Filter/Choose* (sous *Open file*) puis choisir *filters/unsupervised/attribute/Discretize*.
Cliquer ensuite dans la ligne de commande de discrétisation qui s'est affichée (*Discretize -unset ...*) pour faire afficher la fenêtre des paramètres.
Par défaut, Weka discrétise TOUS les attributs (paramètre *attributeindices*). Si vous voulez discrétiser un attribut de votre choix, ou bien appliquer des discrétisations différentes, modifier le paramètre *attributeindices*. Par exemple, la valeur *2* ne discrétise que le 2e attribut et *2,4* applique une discrétisation aux 2e et 4e attributs. Les valeurs *first-* et *-last* vous permettent de désigner le 1er et le dernier attributs.
Le paramètre *bins* désigne le nombre d'intervalles souhaités. La valeur de *bins* est fixée à 10 par défaut.
Pour les attributs numériques de l'exemple "météo" chargé, tester 3 intervalles (*bins*) puis faites afficher les partitions obtenues en cliquant sur le nom de l'attribut.
Pour l'application de la discrétisation, appuyer sur **Apply** à droite de la ligne de commande.

4.1.1 Un exemple de discrétisation

Chargez le fichier *weather.arff* (météo). Il y a deux attributs numériques *temerature* et *humidity*.
Dans la même fenêtre, devant *Filter* (dans la partie supérieure de la fenêtre), cliquez sur *Choose*. Sélectionner *filters/unsupervised/attribute/Discretize*. Vous verrez devant le bouton *Choose* s'afficher :

Discretize ...
-B 10 : 10 partitions
-M -1.0 : poids des instances par intervalle (ici rien, cliquer sur more pour en savoir plus))
-R first-last : tous les attributs.

Cliquez dans cette zone des paramètres, fixer le nombre de partions (bins) à 3, changez *attributeindices* en 2,3 (seuls les 2 attributs numériques concernés). Fermer cette petite fenêtre des paramètres puis cliquez sur *apply* tout à droite de cette zone.
Maintenant, dans la zone où sont affichés la liste des attributs, cliquez sur *temerature* ou *humidity* pour voir les partitions.

> **Nota Bene :** Après l'application d'une discrétisation, les données sont modifiées en mémoire et peuvent être utilisées par exemple pour une classification. Les données d'origine ne sont plus en mémoire.
> Si vous voulez par exemple essayer une autre discrétisation, **il faudra recharger** le fichier initial.

4.2 Suppression des instances par leur indice

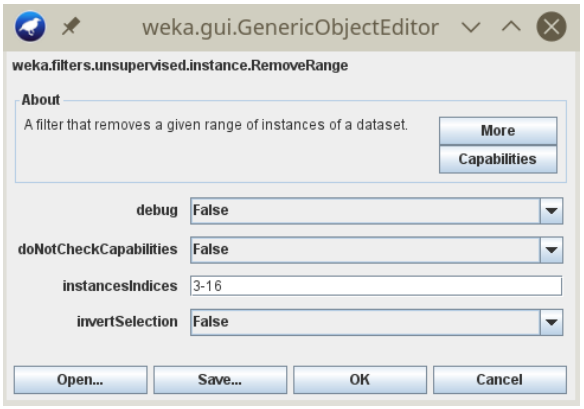
Sous WEKA, on peut supprimer des instances de la DB suivant leur indice (numéro d'ordre).

• Dans l'exemple ci-contre appliqué à *banque-app.arff*, on se donne l'objectif d'éliminer les instances allant de 3 à 16.

- Charger le fichier de données *banque-app.arff*.

- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance* et dans la liste proposée, choisir **RemoveRange**.

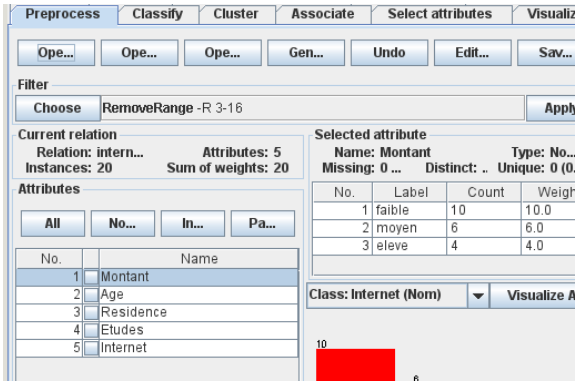
→ Cliquer sur **Apply** pour que sur les 20 instances de la BD, il nous reste 6.



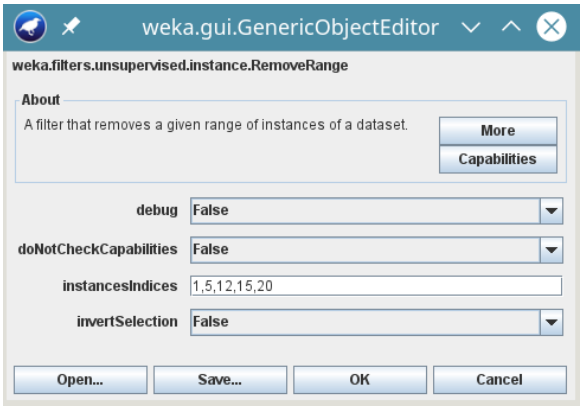
• Comme pour toute commande, dans la fenêtre qui s'affiche, le bouton **More** nous dit davantage sur cette commande.

En particulier, ici, on remarquera :

- `instancesIndices` : l'intervalle d'indice ou les numéros des instances à supprimer.
  On donnera l'attribut qui servira de filtre. Pour nous, ce sera **1** (premier attribut = **Montant**).
  Cette information est visible dans l'onglet *Preprocess* (où on est actuellement), dans la zone des attributs de la base de données *banque-app.arff* chargée.



- `invertSelection` : inverser le sens de la sélection : au lieu de supprimer les instances désignées, on les garde et on supprime les autres.



• L'exemple ci-contre montre les options pour supprimer les instances dont on précise les numéros, séparés par des virgules.

4.3 Anonymisation de données sous Weka

Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance*. Puis choisir :
Datafly : anonymisation des données (pour ne pas pouvoir identifier des informations éventuellement sensibles telle qu'un nom, etc...

5 Conversion / modification des des données

5.1 Normalisation / centrage-réduction des données

• Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/attribute*.

