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

LE PALMARÈS DES HÔTELS DE LUXE

Année académique 2017/2018

L'objectif de ce projet est d'élaborer, grâce à des modèles d'évaluation transparents, le palmarès (classement) des hôtels de luxe de six pays d'Europe. Le palmarès obtenu sera comparé au classement des hôtels proposé par le site internet de réservation `www.booking.com`.

Chaque groupe devra proposer un palmarès relatif à un pays différent. Les données sont disponibles sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`, fichier Hotels Luxe.xlsx, où pour chaque hôtel, nous disposons des informations suivantes :

- ⋆ **Nom Hôtel** : le nom de l'hôtel de luxe ;
- ⋆ **Total Négatifs** : le nombre total de mots "négatifs" recensés dans les commentaires laissés par les internautes pour cet hôtel *(critère à minimiser)* ;
- ⋆ **Total Positifs** : le nombre total de mots "positifs" recensés dans les commentaires laissés par les internautes pour cet hôtel *(critère à maximiser)* ;
- ⋆ **Moyenne Reviews** : la moyenne arithmétique des notes (/10) données par les internautes ayant laissé des commentaires pour cet hôtel *(critère à maximiser)*.
- ⋆ **Total Reviews** : nombre total de commentaires laissés pour cet hôtel *(critère à maximiser)* ;
- ⋆ **Note de Booking** : Note globale (/10) attribuée par le site internet `www.booking.com` à cet hôtel.

La construction du palmarès peut être vue comme un problème d'évaluation des alternatives, ou encore de classement des alternatives de la meilleure à la moins bonne, en tenant compte de plusieurs points de vue ou critères très souvent conflictuels entre eux. Le domaine scientifique dédié à la résolution de ce type de problème est appelé Aide MultiCritère à la Décision (AMCD). C'est une branche de la recherche opérationnelle qui a pour but d'aider le décideur à analyser de manière scientifique un problème de décision avec plusieurs critères ou points de vue, et de lui apporter une aide dans sa prise de décision finale. En ce sens là, elle diffère des autres disciplines de la recherche opérationnelle telle que l'optimisation.

Exemple 1. *L'onglet France de ce fichier regroupe 457 hôtels de luxe français. Nous considérerons un ensemble N de 4 critères pour évaluer chaque hôtel de luxe français :*

$$N = \{1 : \textit{Total Négatifs};\ 2 : \textit{Total Positifs};\ 3 : \textit{Moyenne Reviews};\ 4 : \textit{Total Reviews}\}.$$

Par exemple, les hôtels "Hotel Royal Elysées" (deuxième ligne du fichier Excel) et "Le Lavoisier" (neuvième ligne du fichier Excel) seront décrits par un quadruplet (composé de 4 composants issus des 4 critères) :

- *Hotel Royal Elys es* $= (509; 469; 9; 45)$ *;*
- *Le Lavoisier* $= (217; 107; 10; 12)$.

L'ensemble des 457 hôtels constitue en AMCD l'ensemble des alternatives (objets) à évaluer.

L'implémentation des fonctions ci-dessous devra être la plus générique possible.

1 Préparation des données et normalisation des échelles

1. Construire la fonction `criteriaSet(fichier)` qui retournera, à partir d'un fichier Excel ou csv, l'ensemble N des critères à prendre en compte dans notre problème. On supposera donc que le fichier Excel ou csv contient n critères et pour chaque critère l'information indiquant s'il est à minimiser ou à maximiser.

   L'ensemble N retournée pourra par exemple être un dictionnaire (ou une liste ou une matrice, etc).

   Dans notre exemple à 4 critères, on obtiendrai :

   ```
   criteriaSet={"Total Négatifs": "Min"; "Total Positifs": "Max"; "Moyenne
   Reviews": "Max"; "Total Reviews": "Max"}
   ```

2. Construire la fonction `performanceMatrix(fichier)` qui retournera, à partir d'un fichier Excel ou csv, la matrice de performance (encore appelée tableau de performance) associée à notre problème. Cette matrice pourra par exemple être un dictionnaire ou une liste ou une matrice, etc.

3. Pour établir un classement des hôtels, nous allons dans un premier temps transformer les valeurs prises par un hôtel en des valeurs comprises entre 0 et 1. Cette étape est connue sous les termes de **normalisation des échelles**. Il existe de nombreuses méthodes de normalisation.

   On supposera ici que l'ensemble des valeurs sur un critère i est donné sous forme d'un intervalle $[L_i, U_i]$ (L_i étant la borne inférieure et U_i étant la borne supérieure).

