Pourquoi ? C'est un besoin, gain de temps.

Déroulement du module Deux premières semaines : compréhension et fouille de données. Dernière séance : contrôle des connaissances du cours et du mini projet.

1 Qu'est-ce que la fouille de données ?

On veut sortir quelque chose de nos données via des statistiques. Apprentissage automatique via intelligence artificielle (deep learning), beaucoup de données et beaucoup de calculs.

Aucune méthode d'apprentissage ne peut être la meilleure. Chacune a ses avantages.

Durant les années 90, des logiciels ont été développés pour tirer des connaissances des bases de données. La puissance de calcul des machines qui était déjà très bonne.

Data mining, les données sont des matériaux brutes, c'est de la terre, de la roche, il faut extraire, ça ne pousse pas tout seul.

Mais on découvre que ces données ne sont pas terribles, mais ce qui nous intéresse c'est de faire des formes de prédiction via l'apprentissage de règles qui conduit à un modèle qui s'applique à tout les clients. Le deep learning peut donc être de trop.

Le problème est qu'il faut préparer les données. Car elles ne sont pas forcément conçus pour la fouille de données. Mais il faut aussi nettoyer la base de données, qui est polluée de bêtises et de fausse données (le stagiaire doit s'occuper de ça). C'est 80% du temps de travail. Les 20% restants vérifient que le modèle généré est correcte. La génération du modèle est quasi instantané.

On a rarement trop de données.

"Data scientist", c'est la science des données.

1.1 Découvrir des motifs

Un exemple est "l'analyse du panier de la ménagère", on analyse le cadis des gens, via le ticket de caisse des produits achetés en même temps. On regarde les données et on les généralises.

L'objectif du patron du super marché est d'augmenter le pouvoir d'achat. Une solution est d'augmenter le salaires de clients, mais le patron du super marché ne peut pas. Il doit donc trouver d'autres solutions.

Une chaîne de magasin Américaine a découvert que lorsqu'une personne achète des couches pour bébé, elle achète des bières. C'est quelque chose de nouveau, qu'on n'avait connaissance. C'est à la chaîne de prendre une initiative, de quoi faire de cette information.

Bien entendu, on se fiche de savoir qui a acheté les couches pour bébé. On veut juste tirer des règles.

1.2 Définition de l'apprentissage automatique

On dit qu'un programme informatique apprend, à partir d'une expérience E, par rapport à une classe de tâches T et une mesure de performance P, si sa performance sur des tâches de T, mesurée par P, s'améliore avec l'expérience E.

1.3 Exemple introductif

On veut savoir quand est-ce qu'il joue à partir des données ci-dessous, donc, en tirant des règles.

Jour	Ciel	Température	Humidité	Vent	Jouer
J1	Soleil	Chaud	Élevée	Faible	Non
J2	Soleil	Chaud	Élevée	Fort	Non
J3	Couvert	Chaud	Élevée	Faible	Oui
...	...	...	...	...	...

TABLE 1 – Exemple de données à fouiller

1.4 Exemples d'applications réelles

— Aide à la décision (attribution de prêts bancaires, cartes de crédit, etc.)
— Analyse d'images (détection de nappes de pétrole, structures astronomiques, médical, etc.)
— Apprendre à reconnaître des mots parlés
— Apprendre à conduire un véhicule autonome
— Apprendre à jouer au backgammon à un niveau de champion
— Commerce et marketing (panier de la ménagère, ticket de caisse, carte de fidélité, mailing, etc.)

La carte de fidélité est pratique car à la place d'avoir un seul ticket, on a un ensemble de ticket et on incite le client à revenir. On se fiche du nom, on veut juste pouvoir associer les tickets à une carte.

On veut faire plaisir au client, c'est notre rôle.

On collabore au big brother.

2 Différents types d'apprentissages

2.1 Apprentissage supervisé (ce n'est pas interactif)

On a une classe, un attribut de sortie. Il y a une entrée, une sortie.
— La majorité des exemples précédents ;
— On a une classe ;
— Exemples étiquetés par un professeur (par un oracle) ;
— Classe numérique (régression) ou catégorielle (classification).

2.2 Apprentissage non-supervisé

C'est le cas où on n'a pas de classe prédéfinie.