- **Normalize** : normilise les données numériques (sauf la classe, si on le précise dans la commande). Le résultat sera des valeurs réelles dans l'intervalle $[0, 1]$. Ce qui veut dire que min=0 et max=1.
  On utilise la normalisation quand on ne connaît pas la distribution des valeurs d'un attribut ou bien on sait qu'elle n'est pas gaussienne.
  ☞ Si vous choisissez dans les options de ce filtre $scale = 2.0$ et $translation = -1.0$, vous obtiendrez des valeurs normalisées réelles dans l'intervalle $[-1, +1]$.
- **Standardize** (ou centré-déduit) : on remet à l'échelle les valeurs d'un attribut pour avoir un eùoyenne nulle et une variance 1.
- Voir les autres filtres / possibilités

5.2 Dénormalisation : Conversion de données transactionnelles

Les données transactionnelles (par exemple un ticket de caisse pour des achats) sont souvent stockés dans des BDs avec un identifiant (ID). Par exemple un numéro d'ordre de client (ou un nom). Cette information joue le rôle d'une clef discriminante : une clé qui désigne un ensemble d'informations de manière unique.

Les transactions (concernant par exemple un numéro de client) peuvent être scindées sur plusieurs lignes de la table (chacune avec le même ID). Une BD. dans ce format doit être convertie en une ligne par transaction avant de pouvoir être utilisée par un classificateur ou d'en extraire des règles d'association, clustering, etc sous Weka .

Le filtre de dé-normalisation (*denormalize*) de Weka peut effectuer ce genre de processus d'aplatissement. Ce filtre nécessite que :
1- les données contiennent un champ ID unique qui identifie chaque transaction distincte, et que
2- les données soient déjà triées dans l'ordre de cet ID.

Exemple : la base de données (à gauche) et le fichier de données de Weka (*exemple_numerique.arff*) au milieu et le fichier (*exemple_nominal.arff*) avec les attributs numériques transformés en catégoriels (à droite) :

User	ItemID	Sequence	TimeSpent
1	1	1	5
1	2	2	1
1	5	3	8
1	6	4	12
1	8	5	2
2	1	1	7
2	2	2	3
2	3	3	3
2	4	4	2
2	5	5	7

```
@relation exemple_numerique
@attribute User numeric
@attribute ItemID numeric
@attribute Sequence numeric
@attribute TimeSpent numeric
@data
1, 1, 1, 5
1, 2, 2, 1
1, 5, 3, 8
1, 6, 4, 12
1, 8, 5, 2
2, 1, 1, 7
2, 2, 2, 3
2, 3, 3, 3
2, 4, 4, 2
2, 5, 5, 7
```

```
@relation exemple_nominal
@attribute User numeric
@attribute ItemID {1,2,3,4,5,6,8 }
@attribute Sequence {1,2,3,4,5 }
@attribute TimeSpent {1,2,3,5,7,8,12 }
@data
1,1,1,5
1,2,2,1
1,5,3,8
1,6,4,12
1,8,5,2
2,1,1,7
2,2,2,3
2,3,3,3
2,4,4,2
2,5,5,7
```

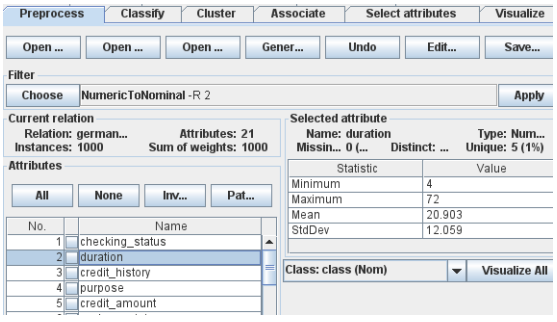
On applique un premier filtre pour transformer ces données numériques en nominales (cf. ci-dessus). Ce qui a donné le fichier *exemple_nominal.arff* (ci-dessus, à droite).
Dans un second temps, il faudra veiller à ce que les données contenu dans ce fichier (la section @data) sont triées selon ID, . Puis on applique le filtre *denormalize* (à gauche de la figure ci-dessous) et on obtient le fichier *exemple_final.arff* (à droite).


```
@relation exemple_final
@attribute User numeric
@attribute ItemID_1 {f,t }
@attribute ItemID_2 {f,t }
@attribute ItemID_3 {f,t }
@attribute ItemID_4 {f,t }
@attribute ItemID_5 {f,t }
@attribute ItemID_6 {f,t }
@attribute ItemID_8 {f,t }
@attribute Sequence_1 {f,t }
@attribute Sequence_2 {f,t }
@attribute Sequence_3 {f,t }
@attribute Sequence_4 {f,t }
@attribute Sequence_5 {f,t }
@attribute TimeSpent_1 {f,t }
@attribute TimeSpent_2 {f,t }
@attribute TimeSpent_3 {f,t }
@attribute TimeSpent_5 {f,t }
@attribute TimeSpent_7 {f,t }
@attribute TimeSpent_8 {f,t }
@attribute TimeSpent_12 {f,t }

@data

1,t,t,f,f,t,t,t,t,t,t,t,t,t,f,t,f,t,t
2,t,t,t,t,t,f,f,t,t,t,t,t,f,t,t,f,f,f
```

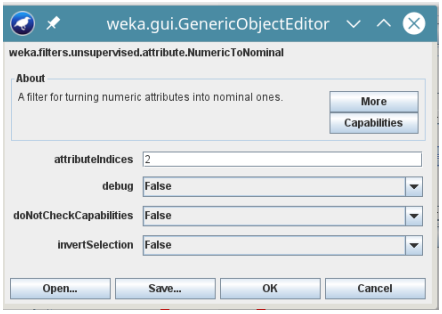
Pour pouvoir appliquer une méthode telle que *A Priori* (règles d'association), il faudra se "débarrasser" du premier attribut qui est numérique (l'ID). Il suffit pour cela de sélectionner, dans l'onglet *Preprocess*, l'attribut *User* (cocher la case) puis faire "remove". Maintenant, on peut appliquer *A Priori* et obtenir les règles d'association.

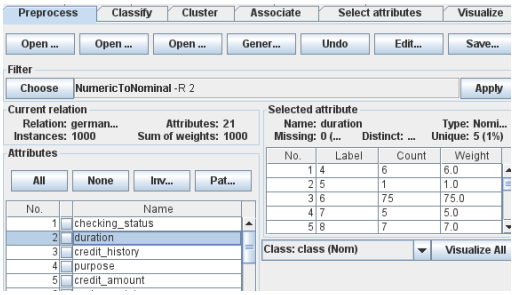
5.3 Conversions numériques vers nominales

- Charger le fichier de données *credi-g.arff*.
- Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/attribute* et dans la liste proposée, choisir **NumericToNominal**.
- Utiliser le bouton **More** pour en savoir davantage sur cette commande.
- En particulier, ici, on remarquera :
  - attributIndices : le numéro (d'ordre) de l'attribut concerné. On choisira pour cet exemple le 2e attribut *duration* qui est numérique (v. fig ci-contre).
    On peut également spécifier une plage de numéros d'attributs. Si vous voulez transformer tous les attributs numériques en nominaux, conserver la valeur par défaut *first-last* proposée par Weka qui appliquera la méthode à tous les attributs numériques.
  - `invertSelection` : inverser le sens de la sélection : au lieu de transformer les attributs sélectionnés, transformer les autres.