   Nous choisirons pour ce projet, les fonctions de normalisation suivantes qui permettent, pour un hôtel $h = (h_1, \ldots, h_n)$ évalué sur n critères, d'obtenir un vecteur normalisé $(u_1(h_1), \ldots, u_n(h_n)) \in [0, 1]$:

$$\begin{cases} u_i(h_i) = \dfrac{h_i - L_i}{U_i - L_i} & \text{si } i \text{ est un critère à maximiser } (critère\ 2,\ 3 \text{ et } 4 \text{ dans notre exemple}) \\ \\ u_i(h_i) = \dfrac{h_i - U_i}{L_i - U_i} & \text{si } i \text{ est un critère à minimiser } (critère\ 1 \text{ dans notre exemple}) \end{cases} \tag{1}$$

$$\begin{cases} u_i(h_i) = \dfrac{h_i}{U_i} & \text{si } i \text{ est un critère à maximiser } (critère\ 2,\ 3 \text{ et } 4 \text{ dans notre exemple}) \\ \\ u_i(h_i) = 1 - \dfrac{h_i}{U_i} & \text{si } i \text{ est un critère à minimiser } (critère\ 1 \text{ dans notre exemple}) \end{cases} \tag{2}$$

   3.1 Construire, pour chacune de ces 4 formules de normalisation, la fonction en langage python permettant d'obtenir la valeur normalisée $u_i(h_i)$ de la composante h_i. Il peut être judicieux de construire au préalable une fonction qui retourne, pour un critère i donné, ses valeurs minimale L_i et maximale U_i.

   Exemple 2.

$$N = \{1 : \textit{Total Négatifs};\ 2 : \textit{Total Positifs};\ 3 : \textit{Moyenne Reviews};\ 4 : \textit{Total Reviews}\}.$$

   Nous aurons :

   - *Pour le critère "Total Négatifs" :* $L_1 = 117$ *et* $U_1 = 26735$.
   - *Pour le critère "Total Positifs" :* $L_2 = 107$ *et* $U_2 = 22105$.
   - *Pour le critère "Moyenne Reviews" :* $L_3 = 5$ *et* $U_3 = 10$.
   - *Pour le critère "Total Reviews" :* $L_4 = 10$ *et* $U_4 = 1459$.

   Les valeurs de l'hôtel Le Lavoisier $= (217; 107; 10; 12)$ *seront normalisées (entre 0 et 1) comme suit :*

   - *Le Lavoisier* $= (0.9962; 0; 1; 0.0014)$ *en utilisant la formule* (1).
   - *Le Lavoisier* $= (0.992; 0.005; 1; 0.008)$ *en utilisant la formule* (2).

   3.2 Construire les fonctions `normalizedperformanceMatrix1` et `normalizedperformanceMatrix2` qui retourneront chacune, en utilisant respectivement les formules (1) et (2), la matrice de performance *normalisée* du problème traité.

2 Palmarès construit avec le modèle de la somme pondérée

Nous allons dans cette partie établir un premier classement de nos hôtels en utilisant une somme pondérée classique comme modèle d'évaluation.

Nous supposons que la note globale $f(u_1(h_1), \ldots, u_n(h_n))$ obtenue par un hôtel $(u_1(h_1), \ldots, u_n(h_n))$, en d'autres termes son évaluation globale à partir de son vecteur normalisé, est donnée par la formule :

$$F(u_1(h_1), \ldots, u_n(h_n)) = \sum_{i=1}^{n} w_i \, u_i(h_i) \tag{3}$$

où

- Le poids associé au critère i est représenté par le réel positif w_i, $i = 1, \ldots, n$;
- $\sum_{i=1}^{n} w_i = 1$.

Exemple 3.

$$N = \{1 : \textit{Total Négatifs};\ 2 : \textit{Total Positifs};\ 3 : \textit{Moyenne Reviews};\ 4 : \textit{Total Reviews}\}.$$

Si on attribue le vecteur poids $(0.2; 0.2; 0.3; 0.3)$ *aux critères 1,2,3 et 4 respectivement, nous aurons pour l'hôtel Le Lavoisier* $= (217; 107; 10; 12)$ *:*

- $F(0.9962; 0; 1; 0.0014) = 0.2 \times 0.9962 + 0.2 \times 0 + 0.3 \times 1 + 0.3 \times 0.0014 = 0.4996$ *en utilisant la formule de normalisation* (1).
- $F(0.992; 0.005; 1; 0.008) = 0.2 \times 0.992 + 0.2 \times 0.005 + 0.3 \times 1 + 0.3 \times 0.008 = 0.5018$ *en utilisant la formule* (2).

