M2 ID APPRENTISSAGE
PYTHON ET MODÈLES DÉCISIONNELS COMPRÉHENSIBLES
LANGAGES DE SCRIPT (ANCIEN INTITULÉ)
PROJET
Brice Mayag

# ANALYSE DES CLASSEMENTS DE PRODUITS

# (LE CAS DU MAGAZINE 60 MILLIONS DE CONSOMMATEURS)

Année académique 2018/2019

L'objectif de ce projet est de sensibiliser les étudiants à une démarche d'analyse, d'interprétation et de transparence des modèles d'évaluation utilisés pour l'élaboration des classements de produits, d'organismes, de personnes, etc. Nous nous appuierons sur des classements de produits publiés par certains magazines comme 60 millions de consommateurs. Ce travail sera réalisé à l'aide d'algorithmes implémentés en langage Python.

## 1 Analyse d'un classement de couches-culottes pour enfants

Le magazine *60 millions de consommateurs* dans son *numéro 540 du mois d'Août 2018* a publié, après des essais comparatifs, un classement de plusieurs marques de couches-culottes pour enfants. Les résultats de leur enquête sont donnés par la Figure 1 ci-dessous.

Dans cette section, nous essayerons d'analyser ce classement élaboré à partir de deux critères : *la performance* et *la composition* avec des poids respectifs de 60% et 40 % [1]. Toutes ces informations peuvent être résumées dans le tableau 1 ci-après :

1. Nous ferons l'hypothèse que les évaluations données pour chaque produit, sur ces deux critères, ont été "correctement" obtenues par agrégation des sous-critères associés à chacun de ces critères.


FIGURE 1 – Classement des couches-culottes issu du magazine 60

| | 1-Performance | 2-Composition | Score Global (/20) |
|---|---|---|---|
| a- Joone | +++ | +++ | 17 |
| b- Pamp. Prem | ++ | ++ | 14.5 |
| c- Pamp. Baby | + | +++ | 12.5 |
| d- Naty | + | +++ | 12.5 |
| e- Pamp. Activ. | + | + | 12.5 |
| f- Carref. Baby | ++ | + | 12.5 |
| g- Lupilu | ++ | $-$ | 12 |
| h- Mots d'enfants | + | $-$ | 12 |
| i- Love & Green | ++ | $-$ | 9.5 |
| j- Lotus Baby | ++ | $--$ | 9.5 |
| k- Pommette | ++ | $--$ | 9.5 |
| l- Lillydoo | + | $--$ | 6.5 |
| | $w_p = 60\%$ | $w_c = 40\%$ | |

| | |
|---|---|
| Très bon : | $+++ \in [17, 20]$ |
| Bon : | $++ \in [13, 16.5]$ |
| Acceptable : | $+ \in [10, 12.5]$ |
| Insuffisant : | $- \in [7, 9.5]$ |
| Très insuffisant : | $-- \in [0, 6.5]$ |

TABLE 1 – Tableau de performance pour le classement des produits Couches-culottes (magazine 60 M. num. 540)

### 1.1 Le score global (/20) associé à chaque produit peut-il être expliqué par une somme pondérée ?

A la lecture du classement publié, il semble clair, pour un consommateur, que le score global donné à chaque produit **résulte** d'une moyenne pondérée (somme pondérée) de ses évaluations sur les deux critères *performance* et *composition*. ***L'objectif dans cette partie est de vérifier l'exactitude ou non de cette hypothèse***.

Pour cela, nous allons associer un nombre réel positif à chaque évaluation qualitative attribuée à un produit. Ainsi, pour le produit "Joone" représenté par la lettre $a$ sur le Tableau 1, si son score de $17/20$ est obtenu à partir d'une somme pondérée notée $f$, alors il existe des nombres réels positifs $U_{1a}(+++) \in [17, 20]$ et $U_{2a}(+++) \in [17, 20]$ tels que

$$f(a) = 0.6\, U_{1a}(+++) + 0.4\, U_{2a}(+++) = 17. \tag{1}$$

Plus généralement, pour un produit $x$ dont le score global $f(x)$ est donné par une somme pondérée $f$, si on note $U_{ix}(\alpha_{ix})$ le nombre réel associé à l'évaluation qualitative $\alpha_{ix}$ du produit $x$ sur le critère $i$ (la valeur qualitative $\alpha_{ix}$ étant donnée par le magazine), alors on aura

$$f(x) = 0.6\, U_{1x}(\alpha_{1x}) + 0.4\, U_{2x}(\alpha_{2x}) \tag{2}$$

Pour déterminer si le classement des couches-culottes, fourni par le magazine, est compatible avec un modèle d'évaluation basé sur la somme pondérée, il nous suffit alors de résoudre le programme linéaire $\mathrm{PL}_1$ suivant (un système d'inéquations linéaires est suffisant pour ce test) :