Ci-dessous, les options de la méthode pour notre exemple (à gauche) et le résultat de l'application de la méthode (à droite) : l'attribut numéro 2 (*duration*) est devenu nominal.





☞ Pour réaliser le même effet, exécuter la commande suivante (sous Linux/Mac) dans le répertoire où il y a weka.jar :
java -cp ./weka.jar weka.filters.unsupervised.attribute.NumericToNominal -R 2-last -i exemple_num.arff >exemple_no

5.4 binéairisation des données

NominalToBinary : Transfromation des données catégorielles en binaire : on n'aura que des valeurs 0 ou 1 pour les attributs.

On souhaite donc transformer les attributs nominaux en binaire. Si un attribut a par exemple trois valeurs énumérées (soit v1, v2 et v3), pour ces 3 valeurs, 3 nouveaux attributs seront crées dont les noms seront "=v1", "=v2" e "=v3". Pour toute instance, un seul de ces trois attributs sera vrai ("1"), les deux autres seront =0.

5.5 Autres méthodes de préparation / conversion de données

• Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance*. Dans la liste des méthodes d'échantillonnage proposées, on peut noter :

- **Datafly** : anonymisation des données (pour ne pas pouvoir identifier des informations éventuellement sensibles telle qu'un nom, etc...
- Voir les autres ....
- ...

5.6 Autres filtres suivant les attributs

Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/attribute*, on trouve les filtres (importants) suivants

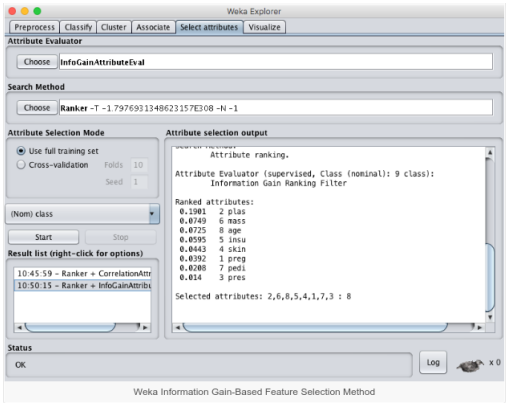
- **ClassOrder** : modifier l'ordre d'apparition des classes dans la BD.
- **MergeNominalValues** : pour fusionner certains attributs nominaux (catégoriels).
- **PLSFilter** : exécute la méthode *Partial Least Square Regression* sur les instances et produit une matrice de prédiction appelée *matrice beta*. Par défaut, les numériques seront centrés.

On développe quelques uns de ces filtres ci-dessous.

6 Choix des attributs : Feature Selection

Dit également *Feature Selection* ou classement des attributs (sans forcément passer par une méthode d'apprentissage. La corrélation trouvée est entre un attribut et la classe.
En général, les indices proches de +1 et -1 nous intéressent. A contrario, les indices de corrélation proches de zéro seront écartées.
On doit préciser une méthode de recherche et un évaluateur.
Ci-dessous, la méthode de recherche de Weka est par défaut *RankerSearchMethod*.

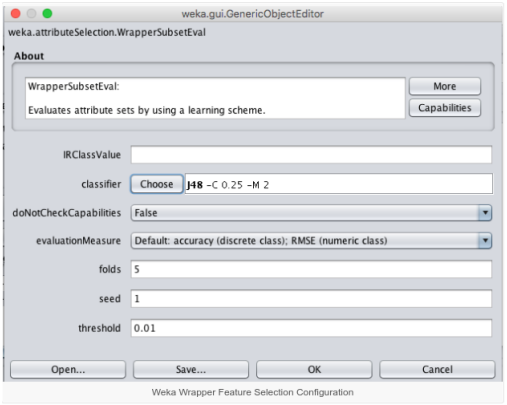
- Sélection basée sur la corrélation de Pearson : le filtre *CorrelationAttributeEval*
- Sélection basée sur le Gain d'information (cf. ID3 du BE1). Sur une BD, on verrait l'attribut que l améthode ID3 mettrait à la racine de son arbre de décision.



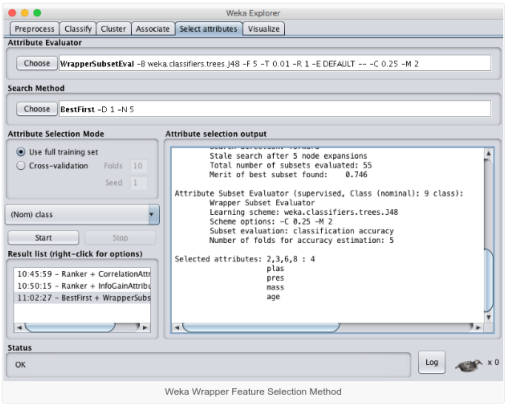
Weka Information Gain-Based Feature Selection Method

- Sélection basée sur le ration de Gain d'information (cf. C4.5 du BE1 qui s'appelle J48 sous Weka).

Aux côtés de ces filtres, on peut procéder à la sélection (classement) des attributs à l'aide de méthode d'apprentissage.
Il faudra choisir le filtre *WrapperSubsetEval* qui devrait utiliser la méthode de recherche *BestFirst* puis lui donner la méthode que l'on veut utiliser (tel que J48). Ci-dessous, les résultats pour la BD. *Pima Indians* :



Weka Wrapper Feature Selection Configuration



Weka Wrapper Feature Selection Method

7 Échantillonnage des données

Parfois, la BD est trop large et l'application d'une méthode peut prendre beaucoup de temps (cf. BD **adults**). Dans ce cas, on peut procéder à un re-échantillonnage correct (stratifié). Suivez les étapes suivantes :

- On va échantillonner 10%
- Dans l'onglet "preprocess", choisir "filter/supervised/instances/Resample"
- Cliquer sur apply
- Choisir 10%, les autres options par défaut
- Sauvegarder éventuellement le résultat

➛ Vous pouvez choisir l'échantillonnage avec ou sans remise.
➛ L'option "biasToUniformClass" : Si 0, on considère la *classe* telle qu'elle est. Si 1, Weka s'arrange pour que la distribution de la valeur de la *classe* soit uniforme dans l'échantillon.

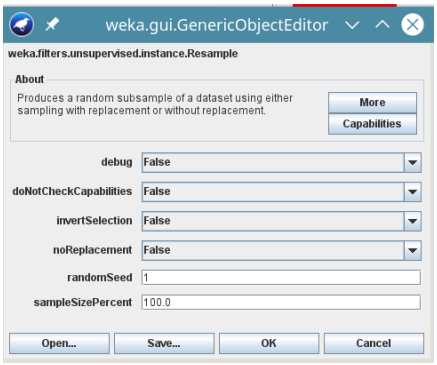
On peut obtenir les mêmes résultats via "filter/unsupervised/instances/StratifiedRemoveFolds" pour avoir 10% des données.