4. Construire une fonction `palmaresSommePoidsConnus` qui retourne le classement des hôtels de luxe à partir d'un jeu de poids connu sur les critères et une matrice de performance.

   Concernant les hôtels de luxe, le classement devra être testé avec les trois jeux de poids suivants :

   $$N = \{1 : \text{Total Négatifs};\ 2 : \text{Total Positifs};\ 3 : \text{Moyenne Reviews};\ 4 : \text{Total Reviews}\}.$$

   - $(w_1, w_2, w_3, w_4) = (1/4; 1/4; 1/4; 1/4)$
   - $(w_1, w_2, w_3, w_4) = (0.2; 0.2; 0.3; 0.3)$
   - $(w_1, w_2, w_3, w_4) = (0.3; 0.3; 0.2; 0.2)$

   Pour chacun de ces cas, on établira les classements pour les deux types d'échelles normalisées (voir les équations (1) et (2)). En les comparant, quelles conclusions tireriez-vous ?

5. En AMCD, les poids associés aux critères s'obtiennent en général à partir des préférences exprimées par le décideur. Ces préférences peuvent des comparaisons d'objets (Ex : l'hôtel a est jugé meilleur que l'hôtel b) ou de poids entre critères (Ex : le poids du critère "Total négatifs" est plus important que celui du critère "Total positifs").

   - ▷ Une préférence stricte exprimée entre deux objets a et b pourra être considérée comme un triplet (a, b, P) signifiant $a \ P \ b$ (l'objet a strictement préférée à l'objet b) ;
   - ▷ Une indifférence exprimée entre deux objets a et b pourra être considérée comme un triplet (a, b, I) signifiant $a \ I \ b$ (l'objet a indifférent à l'objet b) ;
   - ▷ Une préférence stricte exprimée entre les poids de deux critères i et j pourra être considérée comme un triplet (w_i, w_j, P) signifiant le poids du critère i est strictement plus important que le poids du critère j) ;
   - ▷ Une indifférence exprimée entre les poids de deux critères i et j pourra être considérée comme un triplet (w_i, w_j, I) signifiant le poids du critère i est égal au poids du critère j).

   Cette modélisation des préférences par un triplet est donnée à titre indicatif. Bien entendu, vous pourrez utiliser une autre structure de données pour prendre en compte les préférences du décideur.

5.1 Construire une fonction `contraintePref` qui retournera la contrainte linéaire associée à une préférence. On tiendra compte des ici deux types d'échelles.

Exemple 4. *Si Hotel Royal Elys es* $= (509; 469; 9; 45)$ *est préféré strictement à Le Lavoisier* $= (217; 107; 10; 12)$*, alors on aura le triplet* $(\textit{Hotel Royal Elys es}, \textit{Le Lavoisier}, P)$ *comme préférence. La fonction* `contraintePref` *retournera pour cette préférence une contrainte en python traduisant l'inéquation suivante (en choisissant l'échelle* (1)*) :*

$$w_1 \times 0.985 + w_2 \times 0.016 + w_3 \times 0.8 + w_4 \times 0.0014 > w_1 \times 0.9962 + w_2 \times 0 + w_3 \times 1 + w_4 \times 0.024$$

5.2 Construire une fonction `poidsPrefInferes` qui retourne un jeu de poids à partir des d'une matrice de performance et des préférences utilisateurs. Le nombre de préférences a demander au décideur sera au moins égal à 5.

5.3 Construire une fonction `palmaresSommePoidsInferes` qui retourne le classement des hôtels de luxe à partir d'un jeu de poids inférés (déterminés à partir des préférences du décideur) et d'une matrice de performance.

Les fonctions à implémenter pour cette question devront certainement résoudre un système d'inéquations linéaires ou un programme linéaire, en utilisant par exemple les modules python CVXOPT, numpy ou Scipy.

6. Comparer les différents classements obtenus, aux questions précédentes, avec celui du site internet `www.booking.com` (voir colonne **Note de Booking**).