Deux méthodes :

Découverte de règles on veut découvrir des règles, peu importe leur conclusion.

Clustering, on veut pouvoir trier les données par paquet de similarité (qu'on a définie). Permet donc une représentation des données. On crée donc un ensemble de classes à analyser.

3 La fouille de données

3.1 Apprentissage de concepts

Fonction booléenne, Par exemple il pleut, il pleut pas.

Représentation des hypothèses : une règle. Si conjonction d'attributs-valeurs, alors appartient au concept.

Espace des hypothèses, combien de règles on peut avoir. $4 \times 4 \times 3 \times 3 = 144$ conjonctions.

Espace des instances, combien de représentations possibles. $3 \times 3 \times 2 \times 2 = 36$ instances.

3.2 Le principe de l'apprentissage inductif

Toute hypothèse qui est une bonne approximation de la fonction cible sur l'ensemble d'apprentissage est aussi une bonne approximation de la fonction cible sur les instances non observées.

3.3 Parcours de l'espace des hypothèses

— On énumère et on garde ce qui est positif.
— Relation d'ordre.
— Recherche de toutes les hypothèses cohérentes.

3.4 Problèmes

Il reste plusieurs hypothèses cohérentes avec l'ensemble d'apprentissage. Il y a pas assez d'exemple pour isoler la "bonne" hypothèse. Critère supplémentaire pour choisir la "bonne".

Il n'y a plus d'hypothèse cohérente avec l'ensemble d'apprentissage. Soit le langage n'est pas assez expressif, soit les exemples sont bruités (ex. : un jour où il joue pas malgré qu'il fait moche).

3.5 Le bruit en apprentissage

— Origines diverses (physiques, expérimentales, humaines, etc.) ;
— Mauvaise classe associée à la description.

3.6 Pas d'apprentissage sans biais

— $2^{36} = 6.9 \times 10^{10}$ (69MM) concepts possibles.
— En fait, $2^{36-14} = 4096$ concepts cohérents avec les exemples.
— Il est impossible de choisir sans biais supplémentaire.
— Biais de langage.
— Biais de recherche.
— Biais pour éviter le sur-apprentissage.

3.7 La méthodologie de la fouille de données

1. Identifier le problème à résoudre.
2. Préparer les données d'entrée.
3. Explorer des modèles multiples.
4. Utiliser le modèle sur le réel.
5. Suivre le modèle et l'améliorer.

3.8 Identifier le problème à résoudre

3.8.1 Concept

Reformuler le problème en un des types connus.

3.8.2 Entrées

— Attributs.....

4 OLAP

4.1 Fonction d'agrégation

4.1.1 Appel par ligne

Afficher le numéro de ligne de chaque entrée.

```
    SELECT ROW_NUMBER() OVER() AS row_num, order_no, price FROM aroma.orders;
```

```
    ROW_NUM            ORDER_NO   PRICE
-------------------- ----------- ---------
                   1        3602    780.00
                   2        3604    800.66
                   3        3615   4600.10
                   4        3623   4425.00
                   5        3624   4425.00
                   6        3625   3995.95
                   7        3621  10234.50
```

4.1.2 Appel sur l'ensemble des lignes précédentes

`OVER(...)` nous permet d'exécuter une fonction OLAP ou du SGBD sur chaque ligne, plutôt que sur l'ensemble des lignes.

Faire la somme des moyennes au-dessus de la ligne.

```
SELECT SUM(AVG(dollars)) OVER(ORDER BY dollars ROWS UNBOUNDED PRECEDING)
FROM aroma.sales GROUP BY dollars ORDER BY dollars;
```

```
     1.750000000000000000000000000
     3.750000000000000000000000000
     6.000000000000000000000000000
     8.500000000000000000000000000
    11.250000000000000000000000000
    14.250000000000000000000000000
    17.500000000000000000000000000
    21.000000000000000000000000000
```

— `OVER(...)` indique au SGBD que la fonction `SUM` est une fonction d'agrégation OLAP.
— `ROWS UNBOUNDED PRECEDING` permet d'instruire OLAP de prendre les lignes du dessus.
— Le `ORDER BY` d'OLAP (le premier, donc) est obligatoire, car il assure que les lignes d'entrée sont correctement triées. Autrement, les résultats peuvent prendre aucun sens logique.
— Le dernier `ORDER BY` permet de modifier l'affichage des résultats, et donc de distinguer les résultats depuis le `ORDER BY` de OLAP .