7.1 Échantillonnage de x%

Pour procéder à l'échantillonnage de x% des données :

◦ Charger le fichier de données *banque-app.arff*.
◦ Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance* et dans la liste proposée, choisir **Resample**.
◦ Utiliser le bouton **More** pour en savoir davantage sur cette commande.

En particulier, ici, on remarquera :
◦ `noReplacement` : échantillonnage avec **remise** (par défaut). Mettre cette valeur à True pour un échantillonnage sans remise.
◦ `sampleSizePercent` : le pourcentage à échantillonner. La valeur par défaut de 100% permet simplement de brasser les instances (si échantillonnage sans remise). Par contre 100% avec remise produira une nouvelle base de données en mémoire où certaines instances seront répétées. Ce qui peut être néfaste dans certains cas.
◦ `invertSelection` : inverser le sens de la sélection : les x% échantillonnés seront écartés et on conserve le reste : $(1-x)$%.
◦ `randomSeed` : par défaut =1. Le *seed* est la *graine* utilisée pour initialiser une séquence aléatoire. La variation de cette valeur permet de diversifier les x% échantillonnés. En théorie, deux tirages dans les mêmes conditions avec la même graine produisent la même séquence de x%.



Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance*, on peut dispose également de :

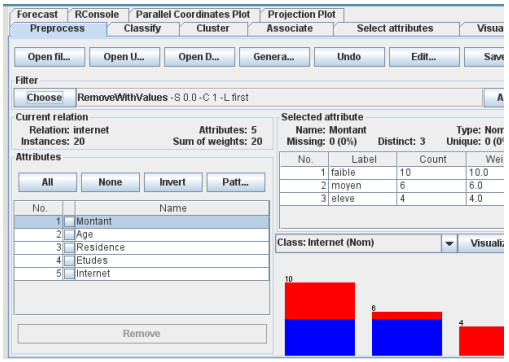
- **SpreadSubsample** : échantillonnage aléatoire. Au besoin, charger le package correspondant.
  On peut spécifier le maximum de "spread" (répartition de la classe) entre la valeur la plus commune de la classe et celle la plus rare. On peut par exemple demander un "spread" de 2 :1 pour avoir deux fois plus souvent la classe la plus commune.
  ➛ Ce paramètre ("distributionSpread") sera donc : 0 = rien changer ; 1 = distribution unifome des valeurs de la classe dans l'échantillon ; 10 = on veut voir la classe la plus commune de se répéter au maximum dix fois plus que la plus la plus rare. L'option "maxCount" permet de contrôler le nombre d'occurrence maximum d'un classe (0 = sans limite.)
- Voir les autres méthodes concernant les instances.

7.2 Échantillonnage selon valeur d'attribut

Nous utilisons ici la BD *banque-app.arff* pour illustrer ces exemples.

Sous WEKA, on peut supprimer des instances de la DB suivant la valeur d'un attribut. Dans l'exemple ci-dessous appliqué à *banque-app.arff*, on se donne l'objectif d'éliminer les instances dont la valeur de l'attribut **Montant** est égale à **Faible**.

◦ Charger le fichier de données *banque-app.arff*.
◦ Dans l'onglet *Preprocess*, sous *Filter*, cliquer sur le bouton **Choose**, puis sélectionner *filters/unsupervised/instance* et dans la liste proposée, choisir **RemoveWithValues**.
◦ Comme pour toute commande, dans la fenêtre qui s'affiche, le bouton **More** nous dit davantage sur cette commande.



En particulier, ici, on remarquera :

◦ `attributeIndex` : l'attribut qui servira de filtre. Pour nous, ce sera **1** (premier attribut = **Montant**). Cette information est visible dans l'onglet *Preprocess* (où on est actuellement), dans la zone des attributs de la base de données *banque-app.arff* chargée. Voir figure ci-contre.

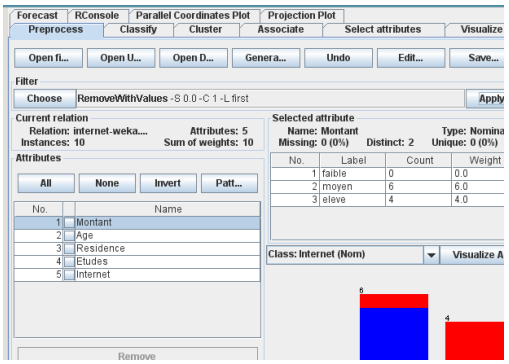
◦ `nominalIndices` : la rang de la valeur utilisée pour la sélection sur les attributs nominaux (catégoriel, non numérique). Cette valeur va de *firts* à *last* et dans notre cas, ce sera *first* car **Faible** est la première valeur de l'attribut *Montant* (ce rang est visible dans l'onglet Preprocess, si vous sélectionner l'attribut *Montant*). Voir figure ci-contre.

◦ `splitPoint` : un seuil numérique à utiliser pour les attributs numériques. Les instances dans lesquelles la valeur de cet attribut est inférieure à ce seuil seront sélectionnés.

◦ `invertSelection` : inverser le sens de la sélection : au lieu de supprimer les instances qui satisferont nos choix, on les garde et on supprime les autres.

◦ `matchMissingValues` : si True, les valeurs manquantes seront considérées comme satisfaisant le filtre. Cette option est indépendante du choix `invertSelection` ci-dessus.

◦ `modifyHeader` : si True, et si l'attribut en question est nominal (*Montant* dans notre exemple), alors enlever toute trace de la valeur **Faible** de cet attribut dans kl'entête de la base de données (cf. le format des fichiers arff de WEKA). Si tel est notre choix, la BD. sera modifiée comme si la valeur *Faible* de l'attribut *Montant* n'avait jamais existé.



Après avoir appliqué ce filtre, on se retrouve avec seulement 10 instances (contre 20 dans la BD initiale, voir figure) : les instances dont Montant=Faible ont été supprimées.