$$\max \quad U_{1a}(+++) \tag{3}$$

Sous les contraintes

$$0.6\ U_{1a}(+++) + 0.4\ U_{2a}(+++) = f^a \tag{4}$$
$$0.6\ U_{1b}(++) + 0.4\ U_{2b}(++) = f^b \tag{5}$$
$$0.6\ U_{1c}(+) + 0.4\ U_{2c}(+++) = f^c \tag{6}$$
$$0.6\ U_{1d}(+) + 0.4\ U_{2d}(+++) = f^d \tag{7}$$
$$0.6\ U_{1e}(+) + 0.4\ U_{2e}(+) = f^e \tag{8}$$
$$0.6\ U_{1f}(++) + 0.4\ U_{2f}(+) = f^f \tag{9}$$
$$0.6\ U_{1g}(++) + 0.4\ U_{2g}(-) = f^g \tag{10}$$
$$0.6\ U_{1h}(+) + 0.4\ U_{2h}(-) = f^h \tag{11}$$
$$0.6\ U_{1i}(++) + 0.4\ U_{2i}(-) = f^i \tag{12}$$
$$0.6\ U_{1j}(++) + 0.4\ U_{2j}(--) = f^j \tag{13}$$
$$0.6\ U_{1k}(++) + 0.4\ U_{2k}(--) = f^k \tag{14}$$
$$0.6\ U_{1l}(+) + 0.4\ U_{2l}(--) = f^l \tag{15}$$

($PL_1$)

$$f^a \geq f^b + 0.1; f^b \geq f^c + 0.1; f^c = f^d \tag{16}$$
$$f^d = f^e; f^e = f^f; f^f \geq f^g + 0.1; \tag{17}$$
$$f^g = f^h; f^h \geq f^i + 0.1; f^i = f^j \tag{18}$$
$$f^j = f^k; f^k \geq f^l + 0.1 \tag{19}$$

$$17 \leq U_{1a}(+++) \leq 20; 17 \leq U_{2a}(+++) \leq 20 \tag{20}$$
$$17 \leq U_{2c}(+++) \leq 20; 17 \leq U_{2d}(+++) \leq 20 \tag{21}$$

$$13 \leq U_{1b}(++) \leq 16.5; 13 \leq U_{2b}(++) \leq 16.5; 13 \leq U_{1f}(++) \leq 16.5 \tag{22}$$
$$13 \leq U_{1g}(++) \leq 16.5; 13 \leq U_{1i}(++) \leq 16.5 \tag{23}$$
$$13 \leq U_{1j}(++) \leq 16.5; 13 \leq U_{1k}(++) \leq 16.5 \tag{24}$$

$$10 \leq U_{1c}(+) \leq 12.5; 10 \leq U_{1d}(+) \leq 12.5; 10 \leq U_{1e}(+) \leq 12.5 \tag{25}$$
$$10 \leq U_{1e}(+) \leq 12.5; 10 \leq U_{1f}(+) \leq 12.5; 10 \leq U_{2h}(+) \leq 12.5 \tag{26}$$
$$10 \leq U_{1l}(+) \leq 12.5 \tag{27}$$

$$7 \leq U_{2g}(-) \leq 9.5; 7 \leq U_{2h}(-) \leq 9.5; 7 \leq U_{2i}(-) \leq 9.5 \tag{28}$$

$$0 \leq U_{2j}(--) \leq 6.5; 0 \leq U_{2k}(--) \leq 6.5; 0 \leq U_{2l}(--) \leq 6.5 \tag{29}$$

où

- La fonction objectif donnée par l'équation (3) consiste, ici, à maximiser le nombre réel $U_{1a}(+++)$ associé à l'évaluation de la couche-culotte "Joone" sur le critère Performance. En réalité, étant donné que nous cherchons simplement à satisfaire les contraintes de ce programme linéaire, le choix de n'importe quelle fonction objectif suffira à résoudre notre problème. En clair le choix de la fonction objectif de $PL_1$ est arbitraire.
- Les contraintes (4) à (15) correspondent à l'écriture sous forme de somme pondérée, comme dans la formule 2

ci-dessus, des 12 produits à évaluer.

- Les contraintes (16) à (19) correspondent à l'ordre établi dans le classement des 12 produits du magazine. A noter que la valeur $0.1$, dans certains de ces contraintes, permet simplement de satisfaire une inégalité stricte, en termes de notes globales, entre certains produits.
- Les contraintes (20) à (29) permettent d'assurer l'appartenance de chaque nombre réel $U_{ix}(\alpha_{ix})$ à un intervalle lié à la note qualitative $\alpha_{ix}$ telle que donnée par le magazine.