3 Palmarès construit avec le modèle de la somme pondérée avec interaction

Le second modèle d'évaluation est la somme pondérée avec interaction. Pour ce modèle, la note globale attribuée à un hôtel de luxe h, décrit par $h = (h_1, h_2, h_3, h_4)$, sera donnée par la formule suivante :

$$\begin{array}{rcl} G(h_1,h_2,h_3,h_4) & = & \Phi_1\, u_1(h_1) + \Phi_2\, u_2(h_2) + \Phi_3\, u_3(h_3) + \Phi_4\, u_4(h_4) \\ & & -\frac{1}{2} I_{12} \left|u_1(h_1) - u_2(h_2)\right| - \frac{1}{2} I_{13} \left|u_1(h_1) - u_3(h_3)\right| - \frac{1}{2} I_{14} \left|u_1(h_1) - u_4(h_4)\right| \\ & & -\frac{1}{2} I_{23} \left|u_2(h_2) - u_3(h_3)\right| - \frac{1}{2} I_{24} \left|u_2(h_2) - u_4(h_4)\right| - \frac{1}{2} I_{34} \left|u_3(h_3) - u_4(h_4)\right| \end{array} \quad (4)$$

où

- $u_i(h_i)$ est la valeur normalisée associée à h_i (cette valeur supposée positive est déterminée le plus souvent par une formule de normalisation comme les formules (1) et (2)), $i = 1, 2, 3, 4$.

- Les paramètres $I(\{i, j\})$ et Φ_i de ce modèle sont déterminées à partir de la donnée des poids (à valeurs dans [0,1]) $w(\{i\}) = w_i$ et $w(\{i, j\}) = w_{ij} = w_{ji}$ associés respectivement aux critères $i = 1, 2, 3, 4$ (singletons $\{i\}$) et aux paires de critères $\{i, j\}$.

  Pour l'évaluation des hôtels, notre modèle de somme pondérée avec interaction sera donc entièrement déterminé par la connaissance des valeurs (poids) $w_1,\ w_2,\ w_3,\ w_4,\ w_{12},\ w_{13},\ w_{14},\ w_{23},\ w_{24},\ w_{34}$ satisfaisant les propriétés suivantes :

$$\left\{ \begin{array}{l} w_{12} \geq w_1 \\ w_{12} \geq w_2 \\ w_{13} \geq w_1 \\ w_{13} \geq w_3 \\ w_{14} \geq w_1 \\ w_{12} \geq w_4 \\ w_{23} \geq w_2 \\ w_{23} \geq w_3 \\ w_{24} \geq w_2 \\ w_{24} \geq w_4 \\ w_{34} \geq w_3 \\ w_{23} \geq w_4 \\ w_{12} + w_{13} + w_{14} + w_{23} + w_{24} + w_{34} - 2(w_1 + w_2 + w_3 + w_4) = 1 \end{array} \right. \quad (5)$$

  En effet, la détermination de ces poids, pour les singletons et les paires, permettront de calculer les paramètres $I(\{i, j\})$ et Φ_i de la formule (4) comme suit :

⋆ $I(\{i,j\}) = I_{ij} = I_{ji} = w_{ij} - w_i - w_j$ est l'indice d'interaction entre les critère i et j (ici les interactions seront supposées positives).

Nous aurons : $I(\{1,2\}) = I_{12} = I_{21} = w_{12} - w_1 - w_2$, $I(\{1,3\}) = I_{13} = I_{31} = w_{13} - w_1 - w_3$, $I(\{1,4\}) = I_{14} = I_{41} = w_{14} - w_1 - w_4$, $I(\{2,3\}) = I_{23} = I_{32} = w_{23} - w_2 - w_3$, $I(\{2,4\}) = I_{24} = I_{42} = w_{24} - w_2 - w_4$, $I(\{3,4\}) = I_{34} = I_{43} = w_{34} - w_3 - w_4$.

⋆ $\Phi_i = w_i + \frac{1}{2}\sum_{j\in N, j\neq i} I_{ij}$ est appelé degré d'importance du critère i, avec $\Phi_1 + \Phi_2 + \Phi_3 + \Phi_4 = 1$.

Nous aurons : $\Phi_1 = w_1 + \frac{1}{2}I_{12} + \frac{1}{2}I_{13} + \frac{1}{2}I_{14}$, $\Phi_2 = w_2 + \frac{1}{2}I_{12} + \frac{1}{2}I_{23} + \frac{1}{2}I_{24}$, $\Phi_3 = w_3 + \frac{1}{2}I_{13} + \frac{1}{2}I_{23} + \frac{1}{2}I_{34}$, $\Phi_4 = w_4 + \frac{1}{2}I_{14} + \frac{1}{2}I_{24} + \frac{1}{2}I_{34}$.