7.3 supprimer les instances avec "missing values"

Comment supprimer les instances avec missing values ?
Un exemple :

1. Charger la BD. *weather-nominal.arff* et dans l'onglet "Preprocessing", noter le nombre d'instances (14) puis choisir le bouton "edit" pour éditer la base de données,
2. Remplacer dans une des colonnes (sauf la colonne de décision "play") 3 valeurs prises au hasard par rien (missing value). Pour insérer "rien", cliquer sur une des cases par exemple dans la colonne "Temperature" et choisir aucune valeur !,
3. Cliquer sur "OK" pour sauvegarder vos changements,
   ➛ La BD. sur le disque n'est pas touchée, seule la BD. en mémoire est modifiée,
4. Toujours dans "PreProcessing", dans la partie "filtre", cliquer sur "choose" et choisir "Multifilter" qui est disponible immédiatement (sans rentrer dans aucun dossier),
5. Une fois "multifilter" choisi, cliquer dans la zone de la commande et aller en bas (la 3e zone) : "filtres" (qui contient par défaut "2 weka.filters.Filter") et cliquer dessus,

6. Dans la petite fenêtre qui apparaît, par défaut, "AllFilter" est inséré,
7. Cliquer dans cette petite fenêtre sur "Choose" au dessus pour spécifier notre filtre,
8. On arrive dans la fenêtre des paramètres de ce filtre qui est la fenêtre de choix de filtres de Weka. Choisir le filtre "Unsupervised/instances/RemoveWithValue" puis faire "Add".
9. A partir de maintenant, à chaque fois qu'on clique sur "Add", le même filtre sera inséré dans la petite fenêtre,

   ➛ Pourquoi referaire "Add" ? Parce que chaque filtre va concerner une colonne et donc si on veut enlever toutes les instances avec des valeurs manquantes dans une colonne différente on doit faire "Add" autant de fois. Par exemple, si vous avez créé des valeurs manquantes dans 3 colonnes différentes, vous devez faire "Add" 3 fois.
10. On édite le 1er filtre en cliquant sur "RemoveWithValue" et dans la fenêtre des paramètres du filtre qui s'affiche, faire les modification suivantes :

    attributeIndx : mettre l'indice de la colonne concernée : p. ex. on met 3 car on a introduit des missing values dans la colonne 3,

    - debug True si on veut
    - inversSelection : True (important)
    - matchMissingVAlue : True (important)
    - laisser le reste
11. Cliquer sur OK puis fermer la petite fenêtre puis OK dans la dernière fenêtre encore ouverte.
12. Maintenant, appuyer sur "Apply" en face du "Filter",
13. Le filtre s'applique, les instances avec des valeurs manquantes sont supprimées, ce qui peut être contrôlé en suivant le nombre d'instances dans la BD relevé au début.
14. Il faut répéter l'étape 10... 12 pour chaque colonne concernée. Weka n'accepte pas qu'on donne plusieurs valeurs dans le champ "attribueIndex" Et donc on doit répéter autant de fois ces actions que de colonnes concernées.
15. A la fin, il nous reste seulement les instances sans missing value.

8 Comparaisons des évaluations

Nous étudions quelques indices puis montrons une méthode (empirique mais courante) de comparaisons de différents modèles obtenus sur une même BD.

- Matrice de confusion (cas bi-classes)
- Matrice de confusion (cas n-classes)
- Indice kappa
- La signification des erreurs affichées par Weka
- Évaluation des modèles

Voir également le cours.

9 Évaluation des modèles : quelques éléments

☞ Voir ci-dessous comment (empiriquement) comparer plusieurs modèles obtenus pour une BD. Nous verrons une méthode de comparaison statistique au BE2.

La comparaison passe toujours par celle de différents indicateurs. Nous ferons des comparaison simple ci-après.
Les méthodes plus élaborées (voir BE2) utiliseront des tests statistiques.

9.1 Matrice de Confusion de classification non binaire

Nous avons utilisé ci-dessus la matrice de confusion d'une classification binaire. Voyons comment faire pour les autres cas.

Suivant la méthode appliquée, on doit adapter certaines mesures au modèle obtenu. Par exemple, dans le cas de 3 classes, la valeur **TN** (*True Negative*) ne s'obtient pas directement.

Pour un cas bi-classes, les valeurs TP, TN, FP et FN sont directement accessibles. Par contre, pour une classification avec plus de 2 classes, comment procéder ?

• Exemple des sorties d'un modèle avec les codes couleurs :
➛ noire : Bien classés, orange : Chats pris pour autre chose,
magenta : Chiens pris pour autre chose, bleu : Loups pris pour autre chose.

		Actuels (B.D.)		
		Chat	Chien	Loup
Prédictions	Chat	5	2	0
	Chien	3	3	2
	Loup	0	1	11

• Constat : sur 8 Chats, 5 ont été bien classés (TP) et 3 ont été pris pour des Chiens.
• Parmi les 6 Chiens, 3 ont été bien classés (TP) et 2+1=3 autres non.
• Parmi les 13 loups, 11 ont été bien classés (TP) et 2 autres non.
Calcul de TP,TN,FP et FN pour le **Chat** (suivre les couleurs) :

• Rappel de la table de confusion des 3 classes :

		Actuels (B.D.)		
		Chat	Chien	Loup
Prédictions	Chat	5	2	0
	Chien	3	3	2
	Loup	0	1	11

• Considérez *Chat* × *Chien* mais on a besoin des non-Chats (= Chien + Loup)

	Chat	**Chien**	
Chat	TP=5 Chiens	FP= 2 + 0	← Positifs (Modèle)
Chien	FN= 3 + 0	TN= 3+11 + 1+2 =17	← Négatifs (Modèle)

• Il y a dans cette table :
  ◦ 5 Chats bien classés (TP) et 3 des (vrais) Chats sont pris pour des Chiens (FN)
  ◦ 2 Chiens Pris pour Chats (FP)
  ◦ Les TN pour notre matrice (on prend Chats comme repère) :
    ➛ 11 Loups et 3 Chiens sont bien classés (donc TN pour Chats),
    ➛ 2 Loups sont pris pour des Chiens (TN pour Chats) car ils ne sont pas pris pour des Chats ; 1 Chien est pris Loup (TN pour la même raison.)

• De même pour le *Chien* et le *Loup* : on crée les matrices de confusion 2 à 2.

La mesure kappa ci-dessous est l'un de ces indicateurs.

9.2 TP/TN/FP/FN si nb_classes >2

Si vous travaillez avec une matrice de confusion pour une BD. dont le nombre de classes dépasse 2, les différentes valeurs deviennent :

C1	*C2*	*C3*	*C4*	← **Classes obtenues par le modèle**
TP(C1)	FN(C1)	FN(C1)	FN(C1)	*C1 (vraie C1, dans la BD.)*
FP(C1)	TP(C2)			*C2*
FP(C1)		TP(C3)		*C3*
FP(C1)			TP(C4)	*C4*

1. **TP(C1)** est le nombre d'instances bien classées dans la classe *C1* (conforme à la BD.). De même pour TP(C2), TP(C3) et TP(C4) sur la diagonale.
2. En 1ère colonne, nous avons FP(C1) qui donne les instances classées en C1 par le modèle mais qui sont de la classes C2 (resp. C3 et C4 en colonne) dans la BD.
3. Sur la première ligne, outre TP(C1), nous avons FN(C1) les instances qui devraient être classées en C1 mais qui ne le sont pas (elles ont été classées par erreur dans C2,C3,C4 par le modèle).
4. Sur la diagonale, les instances TP(C2), TP(C3) et TP(C4) (la somme de ces trois) sont des instances (dans leurs bonnes classes par le modèle) qui ne sont pas mises dans C1 : donc ce sont les TN(C1).
5. On peut procéder classes par classe : sur une ligne, pour la classes x, on aura les FN(x), sur une colonne, on aura les FP(x) et sur la diagonale (principale) les TN(x).
6. Un raisonnement similaire sur la seconde diagonale est possible.
7. Notons qu'une case de cette matrice peut donc représenter plusieurs mesures : $\{T, F\} \times \{P, N\}$.