4.1.3 Appel sur l'ensemble des lignes précédentes par groupe donné

```
SELECT date, SUM(dollars) FROM aroma.period a, aroma.sales b, aroma.product c
WHERE a.perkey = b.perkey AND c.prodkey = b.prodkey AND c.classkey = b.classkey
GROUP BY week, date ORDER BY week, date;
```

Permet de faire la somme des ventes sur une journée.

```
 SELECT date, SUM(dollars), SUM(SUM(dollars)) OVER(PARTITION BY week ORDER BY date ROWS
     UNBOUNDED PRECEDING) FROM aroma.period a, aroma.sales b, aroma.product c WHERE
```

```
    a.perkey = b.perkey AND c.prodkey = b.prodkey AND c.classkey = b.classkey GROUP BY
    week, date ORDER BY week, date;

```

```
DATE      2                               3
---------- ------------------------------- -------------------------------
01/02/2004                         8851.70                         8851.70
01/03/2004                         8039.00                        16890.70
01/01/2005                         8988.60                        25879.30
01/01/2006                        11529.50                        37408.80
01/02/2006                         8795.00                        46203.80
01/03/2006                         7825.20                        54029.00
01/04/2006                         8849.50                        62878.50
01/05/2006                         9964.10                        72842.60
01/06/2006                         7788.50                        80631.10
01/07/2006                         9598.90                        90230.00
01/04/2004                         7828.35                         7828.35
01/05/2004                         9036.95                        16865.30
01/06/2004                        10089.70                        26955.00
01/07/2004                         6798.75                        33753.75
```

`PARTITION BY` nous permet de faire la somme des lignes précédentes et se remet à zéro dès que la semaine change.

4.2 Opérations arithmétiques

4.2.1 Opérations

Les opérations standards sont supportées (*, /, +, -, (,)).

Prix moyen de chaque produit par vente

```
SELECT date, SUM(dollars) as TotalSales, SUM(quantity) AS TotalQ,
   SUM(dollars) / SUM(quantity)
FROM aroma.period a, aroma.sales b, aroma.product c
WHERE a.perkey = b.perkey AND c.prodkey = b.prodkey AND c.classkey = b.classkey
GROUP BY week, date ORDER BY week, date;
```

```
DATE      TOTALSALES                      TOTALQ     4
---------- ------------------------------- ----------- -------------------------------
01/02/2004                         8851.70        1246                            7.10
01/03/2004                         8039.00        1212                            6.63
01/04/2004                         7828.35        1067                            7.33
01/05/2004                         9036.95        1261                            7.16
01/06/2004                        10089.70        1390                            7.25
01/07/2004                         6798.75        1014                            6.70
01/08/2004                         7989.05        1024                            7.80
01/09/2004                         8416.45        1179                            7.13
01/10/2004                         5809.80         841                            6.90
01/11/2004                         7483.55        1109                            6.74
01/12/2004                         9423.05        1272                            7.40
01/13/2004                         7890.85        1117                            7.06
01/14/2004                         8324.95        1142                            7.28
01/15/2004                         7693.70        1089                            7.06
01/16/2004                         8270.40        1117                            7.40
01/17/2004                         6564.00         928                            7.07
01/18/2004                         9736.95        1328                            7.33
```

4.2.2 Changement de précision décimale

`DEC(VAL, NB_DIGITS_BEFORE, NB_AFTER)` , par exemple `DEC(1, 5, 2)` .