1. Écrire une fonction python `CheckAdditiveModel` qui prend, entre autres, en entrée, un fichier Excel (ou son équivalent) contenant des informations nécessaire à l'établissement du classement de produits fourni par le magazine (par exemple la matrice donnée par la Table 1).

   Cette fonction doit être générique,, c'est-à-dire, indépendante d'un classement donné, avec un nombre de critère inférieur ou égal à 10. Vous l'utiliserez également pour tester d'autres classements.

   `CheckAdditiveModel` retournera un message d'erreur si le classement n'est pas explicable par un modèle type somme pondérée (dans ce cas, le programme linéaire ou système linéaire correspondant n'admet pas de solution réalisable). Au cas où cela est possible, le programme retournera alors des valeurs numériques à associer à chaque produit (pour chaque critère d'évaluation), ainsi que la valeur de la somme pondérée qui lui correspond.

   Vous pourrez utiliser les outils de calcul scientifique proposés en Section 4 pour implémenter votre fonction.

2. 2.1 Testez votre fonction `CheckAdditiveModel` avec le classement des couches.

   On rappelle que cela revient, par exemple, à résoudre le programme linéaire $\text{PL}_1$ lorsque la note globale de chaque couche-culotte est exactement égale à celle donnée par le magazine, c'est-à-dire,

   $$f^a = 17,\ f^b = 14.5,\ f^c = 12.5,\ f^d = 12.5,\ f^e = 12.5,\ f^f = 12.5,\ f^g = 12,\ f^h = 12,\ f^i = 9.5,\ f^j = 9.5,\ f^k = 9.5,\ f^l = 6.5.$$

   Quelle(s) conclusion(s) en tirez-vous ?

   2.2 Testez à nouveau votre fonction `CheckAdditiveModel` avec le classement des couches lorsque la note globale de chaque couche-culotte n'est pas fixée (les variables $f^\alpha$ associées aux couches-culottes n'ont pas de valeurs attribués dans le programme linéaire $\text{PL}_1$).

   2.3 Est-ce que les conclusions que vous en tirez peuvent remettre en cause l'information donnée par ce magazine aux consommateurs ?

3. Dans cette question, nous allons supposer que l'évaluation globale n'est pas la même pour les couches-culottes "Naty" et "Pamp. Activ" d'une part, et les couches-culottes "Love & Green" et "Lotus Baby" d'autre part.

   Pour cela, nous allons supprimer du programme $\text{PL}_1$ les deux contraintes suivantes : $f^d = f^e$ de l'équation (17) et $f^i = f^j$ de l'équation (18). Nous allons également supprimer du programme $\text{PL}_1$ la contrainte $17 \leq U_{2c}(\text{+++})$, liée à la couche c-$Pamp.Baby$.

   3.1 Toujours à l'aide de votre script python, résolvez le programme linéaire $\text{PL}_1$ obtenu en prenant successivement comme fonction objectif $\max f^a$, puis $\min f^a$. Il s'agit, dans ce cas, d'analyser le meilleur et le pire score global obtenu par le produit "Joone" sous ces hypothèses.

   Des couches-culottes notées en dessous de $10/20$ par le magazine obtiennent-ils dans ces classements des notes globales supérieures à $10/20$ ? Lesquelles ?

   3.2 Résolvez le programme linéaire $\text{PL}_1$ obtenu en prenant successivement comme fonction objectif $\max f^l$, puis $\min f^l$. Il s'agit d'analyser le meilleur et le pire score global obtenu par le produit "Lillydoo" sous ces hypothèses.

   Des couches-culottes notées en dessous de $10/20$ par le magazine obtiennent-ils dans ces classements des notes globales supérieures à $10/20$ ? Lesquelles ?

   3.3 En comparant ces différents classements avec ceux du magazine, quelle(s) conclusion(s) en tirez-vous ?

4. A présent, nous choisissons de ne plus tenir compte de l'ordre des produits fourni dans le classement publié par le magazine. En d'autres termes, nous supprimons les contraintes (16) à (19). Nous allons également supprimer du programme $\text{PL}_1$ la contrainte $17 \leq U_{2c}(\text{+++})$, liée à la couche $c\text{-}Pamp.Baby$.

   4.1 Déterminez le score global maximal obtenu par chacun des 12 produits en résolvant à chaque fois le programme $\text{PL}_1$ correspondant.