9.3 A propos des mesures Kappa

Kappa (Chance corrected agreement) : voir également cours chapitre 4 part 2.

• Est comme un coefficient de corrélation ; on l'utilise pour la similarité et la fiabilité des résultats.

• Plusieurs références (où hypothèses) sont possibles dans ces calculs dont les résultats dépendent de la prévalence et du biais (v. + loin)

☞ Un $kappa > 0$ veut dire : le modèle appris fait mieux que la chance (pile ou face), $kappa > 0.6$ est préférable

☞ $Kappa \leq 0$: désaccord (ou accord seulement au hasard)

☞ $Kappa = 1$: max d'accord

☞ Observez toujours kappa avec e.g. l'aire sous ROC. Préférer les mesures pondérées

Kappa pour la table de contingence suivante :

$$K = \frac{(O_{ag} - E_{ag})}{(1 - E_{ag})}$$

O_{ag} : observed agreement (diag / total) = **Accord**
E_{ag} : expected agreement (expected in diag / total) = **Hasard**
➛ = la probabilité d'un accord aléatoire

$O_{ag} = 57/91 = .63$

	ref std A	ref std B	ref std C	total
system A	13(6.6)	4	6	23
system B	8	23 (13.1)	2	33
system C	5	9	21 (11.3)	35
Total	26	36	29	91

$13 + 23 + 21 = 57$ les valeurs sur la diagonale

$E_{ag} = 31/91 = .34$ $6.6 + 13.1 + 11.3 = 31$ les valeurs entre parenthèse (attendus = expected) sur la diagonale

$K = (.63 - .34)/(1 - .34) = 0.43$

☞ Les détails de calculs de cet exemple sont donnés dans l'exemple suivant.

Un autre exemple : ici, les valeurs attendues ne sont pas données → on se réfère au calcul du *Hasard*.

	Ref. Std +	Ref. Std -
Système +	25	0
Système -	50	25

Pour trouver Kappa, on calcule :

1- **Accord** : proportion des données sur laquelle les deux sont d'accord
= (TP+TN)/total BD (c-à-d., (la diagonale) / (taille BD))

2- **Hasard** : $(\sum_i (total\ ligne\ i * total\ col\ i))/total\ BD\ au\ carre$

Ici, on a :

pour 1 : $\frac{25+25}{100} = 0.5$

pour 2 : $\frac{(25 * 75(lig/col\ 1)) + (75 * 25(lig/col\ 2))}{100 * 100} = 0.375$

➙ Ce qui donne : $kappa = \frac{0.5 - 0.375}{1 - 0.375} = 0.2$

Pour la première table précédente :

Pour calculer E_{ag} (le Hasard), les valeurs entre parenthèses étaient données pour chaque i mais on peut les retrouver :

Hasard : $(\sum_i (total\ ligne\ i * total\ col\ i))/total\ BD\ au\ carre$

Ici : $E_{ag} = \frac{(23 * 26) + (33 * 36) + (35 * 29)}{(91 * 91)} = 0.338 \sim 0.34$

Notes sur la mesure Kappa pour 2 avis (Kappa de *Cohen*)

$$\frac{P(accord\ relatif\ des\ deux) - P(accord\ par\ hasard)}{1 - P(accord\ par\ hasard)}$$

➙ $1 - P(accord\ par\ hasard)$ représente le maximum d'accord possible.

Kappa mesure l'accord entre deux modèle (2 codeurs)

☞ Dans le cas des méthodes d'apprentissage, les 2 avis sont ceux exprimés dans la matrice de confusion où l'accord relatif des 2 modèles = $TP + TN$

➙ Kappa = (la différence entre l'avis d'un modèle et le hasard) divisée par (1- hasard).
➙ $Kappa = 1$: max d'accord
➙ Kappa ≤ 0 : désaccord (ou accord seulement au hasard)
➙ $Kappa \geq 0$: le modèle fait mieux que le hasard (pile ou face).
➙ Moins il y a des classes, plus $Kappa$ sera grand
➙ Si plus de 2 avis : prendre Kappa de *Fleiss* (cf. cas général ci-dessus)

***Kappa* et les règles d'association** (nous sera utile au BE2) :

Pour une règle (ou l'itemset ={A,B}) $A \Rightarrow B$, on a : $Kappa = \frac{P(AB) + P(\overline{A}\overline{B}) - P(A)P(B) - P(\overline{A})P(\overline{B})}{P(A) + P(B) - P(A)P(B) - P(\overline{A})P(\overline{B})}$

Dans une autre formulation plus simple (sans tenir compte de $\overline{A}$ et $\overline{B}$), on a :

$$Kappa = \frac{P(AB) - P(A)P(B)}{P(A) + P(B) - 2P(A)P(B)}$$

Où $P(A)P(B)$ est la probabilité théorique de A et de B en l'absence de toute hypothèse (de dépendance ou d'indépendance) inspirée d'un processus général de Bernoulli, et qui représente donc E_{ag} dans $K = \frac{(O_{ag} - E_{ag})}{(1 - E_{ag})}$

☞ Un cas particulier (où l'absence de $\overline{A}$ et $\overline{B}$ est justifié) est une BD. où on a forcément A ou B dans chaque instance auquel cas $P(A) + P(B) - P(A)P(B) = 1$ et nous pouvons retrouver la forme originelle de Kappa.

Kappa et la matrice de confusion : soient les données d'un modèle :
Kappa statistic *0.3108*
ROC : *0.709*

• Et la matrice de Confusion :

a	b	<−− classifié comme
59	2	— a = 0
27	12	— b = 1

• Ici, pour 100 instances, on a $TP + TN = 59 + 12 = 71, FP + FN = 27 + 2 = 29$.

• Le pourcentage de correctement classés = justesse simple (accuracy).
➞ Son inconvénient est que la valeur n'est pas pondérée par le hasard (*chance corrected*) et n'est pas **sensible à la distribution** des classes.
➞ Dans ce cas, l'aire ROC est un bon complément (ici, 0.709 pour les 2 classes).

• Kappa est pondérée et mesure l'accord entre le modèle et la BD (d'apprentissage).
☞ $Kappa \geq 0$: le modèle fait mieux que le hasard (pile ou face). En statistiques, certains considèrent que Kappa est exploitable seulement à partir de 0.6 (ou 0.7) ; en deçà, Kappa dit peu de chose.