Exemple 5.

$$N = \{1 : \textit{Total Négatifs};\ 2 : \textit{Total Positifs};\ 3 : \textit{Moyenne Reviews};\ 4 : \textit{Total Reviews}\}.$$

On se donne le jeu de poids suivant sur les singletons et les paires : $w_1 = 0.3,\ w_2 = 0,\ w_3 = 0,\ w_4 = 0,\ w_{12} = 0.8,\ w_{13} = 0.3,\ w_{14} = 0.3,\ w_{23} = 0,\ w_{24} = 0,\ w_{34} = 0.2$.

On peut vérifier que ces poids satisfont les conditions (inéquations) données par le système (5)*. Ainsi, nous obtenons :*

$I_{12} = w_{12} - w_1 - w_2 = 0.5$, $I_{13} = w_{13} - w_1 - w_3 = 0$, $I_{14} = w_{14} - w_1 - w_4 = 0$, $I_{23} = w_{23} - w_2 - w_3 = 0$, $I_{24} = w_{24} - w_2 - w_4 = 0$, $I_{34} = w_{34} - w_3 - w_4 = 0.2$.

$\Phi_1 = w_1 + \frac{1}{2}I_{12} + \frac{1}{2}I_{13} + \frac{1}{2}I_{14} = 0.55$, $\Phi_2 = w_2 + \frac{1}{2}I_{12} + \frac{1}{2}I_{23} + \frac{1}{2}I_{24} = 0.25$, $\Phi_3 = w_3 + \frac{1}{2}I_{13} + \frac{1}{2}I_{23} + \frac{1}{2}I_{34} = 0.1$, $\Phi_4 = w_4 + \frac{1}{2}I_{14} + \frac{1}{2}I_{24} + \frac{1}{2}I_{34} = 0.1$.

La valeur de la somme pondérée avec interaction pour l'hôtel Le Lavoisier $= (217; 107; 10; 12)$ *est :*

$$\begin{aligned}
G(0.9962; 0; 1; 0.0014) &= \Phi_1 \times 0.9962 + \Phi_2 \times 0 + \Phi_3 \times 1 + \Phi_4 \times 0.0014 \\
&\quad -\frac{1}{2}I_{12} \times \left|0.9962 - 0\right| - \frac{1}{2}I_{13} \times \left|0.9962 - 1\right| - \frac{1}{2}I_{14} \times \left|0.9962 - 0.0014\right| \\
&\quad -\frac{1}{2}I_{23} \times \left|0 - 1\right| - \frac{1}{2}I_{24} \times \left|0 - 0.0014\right| - \frac{1}{2}I_{34} \times \left|1 - 0.0014\right| \\
&= 0.55 \times 0.9962 + 0.25 \times 0 + 0.1 \times 1 + 0.1 \times 0.0014 \\
&\quad -\frac{1}{2} \times 0.5 \times \left|0.9962 - 0\right| - \frac{1}{2} \times 0 \times \left|0.9962 - 1\right| - \frac{1}{2} \times 0 \times \left|0.9962 - 0.0014\right| \\
&\quad -\frac{1}{2} \times 0 \times \left|0 - 1\right| - \frac{1}{2} \times 0 \times \left|0 - 0.0014\right| - \frac{1}{2} \times 0.2 \times \left|1 - 0.0014\right| \\
&= 0.249
\end{aligned}$$

Plus généralement, évaluer un hôtel $h = (h_1, \ldots, h_n)$ par une somme pondérée avec interaction revient à lui attribuer une note globale $G(h_1, \ldots, h_n)$ définie par la fonction suivante :

$$G(h_1, \ldots, h_n) = \sum_{i=1}^{n} \Phi_i\, u_i(h_i) - \frac{1}{2} \sum_{i,j\in N,\ i\neq j} I_{ij} \left|u_i(h_i) - u_j(h_j)\right| \qquad \forall h = (h_1, \ldots, h_n) \in X \tag{6}$$

où

- $N = \{1, \ldots, n\}$ est un ensemble fini de n critères dont on doit tenir compte.
- Les valeurs $h_1, \ldots, h_n$ représente respectivement les valeurs prises par l'hôtel h sur chaque critère $i = 1, \ldots, n$.
- $u_i(h_i) \in \mathbb{R}^+$ est la valeur normalisée associée à h_i, $i = 1, \ldots, n$.
- $I_{ij} = w_{ij} - w_i - w_j$ est l'indice d'interaction entre les critère i et j, $i, j \in N, i \neq j$ (les interactions seront supposées positives).
- $\Phi_i = w_i + \frac{1}{2}\sum_{j\in N, j\neq i} I_{ij}$ est appelé degré d'importance du critère i, $i = 1, \ldots, n$. $\sum_{i=1}^{n} \Phi_i = 1$.