   4.2 En comparant les différents classements obtenus à ceux du magazine, quelle(s) conclusion(s) en tirez-vous ?

Vous pourrez implémenter, pour vos comparaisons de classements, une fonction `CompareRankings` basée par exemple sur le taux de Kendall ou coefficient de Spearman entre deux classements. Les liens suivants pourront vous être utiles :

`http://eric.univ-lyon2.fr/~ricco/cours/cours/Analyse_de_Correlation.pdf`

`https://fr.wikihow.com/calculer-le-coefficient-de-corr%C3%A9lation-de-Spearman`

`https://lemakistatheux.wordpress.com/2013/05/26/le-coefficient-de-correlation-et-le-test-de-kendall/`

`https://lemakistatheux.wordpress.com/2013/05/22/le-coefficient-correlation-et-le-test-de-spearman/`

## 1.2 Application d'une démarche de classification multicritère : la méthode ELECTRE TRI

Dans cette partie, au lieu d'attribuer une note ($/20$) à chaque couche-culotte, nous allons plutôt l'affecter à une classe à l'aide d'une approche ne nécessitant pas une normalisation des critères : La méthode ELECTRE TRI. Cette dernière est une approche d'Aide MultiCritère à la Décision qui vise à résoudre des problèmes d'affectation (classification) d'objets dans des catégories prédéfinies.

Comme indiqué ci-dessus, les poids associés aux deux critères du classement des couches-culottes, *Performance et Composition*, sont donnés par le vecteur $W = (3/5; 2/5)$. Nous retenons ici comme catégories ou classes, les 5 catégories suivantes (compatibles avec celles suggérées par le magazine) : 'Très Bon" (catégorie $C_5$), "Bon" (catégorie $C_4$), "Acceptable" ($C_3$), "Insuffisant" (catégorie $C_2$) et "Inacceptable" ($C_1$). La méthode ELECTRE-TRI consiste alors à affecter chacune des 12 couches-culottes à une de ces 5 catégories.

On va également supposer que chaque catégorie $C_i$ est délimitée par une frontière supérieure notée $b_{i+1}$ et une frontière inférieure $b_i$. Les frontières $b_{i+1}$ et $b_i$ sont appelées "profils" et représentent des couches-culottes de référence qui peuvent être fictives. Il y a une dominance paréto-stricte entre $b_{i+1}$ et $b_i$. Pour ce classement des couches, nous aurons :

- $C_1$ délimitée par $b_2$ et $b_1$ ;
- $C_2$ délimitée par $b_3$ et $b_2$ ;
- $C_3$ délimitée par $b_4$ et $b_3$.
- $C_4$ délimitée par $b_5$ et $b_4$.
- $C_5$ délimitée par $b_6$ et $b_5$.

Ainsi, comme l'affectation se fait dans 5 catégories distinctes, le profil $b_3$ représente la frontière entre les classes "Très Bon" et "Bon", et $b_2$ la frontière entre les classes "Bon" et "Moyen".

Le principe de la méthode ELECTRE TRI consiste, non pas à comparer les couches-culottes entre elles, mais à les comparer aux six couches-culottes de référence $b_6$, $b_5$, $b_4$, $b_3$, $b_2$ et $b_1$ dont les évaluations sur chaque critère sont résumés à la Table 2. **Attention : dans ce tableau, les profils $b_6$ et $b_1$ doivent toujours avoir des valeurs extrêmes, ne pouvant jamais être atteintes, sur chaque critère.**

Ainsi, l'affectation d'une couche-culotte à une catégorie dépendra de sa comparaison aux profils $b_6$, $b_5$, $b_4$, $b_3$, $b_2$ et $b_1$. Plus formellement, l'affectation d'une couches-culottes dans les catégories se base sur le concept de sur-classement. On

| | 1 :Performance | 2 :Composition |
|---|---|---|
| $b_6$ | $++++$ | $++++$ |
| $b_5$ | $+++$ | $+++$ |
| $b_4$ | $++$ | $++$ |
| $b_3$ | $+$ | $+$ |
| $b_2$ | $-$ | $-$ |
| $b_1$ | $---$ | $---$ |

TABLE 2 – Matrice de performance des profils

dira qu'une couche-culotte $H$ surclasse le profil $b_i$ (respectivement le profil $b_i$ surclasse la couche-culotte $H$) et on note $H \mathcal{S} b_i$ (respectivement $b_i \mathcal{S} H$) si $H$ est au moins aussi bon que $b_i$ (respectivement $b_i$ est au moins aussi bon que $H$) sur une majorité de les critères (la majorité étant définie par un seuil de majorité $\lambda$). Les étapes de la méthode ELECTRE TRI se décrivent comme suit :

**Étape 1 : Détermination des indices de concordance partiels**

Pour chaque critère $j$, **l'indice de concordance partiel** entre la couche-culotte $H$ et le profil $b_i$ est donné par :

- Si la fonction $g_j$ est à maximiser :

$$c_j(H, b_i) = \begin{cases} 1 & \text{si } g_j(H) \geq g_j(b_i) \\ 0 & \text{sinon} \end{cases} \qquad c_j(b_i, H) = \begin{cases} 1 & \text{si } g_j(b_i) \geq g_j(H) \\ 0 & \text{sinon} \end{cases}$$