Autres remarques :
Les taux d'erreurs numériques (données par Weka) comme *Mean absolute error, Root mean squared error, Relative absolute error, Root relative squared error* sont plus utiles à la prédiction numérique qu'à la classification.
➞ Les prédictions numériques ne sont pas justes ou erronées mais leur erreur a une certaine magnitude reflétée par ces valeurs d'erreur.

9.4 A propos des erreurs affichées par Weka

Vous avez remarqué que Weka affiche des statistiques dont quelques mesures d'erreur expliquées brièvement ci-dessous.

Soit y la classe (la "vraie") dans la BD et $\hat{y}$ son estimation à calculer par un modèle.

- **Corrélation** : exprime combien $y = \{y_1, ..., y_N\}$ et $\hat{y} = \{\hat{y}_1, ..., \hat{y}_N\}$ sont liés. Sa valeur est dans $[-1, 1]$ et 0 veut dire qu'ils n'ont pas de relation, 1 veut dire que cette relation est forte et -1 dira qu'il y a une relation linéaire inverse entre eux (une grande valeur de y indique une petite valeur de $\hat{y}$ et vice versa).
  On peut illustrer ce propos par la figure ci-dessous :

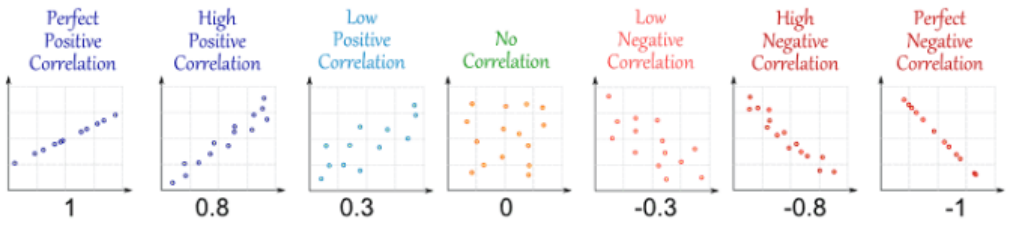


FIGURE 1 – *(source : http ://www.mathsisfun.com/data/correlation.html)*

- **Mean absolute error** : $MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$
- **Root mean square error** : $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$

- **Relative absolute error** : $RAE = \dfrac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{\sum_{i=1}^{N} |\overline{y} - y_i|}$

  où $\overline{y}$ est la moyenne des valeurs de la classe y dans la BD. Notons que dans le cas d'une classe non numérique, cette moyenne est calculée à l'aide de la fréquence de chaque classe.

- **Root relative squared error** : $RRSE = \sqrt{\dfrac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N} (\overline{y} - y_i)^2}}$

- On peut bien entendu ajouter d'autres erreurs / mesures (affichées ou pas) par Weka telles que l'erreur moindre carrée MSE du $RMSE$ ci-dessus, la mesure Coverage, ...) évoquées ci-dessus pour certaines.

Remarques :
∘ La présence du carrée $((..)^2)$ souligne l'importance et l'influence des valeurs extrêmes (voire les anomalies / *outliers*) dans ces erreurs (sans évoquer le fait que ces erreurs seront ≥ 0 grâce à $(..)^2$).
Elle permet également de présenter algébriquement la définition même de la variance : $\mathbb{E}[x^2] - \overline{x}^2$.
Enfin, la minimisation d'une *distance* Euclidienne (de la forme $(A-B)^2$) fait intervenir la dérivée qui nous permet de manipuler ensuite une forme linaire (dans notre cas) ; ce qui simplifie les calculs d'un extremum.

∘ La présence de la racine carrée sur $(.)^2$ nous permet de retourner à l'échelle (linéaire pour un attribut / une classe) des données et répondre à un inconvénient de $(.)^2$ qui est de ne pas être dans la même "unité" que les données, la décision (classe) y en particulier.

∘ Dans le cas de MAE et de $RMSE$, on travaille simplement avec la moyenne des différences dans la même "unité" que la classe y.

∘ Dans le cas de RAE et de $RRSE$, on divise "les différences" (les numérateurs) par la variation de y. La présence de $\sum_{i=1}^{N} (\overline{y} - y_i)^2$ ou $\sum_{i=1}^{N} |\overline{y} - y_i|$ nous dit combien y varie par rapport ("relativement") à sa moyenne ; autrement dit : combien y varie par rapport à "lui-même" (comparé à la variance).

☞ Ne faisons pas l'erreur de penser que cette division (par les dénominateurs $\sum_{i=1}^{N} (\overline{y} - y_i)^2$ ou $\sum_{i=1}^{N} |\overline{y} - y_i|$) "normalise" (place dans $[0, 1]$) la mesure en question car les cas (sous Weka) où $RAE > 100\%$ ne sont pas si rares !

9.5 Comparaison des modèles / méthodes

Pour comparer des modèles obtenus via différentes méthodes, les choix sont :
∘ Weka propose un outil de comparaisons (dans "experimentation", cf. test d'*étudiant*) avec quelques limitations.
Voir BE2 (ou demander à l'enseignant si vous êtes impatient !).
∘ On peut procéder soit-même à ces comparaisons. C'est ce que nous ferons pour ce BE où nous comparons simplement (linéairement) les indicateurs produits par Weka.

☞ Si vous ne voyez pas ces mesures s'afficher dans les sorties, cliquez sur **"More Options"** (au dessus du bouton **"Start"**) puis dans la fenêtre qui s'affiche, cliquer sur **"Evaluation metrics..."** et cochez les mesures que vous voulez voir afficher.

☞ Pour l'interprétation des erreurs affichées par Weka, voir ci-dessous (section 9.4 page 16).

Pour ce faire, supposons avoir appliqué 4 méthodes à une BD. et avons obtenu 4 modèles. Établir un tableau comme ci-dessous (voir après ce tableau pour quelques explications) :

Mesures Modèles →	M_1	M_2	M_3	M_4	Remarques
TP % (True Positive)	TP_1	TP_2	TP_3	TP_4	simple % de TP sur $\|BD\|$
FP % (False Positive)	FP_1	FP_2	FP_3	FP_4	simple % de FP sur $\|BD\|$
Precison $= \frac{TP}{TP+FP}$	Pr_1	Pr_2	Pr_3	Pr_4	☞ le nombre de TP, pas son %
Recall $= \frac{TP}{TP+FN}$	Rc_1	Rc_2	Rc_3	Rc_4	équivalent à *"TP rate" = "sensitivity"*
Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$	Acc_1	Acc_2	Acc_3	Acc_4	TN=True Negative, FN=False Negative
Specificity $= \frac{TN}{TN+FP}$	SP_1	SP_2	SP_3	SP_4	
F-Measure $= \frac{2*Recall*Precision}{Recall+Precision}$	F_1	F_2	F_3	F_4	
FP rate $= \frac{FP}{TN+FP}$	Fr_1	Fr_2	Fr_3	Fr_4	Cas idéal : $FP\ rate = 0$, ici 2 ex aequo
Roc		best			
Kappa			best		
Coverage of cases (0.95 level)		best			
Mean rel. region size (0.95 level)			best		
Mean absolute error			best		à partir d'ici, le best est le minimum
Relative absolute error				best	le best est le minimum
Root relative squared error		best			le best est le minimum
Root mean squared error	best		best		le best est le minimum
Comptes	4	5	6	3	← M_3 semble la meilleure.