4.3 Comparaison de deux totaux

Pour cela, on va pouvoir faire deux jointures (sur les dates) et un soustraction sur les résultats.

```sql
SELECT t1.date, sales_cume_west, sales_cume_south,
               sales_cume_west - sales_cume_south AS west_vs_south
FROM
    (SELECT date, SUM(dollars) AS total_sales,
       SUM(SUM(dollars)) OVER(ORDER BY date
          ROWS UNBOUNDED PRECEDING) as sales_cume_west
       FROM aroma.market a,
            aroma.store b,
            aroma.sales c,
            aroma.period d
       WHERE a.mktkey = b.mktkey
          AND b.storekey = c.storekey
          AND d.perkey = c.perkey
          AND year = 2006
          AND month = 'MAR'
          AND region = 'West'
       GROUP BY date) AS t1
JOIN
    (SELECT date, SUM(dollars) AS total_sales,
       SUM(SUM(dollars)) OVER(ORDER BY date
          ROWS UNBOUNDED PRECEDING) AS sales_cume_south
       FROM aroma.market a,
            aroma.store b,
            aroma.sales c,
            aroma.period d
       WHERE a.mktkey = b.mktkey
          AND b.storekey = c.storekey
          AND d.perkey = c.perkey
          AND year = 2006
          AND month = 'MAR'
          AND region = 'South'
       GROUP BY date) AS t2
ON t1.date = t2.date
ORDER BY date;
```

```
DATE      SALES_CUME_WEST        SALES_CUME_SOUTH    WEST_VS_SOUTH
---------- ------------------------ -------------------- --------------
03/01/2006     2529.25              2056.75               472.50
03/02/2006     6809.00              4146.75              2662.25
03/03/2006     9068.75              6366.55              2702.20
03/04/2006    12679.35              8831.30              3848.05
03/05/2006    16228.60             11100.55              5128.05
03/06/2006    19653.30             12665.65              6987.65
03/07/2006    23515.55             14882.90              8632.65
03/08/2006    27207.80             17494.15              9713.65
03/09/2006    31725.95             19745.40             11980.55
03/10/2006    34836.20             21323.40             13512.80
03/11/2006    38257.00             23256.15             15000.85
03/12/2006    42498.20             25091.35             17406.85
03/13/2006    46691.95             27362.85             19329.10
```

4.4 La moyenne des sommes par partition

Ici on veut calculer la moyenne des ventes par ville par trois semaines. Pour cela, on fera :

```
SELECT city, week, SUM(dollars) AS sales,
    AVG(SUM(dollars)) OVER(PARTITION by city
        ORDER BY city, week ROWS 2 PRECEDING) AS mov_avg,
    SUM(SUM(dollars)) OVER(PARTITION BY city
        ORDER BY week ROWS unbounded PRECEDING) AS run_sales
FROM aroma.store a,
     aroma.sales b,
     aroma.period c
WHERE a.storekey = b.storekey
    AND c.perkey = b.perkey
    AND qtr = 'Q3_05'
    AND city IN ('San Jose', 'Miami')
GROUP BY city, week;
```

```
CITY         WEEK       SALES      MOV_AVG     RUN_SALES
------------ ---------- ---------- ---------- ----------------
Miami        27        1838.55     1838.55     1838.55
Miami        28        4482.15     3160.35     6320.70
Miami        29        4616.70     3645.80     10937.40
Miami        30        4570.35     4556.40     15507.75
Miami        31        4681.95     4623.00     20189.70
Miami        32        3004.50     4085.60     23194.20
Miami        33        3915.90     3867.45     27110.10
Miami        34        4119.35     3679.91     31229.45
Miami        35        2558.90     3531.38     33788.35
Miami        36        4556.25     3744.83     38344.60
Miami        37        5648.50     4254.55     43993.10
Miami        38        5500.25     5235.00     49493.35
Miami        39        4891.40     5346.71     54384.75
Miami        40        3693.80     4695.15     58078.55
San Jose     27        3177.55     3177.55     3177.55
San Jose     28        5825.80     4501.67     9003.35
San Jose     29        8474.80     5826.05     17478.15
San Jose     30        7976.60     7425.73     25454.75
San Jose     31        7328.65     7926.68     32783.40
San Jose     32        6809.75     7371.66     39593.15
San Jose     33        7116.35     7084.91     46709.50
San Jose     34        6512.35     6812.81     53221.85
San Jose     35        6911.50     6846.73     60133.35
San Jose     36        5996.10     6473.31     66129.45
San Jose     37        10000.60    7636.06     76130.05
San Jose     38        7274.70     7757.13     83404.75
San Jose     39        9021.15     8765.48     92425.90
San Jose     40        5045.20     7113.68     97471.10
```

Noter le `ROWS n PRECEDING` qui indique qu'on veut regarder les `n` lignes précédentes + la ligne courante.

