ENSAE Paris

Compte-rendu du projet Python

Programme de génération de recettes de cuisine

Lucas Degeorge, Baptiste Bupré, Yassine Machta

Pour le 4 mai 2022

Table des matières

Introduction

Au cours de ce projet python, nous avons eu pour ambition de créer un programme qui permet de suggérer à un utilisateur un repas et une recette à partir des ingrédients et des produits qu'il possède dans son réfrigérateur. Nous souhaitons également développer un programme qui permettrait de répondre aux contraintes et aux exigences de différents types de régimes alimentaires. Il peut s'agir par exemple de régimes végétariens, des régimes sans gluten, des régimes de pertes de poids ou de prise de masse.

Nous nous sommes restreint à réaliser un programme qui remplit le cahier des charges suivants :

— Déterminer la recette la plus "profitable" pour l'utilisateur
— Déterminer le magasin le plus "avantageux" pour l'utilisateur
— Fournir une liste de courses à l'utilisateur

Nous entendons par "avantageux" le magasin qui permet à l'utilisateur de minimiser le prix et la distance le séparant de l'individu. Nous entendons par "profitable" la recette qui répond la mieux aux préférences de l'utilisateur. Nous détaillons ces deux notions dans la section *les classes utilisées.*

Nous avons dans un premier temps déterminer les données nutritionnelles sur les différents ingrédients pris en compte dans les recettes. Ensuite, nous avons réalisé un programme de géolocalisation permettant de trouver les magasins et supermarchés les plus proches de l'utilisateur. Enfin, nous avons créer un programme permettant de déterminer la recette la plus profitable. Pour cela nous avons utilisé des classes.

1 Utilisation des scripts et tests unitaires

Le projet rendu contient 5 fichiers *.py*, ainsi qu'un dossier *Pictures* contenant les images de fond de l'interface graphique.

Il convient de renseigner au début du programme le chemin du dossier dans lequel les images se trouvent sur l'ordinateur de l'utilisateur dans la variable *path* présente dans les premières cellules du fichier *NutriBoys.py*

Le fichier *NutriBoys.py* est celui à exécuter pour lancer le programme et l'interface graphique.
Les fichiers *classes_ data.py*, *import_ data.py* et *geolocalization.py* contiennent les codes dont nous précisons le contenu dans les sections suivantes. Le fichier *NutriBoys.py*

Les tests unitaires utilisés au cours du développement des codes sont tous regroupés dans le fichier *unit_ tests.py*. Ils concernent les méthodes développés dans la section *Les classes utilisées*. Ils ont été réalisés en ne considérant d'un ensemble d'ingrédients restreint. Cet ensemble est déclaré dans les premières cellules du fichier.

Les différents modules utilisés pour ce projet sont :
— numpy
— pandas
— ramdom
— os
— math
— googlemaps
— geopy
— time
— tkinter
— ttkwidgets

2 Génération et importation des données

Pour répondre à la contrainte "aucune données réelles", nous avons considérer qu'une liste restreinte d'ingrédients.

Nous avons classés les ingrédients en plusieurs catégories :
— Légumes
— Viandes, oeufs ou poissons
— Féculents
— Fruits
— Matières grasses

Grâce à la base de données Ciqual fournie par le gouvernement, nous avons récupéré des données nutritionnelles sur les différents ingrédients. Ces données concernent :
— Énergie
— Protéine
— Glucides
— Sucres
— Lipides

Nous avons également décidé de générer de manière aléatoire les stocks d'ingrédients dont disposaient les différents magasins (trouvés par la fonction *find_ supermarket*, voir la section suivante. La fonction *shops_ and_ stocks* permet de générer ces stocks et renvoie une liste contenant des objet shop (voir la section *La classe shop* correspondant aux magasins trouvés et dont les stocks ont été générés aléatoirement.