☞ La quasi toutes ces mesures sont affichées par Weka, dans la fenêtre des résultats à droite.

• **TP %** (ou *TP rate*) affiché par Weka est une moyenne qui correspond aux *instances correctement classées* (ou l'*accuracy* = *TP+ TN*) dans différentes classes. Remarquez que *TN* pour une classe est *TP* pour les autres.
Par contre, pour FP%, observez les indications de Weka.
➛ Si vous voulez une comparaison plus complète, appliquez la méthode de construction (donnée ci-dessus) de la matrice de confusion pour un cas non binaire et reportez FN et FP dans le tableau de comparaison.

• PRC area : l'aire sous Precision-Recall-Curve (voir ci-dessous).

• La *surface Roc* (voir cours) est une valeur dans l'intervalle [0,1]. Plus elle est proche de 1, mieux c'est. Le hasard (la droite f(x)=x) représente une surface de 50% pour un cas bi-classe.

• Signaler ensuite la meilleure (best, en blue ci-dessus) de chaque mesure puis choisir le modèle qui maximise le nombre de "best".

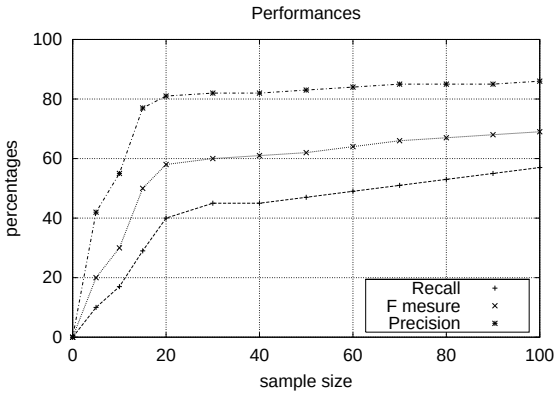
• Nous étudierons toutes ces mesures dans les cours suivants.

Rappel (voir cours) de quelques unes de ces mesures et un ex. d'aire sous les courbes *Precision*, *Recall* **et** *F-mesure* :

$$\textbf{Recall} = \frac{\#\ relevants\ extraits\ (TP)}{total\ relevants\ (TP\ +\ FN)}$$

$$\textbf{Precision} = \frac{\#\ relevants\ extraits\ (TP)}{total\ extraits\ (TP\ +\ FP)}$$

$$\textbf{F-mesure} = \frac{2 \times recall \times precision}{recall + precision}$$



➛ L'aire sous le courbe Precision ou celle de Recall est à droite.
Pour la PRC, la courbe est produite dans un système de coordonnées avec en Axe des Xs la *Precison* et en Ys le *Recall*.
Le repère inverse est également possible.

☞ Sous Weka, cliquer droit sur le modèle appliqué (cadre gauche) pour visualiser différentes courbes (ROC, Precision, Recal, Lift, etc...). Pour la courbe ROC, cliquer droit et choisissez *threshold curve*. Dans la fenêtre proposée, dans la zone "Couleur", choisissez différentes courbes (ROC, Precision, recall, Lift, ...).
Vous pouvez également visualiser le *cost/benefit curve* de la même manière.

A propos des mesures "Coverage" et "Mean rel. region size" (du tableau ci-dessus) :

• Pour estimer un intervalle de confiance, Weka produit également les mesures **coverage** et **Mean rel. region size** (la moyenne relative de la largeur de l'intervalle de confiance à un niveau de 95%).

Pour obtenir ces mesures, la méthode appliquée doit disposer (sous Weka) de l'estimation de l'intervalle (cf. tout schéma général de *régression*).

• **Mean rel. region size** (ou, le nom sous Weka "sizeOfPredictedRegions") ou la largeur relative est une normalisation des largeurs d'intervalles. Cette normalisation est faite par l'intervalle des valeurs de la classe (attribut cible, voir ci-dessous) dans l'ensemble d'apprentissage (toute valeur relative $\geq 100\%$ est écartée).

• La **couverture** (*coverage*) (de l'ensemble de test) correspond à $\frac{N_{out}}{N_{tot}}$ où N_{out} est le nombre d'instances bien classées (faire aussi attention aux instances non classées, s'il y en a) et N_{tot} est le total des instances. Dans une méthode de clustering, N_{out} est le nombre d'instances dans un cluster (voir cours).

Le (*coverage*) empirique d'un estimateur d'intervalle doit être d'un niveau de confiance $\geq 95\%$. Un estimateur raisonnable est celui qui affiche une couverture $\geq 95\%$ tout en produisant un intervalle "étroit".

• Ces deux valeurs sont calculées également pour une classe nominale si la méthode utilise une prédiction probabiliste. Dans ce cas, un "intervalle" est le plus petit ensemble de valeurs de classe (attribut cible) tel que la probabilité cumulative pour ces valeurs dépasse 95%. Dans ce même cas, la largeur relative d'intervalle est le nombre d'occurrences de la valeur cible (la classe) divisé par toues les différentes valeurs cibles.

• Pour les *règles d'association* (voir BE2), on pourra ajouter à ce tableau d'autres mesures (Lift, Conviction, etc. voir cours Chapitre 4, Part 1.2)

• Pour les particularités de ces mesures adaptées en "extraction d'informations" (IR, cf. Google), voir le complément au chapitre 4, Part1.2.

---

10 Quelques Bases de données (pour les BEs)

☞ Voir également annexes de chaque BEs pour les explications sur les BDs. utilisées dans le BE.

10.1 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

10.2 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	agé	village	non	non
18	élevé	âgé	bourg	oui	non
19	faible	moyen	banlieue	non	oui
20	moyen	jeune	village	oui	oui

L'attribut ternaire *Montant* décrit la moyenne des montants sur le compte client. Le second attribut ternaire *Classe-Age* donne la tranche d'âge du client. Le troisième attribut ternaire *Résidence* décrit la localité de

résidence du client. Le dernier attribut binaire *Etudes* a la valeur `oui` si le client a un niveau d'études supérieures.
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 calculette ou d'un tableur : Classer ces exemples dans leur Clausters.

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