- Si la fonction $g_j$ est à minimiser :

$$c_j(H, b_i) = \begin{cases} 1 & \text{si } g_j(b_i) \geq g_j(H) \\ 0 & \text{sinon} \end{cases} \qquad c_j(bi, H) = \begin{cases} 1 & \text{si } g_j(H) \geq g_j(b_i) \\ 0 & \text{sinon} \end{cases}$$

avec $g_j(H)$ et $g_j(b_i)$ représentant respectivement le score de $H$ et $b_i$ sur le critère $j$. Dans notre exemple, ce score représente l'évaluation qualitative attribuée à $H$ et $b_i$. Par conséquent, la comparaison $g_j(H) \geq g_j(b_i)$ signifie simplement que la valeur qualitative de $H$ est au moins aussi bonne que celle de $b_i$.

**Étape 2 : Détermination des indices de concordance globaux**

**L'indice de concordance global** entre la couche-culotte $H$ et le profil $b_i$ est donné par la formule suivante :

$$C(H, b_i) = \frac{\sum_{j=1}^{n} k_j \; c_j(H, b_i)}{\sum_{j=1}^{n} k_j} \qquad C(b_i, H) = \frac{\sum_{j=1}^{n} k_j \; c_j(b_i, H)}{\sum_{j=1}^{n} k_j}$$

où $k_j$ est le poids du critère $j$ et $n$ le nombre de critères.

**Étape 3 : Détermination de la relation de surclassement $\mathcal{S}$**

**La relation de surclassement** se définie à l'aide de l'indice de coupe $\lambda$, appelé seuil de majorité (en général supérieur à $50\%$), qui représente le paramètre déterminant la situation de préférence entre la couche-culotte $H$ et le profil $b_i$. Ainsi pour la couche-culotte $H$ et un profil $b_i$ :

- $H$ surclasse $b_i$ et on notera $H \mathcal{S} b_i$ si et seulement si $C(H, b_i) \geq \lambda$.
- $b_i$ surclasse $H$ et on notera $b_i \mathcal{S} H$ si et seulement si $C(b_i, H) \geq \lambda$.

**Étape 4 : Procédures d'affectation**

Supposons qu'on dispose de $r$ catégories $C_1, \ldots, C_r$ où chaque catégorie $C_i$ est délimitées par deux profils $b_i$ et $b_{i+1}$.

Deux procédures d'affectation de la couche-culotte $H$ sont possibles :

- ***Procédure pessimiste*** : pour chaque couche-culotte $H$, faire décroître les indices des profils de $r$ jusqu'au premier indice $k$ tel que $H \mathcal{S} b_k$. La couche-culotte $H$ est alors affectée à la catégorie $C_k$.

  Dans notre exemple, cette procédure revient à comparer successivement $H$ à $b_6$, $b_5$, $b_4$, $b_3$, $b_2$ et $b_1$. Si $H$ surclasse $b_i$, i.e., $H \mathcal{S} b_i$, alors $H$ est affecté à la catégorie $C_i$.

- ***Procédure optimiste*** : pour chaque couche-culotte $H$, faire croître les indices des profils de 1 jusqu'au premier indice $k$ tel que $b_k \mathcal{S} H$ et $non(H \mathcal{S} b_k)$. La couche-culotte $H$ est alors affectée à la catégorie $C_{k-1}$.

  Dans notre exemple, cette procédure revient à comparer successivement $b_1$, $b_2$, $b_3$, $b_4$, $b_5$ et $b_6$ à $H$. Si $b_i \mathcal{S} H$ et $non(H \mathcal{S} b_i)$ alors $H$ est affecté à la catégorie $C_{i-1}$.

5. Écrire un programme en langage Python permettant d'implémenter les deux versions de la méthode ELECTRE TRI appliquées à un classement de produits fourni par le magazine. Le programme devra donc comporter, au moins les fonctions `EvalOptimiste` et `EvalPessimiste` (dont les paramètres seront définis par chaque groupe) qui retourneront respectivement l'affectation optimiste et pessimiste à partir de la méthode ELECTRE TRI.

   L'ensemble des produits (alternatives), le nombre de critères, la matrice de performance et les paramètres de la méthode (nombre de catégories, poids des critères, seuil de majorité $\lambda$, profils, ...) seront stockés dans un ou plusieurs fichiers Excel ou équivalent. Ces fichiers seront les input du (ou des) programme(s) à implémenter.

6. En utilisant les profils données ci-dessus (voir Table 2) et votre programme Python, appliquez les deux procédures d'affectation d'ELECTRE TRI lorsque le seuil de majorité est $\lambda = 55\%$ et $\lambda = 75\%$.
7. Comparez vos résultats de classification, donnés par ELECTRE TRI, avec ceux du magazine représentés par le Tableau ci-dessous (voir Table 3), en calculant par exemple le(s) taux d'erreur de la classification. La procédure de comparaison, appelée `CompareClassification`, devra être aussi implémentée en langage Python.