De la même manière, le programme génère un certain nombre de recettes à partir des ingrédients disponibles. La fonction *recipe_ generator* permet de générer ces recettes. Par défaut, 100 recettes sont générées. Elles contiennet toutes un ou deux légumes, une viande ou poisson, un féculent, un fruit et une matière grasse. Les quantités sont également générées de manière aléatoire.

Les différents dictionnaires et DataFrames contenant les données des ingrédients sont présents dans le fichier *import_ data.py*

3 La géolocalisation des magasins à proximité

Cette section correspond au fichier *geolocalization.py*.

Nous avons ici utilisé les modules *googlemaps* et *geopy*.

Dans un premier temps, nous déterminons les coordonnées de l'utilisateur, à partir de l'adresse fournie sur la page *user preferences* (voir la section concernant l'interface graphique).

Nous effectuons ensuite une requête à partir du mot-clé *Supermarket* auprès de l'API *googlemaps* de Google. Nous obtenons ainsi un nombre maximal de 59 magasins dans un rayon donné (la valeur par défaut est de $radius = 4 \cdot 10^3$ m).
Les différents magasins trouvées par l'API *googlemaps* sont stockés dans un DataFrame.

La seconde partie du programme consiste à traiter les données disponibles sur les magasins pour déterminer la distance séparant l'utilisateur du magasin et le temps estimé de trajet nécessaire pour ce rendre à ce magasin. Par défaut le trajet est considéré comme étant effectué en voiture.

Dans le fichier *geolocalization.py*, le code est commenté. Nous avons également rassemblé les différentes parties de ce programme de géolocalisation dans la fonction **find_supermarkets**. Elle prend en argument l'adresse de l'utilisateur. Cette fonction renvoie une liste de triplets contenant trois objets string : le nom du magasin, la distance le séparant de l'utilisaeur et le temps de trajet.
Cette fonction est à la fois présente dans les fichiers *geolocalization.py* et *NutriBoys.py*

4 Le programme de détermination de la recette optimale

4.1 Les classes utilisées

4.1.1 La classe *user*

Elle comprend tous les informations personnelles sur l'utilisateur. Il s'agit de donner de santé, sur le budget, sur le contenu sur du réfrigérateur de l'utilisateur et des coefficients qui permettent de déterminer la recette que fournira le programme.

La méthode allergies permet d'enlever les ingrédients auxquels l'utilisateur est allergique des ingrédients que le programme peut proposer.

La méthode *nearest_ shops* détermine les magasins les plus proches de l'utilisateur. Elle utilise la fonction *find_ supermarkets* décrite dans la section précédente

La méthode *which_ recipe* centralise les différents blocs et permet de déterminer la recette que le programme fournit à l'utilisateur. Elle utilise les méthodes *recipe_ value* et *best_ shop*. Elle revoit un triplet contenant l'objet recipe de la recette sélectionnée par le programme, l'objet shop où l'utilisateur doit aller faire les courses, et un DataFrame contenant les ingrédients à acheter pour pouvoir faire la recette.

4.1.2 La classe *shop*

Elle comprend uniquement le nom du magasin, un DataFrame contenant les ingrédients dans les stocks du magasin (déterminée par la fonction *shops_ and_ stocks*), la distance séparant l'utilisateur du magasin et le temps de trajet estimé. Dans le DataFrame stocks, des informations concernant la quantité, le prix et la date de péremption sont spécifiées. Toutefois, nous n'avons finalement pas pris la date de péremption en compte dans notre programme.

4.1.3 La classe *recipe*

Elle contient un DataFrame contenant les ingrédients nécessaires pour faire la recette, et deux variables prep_time et guests que nous n'avons finalement pas pris en compte dans le fonctionnement du programme.

La méthode food_needed détermine les ingrédients que l'utilisateur devra acheter pour pouvoir faire la recette. Elle renvoie un DataFrame contenant les ingrédients et les quantités nécessaires.