4.5 Classer les données par rang

On va classer les magasins qui vendent les plus.

```
SELECT store_name, district, SUM(dollars) AS total_sales,
    RANK() OVER(ORDER BY SUM(dollars) DESC) AS rank
FROM aroma.market a,
     aroma.store b,
```

```
    aroma.sales c,
    aroma.period d
WHERE a.mktkey = b.mktkey
    AND b.storekey = c.storekey
    AND d.perkey = c.perkey
    AND year = 2005
    AND month = 'MAR'
    AND region = 'West'
GROUP BY store_name, district;
```

```
STORE_NAME                 DISTRICT          TOTAL_SALES                    RANK
-------------------------- ----------------- ------------------------------
Cupertino Coffee Supply    San Francisco                  18670.50               1
Java Judy's                Los Angeles                    18015.50               2
Beaches Brew               Los Angeles                    18011.55               3
San Jose Roasting Company  San Francisco                  17973.90               4
Instant Coffee             San Francisco                  15264.50               5
Roasters, Los Gatos        San Francisco                  12836.50               6
```

Noter, qu'il existe `DENSE_RANK` qui fait la même chose que `RANK` , sauf qui si plusieurs lignes ont le même rang, le rang suivant sera incrémenté par rapport au nombre de rang identiques.

```
DENSE_RANK   RANK
1            1
2            2
3            3
3            3
5            4
6            5
```

4.6 Opérations sur les dates

Avec les fonctions OLAP, on peut facilement faire des calculs arithmétiques sur des dates.

```
SELECT date - 2 MONTHS - 10 DAYS as due_date,
    date AS cur_date,
    date + 2 MONTHS + 10 DAYS AS past_due
FROM aroma.period
WHERE year = 2004
    AND MONTH = 'JAN';
```

```
DUE_DATE  CUR_DATE  PAST_DUE
---------- ---------- ----------
10/22/2003 01/01/2004 03/11/2004
10/23/2003 01/02/2004 03/12/2004
10/24/2003 01/03/2004 03/13/2004
10/25/2003 01/04/2004 03/14/2004
10/26/2003 01/05/2004 03/15/2004
```

Idem pour `YEARS` , `MONTHS` , `DAYS` , `HOURS` , `MINUTES` , et `SECONDS` .

4.7 Having et Where

Having agit sur les données après le groupage et on peut utiliser toute fonction, colonne ou ensemble de données.

Where agit sur les données avant le groupage, mais ne peut pas utiliser les fonctions d'ensembles (SUM, AVG, ...).

4.7.1 Exemple

```
SELECT MIN(prodkey), MAX(classkey)
FROM aroma.product
HAVING MIN(prodkey) = 0;
```

```
1          2
---------- ----------
         0         12
```

4.8 Faire la somme par valeur de champ via un switch-case

```
SELECT prod_name,
    SUM(CASE WHEN store_name = 'Beaches Brew'
        THEN dollars ELSE 0 END) AS Beaches,
    SUM(CASE WHEN store_name = 'Cupertino Coffee Supply'
        THEN dollars ELSE 0 END) as Cupertino
FROM aroma.market a,
     aroma.store b,
     aroma.period c,
     aroma.product d,
     aroma.class e,
     aroma.sales f
WHERE a.mktkey = b.mktkey
    AND b.storekey = f.storekey
    AND c.perkey = f.perkey
    AND d.classkey = e.classkey
    AND d.classkey = f.classkey
    AND d.prodkey = f.prodkey
    AND region LIKE 'West%'
    AND year = 2004
    AND class_type = 'Pkg_coffee'
GROUP BY prod_name
ORDER BY prod_name;
```

```
PROD_NAME                      BEACHES                        CUPERTINO
------------------------------ ---------------------------------
Aroma Roma                                          3483.50                        4491.00
Cafe Au Lait                                        3129.50                        4375.50
Colombiano                                          2298.25                        2653.50
Demitasse Ms                                        4529.25                        3936.50
Expresso XO                                         4132.75                        4689.25
La Antigua                                          4219.75                        2932.00
Lotta Latte                                         3468.00                        5146.00
NA Lite                                             4771.00                        4026.00
Veracruzano                                         4443.00                        3285.00
Xalapa Lapa                                         4304.00                        5784.00
```