   Dans ce Tableau 3, nous supposons que le score global attribué à chaque couche-culotte correspond à une des 5 catégories $C_1$ (Très insuffisant) à $C_5$ (Très Bon) avec une sémantique correspondante à celle du magazine. Par exemple, le score global de 17/20 attribué à la couche-culotte "Joone" est remplacé par la catégorie $C_5$ ("Très bon").

### 1.3 Application d'un arbre de décision

8. A partir du TP précédent sur les arbres de décision, écrire un script python `ArbrePourClassement60` permettant de classer, en test, les couches-culottes en fonction des cinq classes $C_1$ (Très insuffisant) à $C_5$ (Très Bon). Vous comparerez vos résultats avec ceux du Tableau 3.

## 2 Analyse d'un autre classement au choix

9. L'objectif dans cette partie, est de mener par vos propres soins, en vous inspirant de l'analyse effectuée précédemment, une analyse d'un autre classement publié par *le magazine 60 millions de consommateurs.* Les produits concernés sont les suivants :

| | 1-Performance | 2-Composition | Catégorie donnée par le magazine |
|---|---|---|---|
| a- Joone | +++ | +++ | $C_5$ |
| b- Pamp. Prem | ++ | ++ | $C_4$ |
| c- Pamp. Baby | + | +++ | $C_3$ |
| d- Naty | + | +++ | $C_3$ |
| e- Pamp. Activ. | + | + | $C_3$ |
| f- Carref. Baby | ++ | + | $C_3$ |
| g- Lupilu | ++ | $-$ | $C_3$ |
| h- Mots d'enfants | + | $-$ | $C_3$ |
| i- Love & Green | ++ | $-$ | $C_2$ |
| j- Lotus Baby | ++ | $--$ | $C_2$ |
| k- Pommette | ++ | $--$ | $C_2$ |
| l- Lillydoo | + | $--$ | $C_1$ |
| | $w_p = 60\%$ | $w_c = 40\%$ | |

Un score global appartenant à $[17, 20]$ est remplacé par la catégorie $C_5$ (Très bon)
Un score global appartenant à $[13, 16.5]$ est remplacé par la catégorie $C_4$ (Bon)
Un score global appartenant à $[10, 12.5]$ est remplacé par la catégorie $C_3$ (Acceptable)
Le score global appartenant à $[7, 9.5]$ est remplacé par la catégorie $C_2$ (Insuffisant)
Le score global appartenant à $[0, 6.5]$ est remplacé par la catégorie $C_1$ (Très insuffisant)

TABLE 3 – Tableau de performance des Couches-culottes où le score global est remplacé par une catégorie (classe) (magazine 60 M. num. 540)

- **Commandes en ligne-drive** (les données sont disponibles dans le dossier intitulé *Commandes en ligne-drive Num 535_4*) (9 produits et 4 critères)
- **Logiciels de mots de passe** (les données sont disponibles dans le dossier intitulé *Logiciels Mots de passe Num 540_4*) (10 produits et 4 critères)
- **Robots de cuisines** (les données sont disponibles dans le dossier intitulé *Robots cuisines Num 539_4*) (8 produits et 4 critères)
- **Aérosols et Sprays** (les données sont disponibles dans le dossier intitulé *Sprays aerosols Num 538_4*) (12 produits et 4 critères)
- **Tablettes Seniors** (les données sont disponibles dans le dossier intitulé *Tablettes Seniors Num 536_3*) (7 produits et 3 critères)
- **Cafetières : modèles à dosettes ou capsules** (les données sont disponibles dans le dossier intitulé *Cafetieres dosettes Num 510_3* ) (10 produits et 3 critères)

Chaque groupe devra choisir un thème (jeu de données) parmi les six proposés, différent des autres groupes.

# 3 Travail à faire

- Pour réaliser ce projet, chaque groupe devra répondre aux neufs questions ci-dessus.
- Toute initiative supplémentaire pour la réalisation du projet est la bienvenue et sera appréciée.
- Les résultats des différentes approches devront être rendus sous fichier Excel ou équivalent contenant pour chaque produits, ses évaluations suivant chaque critère, sa note globale (pour les méthodes de somme pondérée) et la catégorie à laquelle elle est affectée (pour la méthode ELECTRE TRI).
- Dans la mesure du possible, présentez vos résultats sur une page web, comme des réponses accessibles à tout consommateur. Ainsi, l'internaute (consommateur), qui n'est pas un spécialiste en informatique pourra alors voir

pour chaque alternative, ses évaluations suivant chaque critère, sa note globale (pour les méthodes de somme pondérée) et la catégorie à laquelle elle est affectée (pour la méthode ELECTRE TRI).