La méthode *best_ shop* détermine à partir d'une liste de magasins le magasin dans lequel l'utilisateur aura le plus intérêt à aller faire ses courses. Pour déterminer le magasin le plus "intéressant". La fonction va calculer le prix totale des courses à effectuer (grâce à la méthode *price* de la classe *errand*) puis aussi à chaque magasin un score prenant en compte le prix et la distance séparant le magasin et l'adresse de l'individu. Le score est déterminé grâce aux préférences de l'individu présentes dans la variable *coefs* de la classe *user*

La méthode *food_ value* détermine les valeurs de l'énergie, des protéines, etc présents dans l'ensemble des ingrédients nécessaires à la recette.

La méthode *recipe_ value* associe à la recette une valeur prenant en compte les indicateurs suivants :

- les valeurs de l'énergie, des protéines, etc présents dans l'ensemble des ingrédients, déterminées grâce à la méthode *food_ value*
- le prix dans le magasin sélectionné par *best_ shop* de l'ensemble des ingrédients à acheter
- la distance séparant le magasin sélectionné par *best_ shop*

De même, le score est déterminé grâce aux préférences de l'individu présentes dans la variable *coefs* de la classe *user*

4.1.4 La classe *errand*

Elle contient un DataFrame contenant la liste des ingrédients à acheter ainsi que les quantités nécessaires.

Une seule méthode est présente, *price*. Elle détermine le prix de l'ensemble des ingrédients nécessaires si les courses sont faites dans le magasin passé en argument.

4.1.5 Exceptions et autres fonctions

Nous avons également eu besoin de déclarer et traiter différentes erreurs au cours de notre projet. Les trois exceptions déclarées sont *NoRecipeFound, NoWhereToBuy* et *ProductNotAvailable*.

Enfin, la fonction *find_ coefs* permet de déterminer le vecteur de préférences de l'utilisateur. Par soucis de simplicité, nous avons pour l'instant pris uniquement en compte le fait que l'individu souhaite prendre de la masse (et ainsi consommer davantage de protéines) ou non. Le cas échéant, le coefficient de préférences des protéines vaut 0,5 et les autres sont repartis uniformément de façon à ce que leur somme vaille 0,5. Si jamais l'utilisateur ne coche pas la case *MassBuilding* et ne vont donc ainsi pas effectuer une prise de masse, tous les coefficients de préférences sont repartis uniformément de façon à ce que leur somme vaille 1.

5 Interface Graphique et calculateur de macro

Nous avons enfin réalisé une interface graphique à l'aide du module *Tkinter* de python.

La réalisation de cette interface graphique s'étend des lignes 440 aux lignes 1200 du fichier *NutriBoys.py*. Elle comprend trois fenêtres s'appelant les unes aux autres. La première fenêtre est un menu permettant d'accéder aux trois fonctionnalités de l'application.

Le premier bouton *user preferences* permet à l'utilisateur de renseigner ses informations personnelles. L'information la plus importante est l'adresse de l'utilisateur. Les allergies permettent de retirer des ingrédients de la liste des ingrédients à considérer. La grande majorité des autres information demandées n'est pas utilisé mais il était initialement prévu de les prendre en compte dans le calcul du vecteur des préférences de l'utilisateur (notamment dans le corps de la fonction *find_ coefs*).

Il est nécessaire de cliquer sur le bouton *summit* avant de retourner sur la page de menu. Une erreur est déclarée si ce n'est pas le cas, indiquant qu'aucune adresse n'a été renseignée.

Une fois les informations personnelles renseignées, l'utilisateur peut demander à l'application de lui suggérer un menu à l'aide du bouton *meal request*. Il devra alors remplir les ingrédients présents dans son réfrigérateur. Une fois cela effectué, la fenêtre se ferme le programme prend une quinzaine de secondes à déterminer la recette optimale. Une nouvelle fenêtre s'ouvre. Elle fournit à l'utilisateur les ingrédients nécessaires pour faire la recette optimale, le magasin ainsi que la liste des courses à faire pour compléter son réfrigérateur.

Enfin, après son repas terminé, l'utilisateur peut déterminer les apports nutritionnels que lui a apporté la recette fournie par le programme dans la fenêtre *macrocalculator*.