- La valeur w_i, $i \in N$ représente le poids du critère i, tandis que la valeur w_{ij}, $i, j \in N, i \neq j$, représente le poids des deux critères i et j pris ensemble. Ces valeurs sont calculées de telle sorte que :

$$\begin{cases} w_{ij} \geq w_i \quad \forall i, j \in N \\ \sum_{i \neq j} w_{ij} - (n-2) \sum_{i \in N} w_i = 1 \end{cases} \tag{7}$$

7. Construire une fonction `palmaresSommeInteractionsPoidsConnus` qui retourne le classement des hôtels de luxe à partir d'un jeu de poids connu sur les n critères (poids w_i), les paires de critères (poids w_{ij}) et une matrice de performance.

   Dans l'implémentation de cette fonction, il faudra vous assurer que les conditions du type (7) sont vérifiées, ainsi que l'indice d'interaction est positive pour toute paire de critère, i.e., $I_{ij} \geq 0$, $i, j \in N$.

   Concernant les hôtels de luxe, le classement devra être testé avec les jeux de poids suivants :

$$N = \{1 : \text{Total Négatifs};\ 2 : \text{Total Positifs};\ 3 : \text{Moyenne Reviews};\ 4 : \text{Total Reviews}\}.$$

   - $w_1 = 0.3,\ w_2 = 0,\ w_3 = 0,\ w_4 = 0,\ w_{12} = 0.8,\ w_{13} = 0.3,\ w_{14} = 0.3,\ w_{23} = 0,\ w_{24} = 0,\ w_{34} = 0.2$;
   - $w_1 = 0.3,\ w_2 = 0.1,\ w_3 = 0.1,\ w_4 = 0.2,\ w_{12} = 0.6,\ w_{13} = 0.4,\ w_{14} = 0.5,\ w_{23} = 0.3,\ w_{24} = 0.3,\ w_{34} = 0.3$;

   Pour chacun de ces cas, on établira les classements pour les deux types d'échelles normalisées (voir les équations (1) et (2)). En les comparant, quelles conclusions tireriez-vous ?

8. En vous inspirant de la modélisation des préférences vue à la section précédente, construire une fonction `palmaresSommeInteractionPoidsInferes` qui retourne le classement des hôtels de luxe à partir d'un jeu de poids w_i et w_{ij} inférés (déterminés à partir des préférences du décideur) et d'une matrice de performance.

9. 9.1 Comparer les différents classements obtenus, aux questions précédentes, avec celui du site internet `www.booking.com` (voir colonne **Note de Booking**).

   9.2 Comparer les différents classements obtenus avec les deux méthodes d'évaluation (somme pondérée et somme pondérée avec interaction).

4 Affichage en ligne (via une page web) des résultats

Un des objectifs de ce projet est de publier de manière officielle sur le site les classements obtenus.

10. Pour cela, chaque groupe devra élaborer une page web où sera affiché un classement des hôtels de luxe du pays qu'il a traité, le jeu de poids ayant permis de l'obtenir et le type d'échelle utilisé.
11. Il est souhaitable que cette affichage soit dynamique en ce sens qu'un nouveau classement sera généré dès lors que le jeu de poids change.
12. Le classement des mêmes hôtels de luxe, publié par le site `www.booking.com` (voir colonne **Note de Booking**, devra figuré sur la même page afin que l'internaute puisse les comparer.

5 Travail à faire

- Pour réaliser ce projet, chaque groupe devra répondre aux 12 questions ci-dessus (voir Sections 1 à 4).
- Toute initiative supplémentaire pour la réalisation du projet est la bienvenue et sera appréciée.
- Chaque groupe recevra un fichier de données d'hôtels de luxe d'un pays, et aura pour objectif d'afficher un classement dynamique de ces hôtels.
- 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 .html savamment commentés, et tout autre élément pouvant aider à la compréhension du travail réalisé, seront envoyés par mail à `brice.mayag@dauphine.fr`.
- Le rapport contiendra une conclusion où le groupe donnera son sentiment général sur les principes d'évaluation.
- 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 : Samedi 9 Décembre 2017 à 23h59**. Tout retard sera sanctionné. La soutenance des projets (devant toute la promotion), d'une dizaine de minutes par projet, est prévue le **Lundi 11 Décembre 2017**.

6 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)
```