- Les codes sources produits devront être le plus générique possible.
- Chaque groupe est invité à rédiger un rapport écrit (.pdf) décrivant brièvement ses étapes de réalisation du projet. Ce rapport, ainsi que le(s) fichier(s) source .py et/ou .html **savamment commentés**, et tout autre élément pouvant aider à la compréhension du travail réalisé, seront envoyés par mail à `brice.mayag@dauphine.fr`.
- En une page maximum, vous donnerez votre sentiment sur les classements de produits proposés par le magazine *le magazine 60 millions de consommateurs* (cela peut-être aussi un sentiment général). Pour cela, vous pourrez par exemple répondre aux questions suivantes, en justifiant vos réponses :
  - ⋆ Les classements publiés vous paraissent-ils clairs, explicables, ou opaques (pour n'importe quel consommateur) ?
  - ⋆ Ces classements vous paraissent-ils non biaisés ?
  - ⋆ Le magazine devrait-il en dire plus sur la manière de classer les produits ? si oui, comment devra-t-elle être transparente pour convaincre le lecteur ?
  - ⋆ Préférerez-vous une évaluation globale numérique ou qualitative (comme ELECTRE TRI) ?
- La liste de tous les membres du groupe devra être indiquée dans le rapport papier .pdf et le(s) fichier(s) source.
- **Date limite d'envoi du rapport et des fichiers : Dimanche 16 Mars 2019 à 23h59**. Tout retard sera sanctionné. La soutenance des projets (devant toute la promotion), d'une dizaine de minutes par groupe, est prévue le **Lundi 25 Mars 2019**.

# 4 Calcul scientifique en Python

Deux modules de calcul scientifique du langage Python pourront nous être utiles dans la modélisation des préférences du projet : NumPy et SciPy. La distribution Canopy contient les deux modules.

## Module NumPy

NumPy propose des tableaux multidimensionnels pour Python ainsi qu'une large gamme d'opérations efficaces sur ces tableaux : arithmétique, fonctions mathématiques, opérations structurales, etc. Les opérations sont inspirées par des langages comme APL ou Matlab. On y trouve aussi des fonctions permettant de résoudre des problèmes d'algèbre linéaire tels que la résolution de système d'équations. NumPy est donc la bibliothèque de base pour toute application de Python dans le domaine du calcul scientifique. Il s'agit d'un module stable, bien testé et relativement bien documenté : `http://docs.scipy.org/doc/`, `http://docs.scipy.org/doc/numpy/reference/`. Pour importer ce module, on recommande d'utiliser

```
>>> import numpy as np
```

Toutes les fonctions NumPy seront alors préfixées par `np`. Exécutez et analysez le code suivant sur la manipulation simple et efficace des tableaux :

```
>>> x = np.arange(0,2.0,0.1) # De 0 (inclus) à 2 (exclus) par pas de 0.1
>>> x
array([ 0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ,
1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9])
>>> np.size(x) # Sa taille
20
```

```
>>> x[0] # Le premier élément
0.0
>>> x[1] # Le deuxième élément
0.10000000000000001
>>> x[19] # Le dernier élément
1.9000000000000001
>>> x[20] # Pas un élément !
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError : index 20 is out of bounds for axis 0 with size 20
>>> a = np.array ([[1,2,3], [4,5,6], [7,8,9]])
>>> a
array([[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])
>>> b = 2 * a # Multiplication de chaque terme
>>> c = a + b # Sommation terme à terme
>>> np.dot(a, b) # Produit de matrices
array([[ 60, 72, 84],
[132, 162, 192],
[204, 252, 300]])
>>> a * b # Produit terme à terme
array([[ 2, 8, 18],
[ 32, 50, 72],
[ 98, 128, 162]])
>>> a=np.array([1,2])
>>> np.outer(a,a) # Calcule le produit a*transpose(a), soit une matrice.
array([[1, 2],
[2, 4]])
```

On peut facilement effectuer des coupes dans un tableau Numpy. Cette fonctionnalité est particulièrement importante en calcul scientifique pour éviter l'utilisation de boucles.

```
>>> t = np.array([1,2,3,4,5,6])
>>> t[1 :4] # de l'indice 1 à l'indice 4 exclu !!!ATTENTION!!!
array([2, 3, 4])
>>> t[:4] # du debut à l'indice 4 exclu
array([1, 2, 3, 4])
>>> t[4 :] # de l'indice 4 inclus à la fin
array([5, 6])
>>> t[:-1] # excluant le dernier element
array([1, 2, 3, 4, 5])
>>> t[1 :-1] # excluant le premier et le dernier
array([2, 3, 4, 5])
>>> a = np.arange(10)
>>> np.sum(a)
45
>>> np.mean(a)
4.5
>>> M=np.array([[0,1],[2,3]])
>>> M
array([[0, 1],
```

```
[2, 3]])
>>> M.transpose() # Calcule la transposée de la matrice M.
array([[0, 2],
[1, 3]])
```

Il est également possible de résoudre des systèmes linéaires. Par exemple, pour résoudre le système d'équations $\begin{cases} 3x_0 + x_1 = 9 \\ x_0 + 2x_1 = 8 \end{cases}$, on peut exécuter :

```
>>> a = np.array([[3,1], [1,2]])
>>> b = np.array([9,8])
>>> x = np.linalg.solve(a, b)
>>> x
array([ 2., 3.])
```

## Module SciPy

Scipy est un projet visant à unifier et fédérer un ensemble de bibliothèques Python à usage scientifique. Cette distribution de modules est destinée à être utilisée avec le langage interprété Python afin de créer un environnement de travail scientifique très similaire à celui offert par Matlab. Il contient par exemple des modules pour l'optimisation, l'algèbre linéaire, les statistiques ou encore le traitement du signal. Il offre également des possibilités avancés de visualisation grâce au module "Matplotlib". SciPy est un module stable, bien testé et relativement bien documenté. `http://docs.scipy.org/doc/`,`http://docs.scipy.org/doc/scipy/reference/`.

Le module SciPy réalise les différentes opérations sur des tableaux numériques (`ndarray`) de NumPy. On peut donc directement utiliser ces tableaux comme arguments pour les différentes fonctions.

```
>>> import scipy
>>> from scipy import linalg
>>> mat = np.array([[1, 2], [2, 4]])
>>> mat
array([[1, 2],
[2, 4]])
>>> linalg.det(mat)
0.0
```

Exécutez et analysez le code sur la résolution du programme linéaire suivant : $\begin{cases} \min 70\ x_1 + 80\ x_2 + 85\ x_3 \\ s.t. \\ x_1 + x_2 + x_3 = 999 \\ x_1 + 4\ x_2 + 8\ x_3 \leq 4500 \\ 40\ x_1 + 30\ x_2 + 20\ x_3 \leq 36000 \\ 3\ x_1 + 2\ x_2 + 4\ x_3 \leq 2700 \\ x_1 \geq 0 \end{cases}$

```
import numpy as np
from scipy.optimize import linprog
from numpy.linalg import solve

A_eq = np.array([[1,1,1]])
b_eq = np.array([999])
```

```
A_ub = np.array([
[1, 4, 8],
[40,30,20],
[3,2,4]])

b_ub = np.array([4500, 36000,2700])

c = np.array([70, 80, 85])

res = linprog(c, A_eq=A_eq, b_eq=b_eq, A_ub=A_ub, b_ub=b_ub,
bounds=(0, None))

print('Optimal value:', res.fun, '\nX:', res.x)

Optimization terminated successfully.
Current function value: -22.000000
Iterations: 1
('Optimal value:', 73725.0, '\nX:', array([ 636.,  330.,   33.]))
```

Le package `optlang` permet également de résoudre des programmes linéaires via SciPy. Par exemple, en résolvant le programme linéaire :

$$\begin{cases} \max 10\,x_1 + 6\,x_2 + 4\,x_3 \\ s.t. \\ x_1 + x_2 + x_3 \leq 100 \\ 10\,x_1 + 4\,x_2 + 5\,x_3 \leq 600 \\ 40\,x_1 + 30\,x_2 + 6\,x_3 \leq 300 \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_3 \geq 0 \end{cases}$$

```
from optlang import Model, Variable, Constraint, Objective

# All the (symbolic) variables are declared, with a name and
optionally a lower and/or upper bound.

x1 = Variable('x1', lb=0)
x2 = Variable('x2', lb=0)
x3 = Variable('x3', lb=0)

# A constraint is constructed from an expression of variables and a
lower and/or upper bound (lb and ub).

c1 = Constraint(x1 + x2 + x3, ub=100)
c2 = Constraint(10 * x1 + 4 * x2 + 5 * x3, ub=600)
c3 = Constraint(2 * x1 + 2 * x2 + 6 * x3, ub=300)

# An objective can be formulated

obj = Objective(10 * x1 + 6 * x2 + 4 * x3, direction='max')

# Variables, constraints and objective are combined in a Model
object, which can subsequently be optimized.
```

```
model = Model(name='Simple model')
model.objective = obj
model.add([c1, c2, c3])
status = model.optimize()
print("status:", model.status)
print("objective value:", model.objective.value)
print("----------")

for var_name, var in model.variables.items():
print(var_name, "=", var.primal)
```

On obtient comme résultat :

```
('status:', 'optimal')
('objective value:', 733.3333333333333)
----------
('x1', '=', 33.33333333333333)
('x2', '=', 66.66666666666667)
('x3', '=', 0.0)
```