session2

October 21, 2024

0.1 Consigne

Le travail attendu pour ces 3 semaines est à faire dans les fichiers `session1.py`, `session2.py`, `session3.py`.

Les fichiers `session{1|2|3}.py` commenceront par un commentaire avec les noms de leur auteur :

```
# prenom1 nom1
# prenom2 nom2
```

Une attention particulière sera portée à la qualité de la documentation de votre code.

Cette semaine, il s'agit de travaillr dans session2.py

[1]:
```
import session2 as project
import pandas as pd
import numpy as np
```

1 Session 2 - Modélisation d'une microferme en maraîchage diversifié

Dans les microfermes en maraîchage diversifié, les produits récoltés sont souvent vendus en direct aux particuliers sous forme de paniers hebdomadaires. Il faut pour satisfaire la clientèle avoir suffisament de sortes de légumes et suffisament de sortes de légume pour chaque catégorie. Dans la suite, on considère 8 catégories de légume. Le maraîcher choisit donc parmi les **cycles de culture** ceux qui sont adaptés pour respecter les contraintes de quantité et de diversité. La viabilité d'une microferme respectant ces contraintes sera évaluées sur la base du revenu et de la charge de travail qu'elle génère. Le but du projet est de simuler des microfermes de manière aléatoire et d'évaluer leur viabvilité.

1.1 1 - Simulation d'une microferme

On dispose de deux bases de données issues de la thèse de Kevin Morel qui rassemble les plus de 1000 cycles de culture et qui donne les critères de quantité et de diversité.

La base de données **"cropping_cycles.csv"** contient plus de 1000 cycles de culture avec pour chaque cycle de culture: - un identifiant entier unique ('ID'), par exemple 233 - le nom en français du légume concerné ('Crop_french'), par exemple 'Tomate cerise' - la catégorie du légume concerné ('Crop category'), par exemple 'Tomato' - le nombre de lots minimum par mois de récolte ('Shmin') et maximum total ('Shmax') qu'on peut cultiver pour ce cycle - la durée de la récolte en mois

('Harvest_last') - pour chaque mois de l'année, un booléen (0 ou 1) qui indique si le légume cultivé dans ce cycle peut être vendu ('Sale_Jan', 'Sale_Feb', 'Sale_Mar',...)

La base de données **"criteria.csv"** donne pour chaque categorie de légume: - le nom de la categorie ('Crop category') - pour chaque mois le nombre minimum de légumes différent de cette catégorie qui doivent être proposés ('Minimal number of crops_1', 'Minimal number of crops_2', ..., 'Minimal number of crops_12') - pour chaque mois la quantité minimum en nombre de lots de légume proposé dans cette catégorie ('Minimal quantity of shares_1', 'Minimal quantity of shares_2',..., 'Minimal quantity of shares_12')

Dans la base de données, plusieurs scénarios sont disponibles, ici on se restreindra à une stratégie de vente sur 12 mois ('Marketing'='12M') et à des climats modérés ('Climat'='Mild')

1.0 Implémenter les fonctions qui chargent les bases de données *cropping_cycles* et *criteria* sous forme de dataframe avec pandas.

[2]:
```
cropping_cycles = project.get_cropping_cycles_database("data/session2/
  ↪cropping_cycles.csv")
criteria = project.get_criteria("data/session2/criteria.csv")
```

1.1 Implémenter la fonction `choose_cycle` qui, pour une catégorie et un mois de vente donnés, choisit un cycle au hasard, le légume étant choisi uniformément parmi ceux possibles. On renverra False si le légume n'est pas requis dans le panier ce mois selon la base *criteria*, sinon on renverra l'identifiant du cycle.

[3]:
```
category = "Potato"
month = 3 #April since month start at 0

cycle_id = project.choose_cycle(category, month, cropping_cycles, criteria)
print(cropping_cycles[cropping_cycles["ID"]==cycle_id])
print(cropping_cycles[cropping_cycles["ID"]==cycle_id]["Sale_Apr"])
```

```
                   Crop_french   ID  Seq Prod_loc Climate Marketing  \
538  Pomme de terre conservation  516  NaN    Field    Mild       12M

     Settingup  Harvest_start  Harvest_end Winter_storage  …  Growing_Apr  \
538          4              9           10            yes  …            1

     Growing_May Growing_Jun  Growing_Jul  Growing_Aug  Growing_Sep  \
538            1           1            1            1            1

     Growing_Oct  Growing_Nov  Growing_Dec  Sales duration
538            1            0            0               8

[1 rows x 42 columns]
538    1
Name: Sale_Apr, dtype: int64
```

1.2 On maintient un dictionaire ou une liste d'objet qui contient les identifiants de cycle et leur nombre de lots associé. Implémenter la fonction suivante qui met à jour

le nombre de lot pour un cycle donné. On attribuera le nombre minimum de lots la première fois que le cycle est tiré, et on incrémente de 1 les fois suivantes où on tire ce cycle, en limitant à Shmax.

[4]:
```
cycles={}
cycle_id = 516
cycle = cropping_cycles[cropping_cycles["ID"]==cycle_id]
cycles = project.update_shares(cycle, cycles, cropping_cycles)
print(cycles)
```

```
{516: 2}
```

1.3 Implémenter la fonction qui répartit les lots sur les mois où la vente est possible pour ce mois. Elle renvoie une liste avec pour chaque mois le nombres de lots à vendre.

[5]:
```
monthly_shares ={}

cycle_id = int(cycle["ID"].to_numpy()[0])
monthly_shares[cycle_id] = project.spread_shares(cycle, cycles, cropping_cycles)
print(monthly_shares)
```

```
{516: [np.float32(0.25), np.float32(0.25), np.float32(0.25), np.float32(0.25),
np.float32(0.0), np.float32(0.0), np.float32(0.0), np.float32(0.0),
np.float32(0.25), np.float32(0.25), np.float32(0.25), np.float32(0.25)]}
```

1.5 Implémenter les fonctions qui renvoient True ou False selon que les critères de quantité et de diversité sont vérifés.

[6]:
```
criteria_potato = criteria[criteria["Crop category"]==category]

quant = project.check_quant(monthly_shares, criteria_potato, month) #on vérifie␣
 ↪le critère de quantité pour un seul mois
div = project.check_div(monthly_shares, cropping_cycles, criteria) #on vérifie␣
 ↪le critère de diversité pour toute l'année
print("Quantity crit.: %s, Diversity crit.: %s"%(quant, div))
```

```
Quantity crit.: False, Diversity crit.: False
```

1.6 Implémenter une fonction qui pour un catégorie de légume donnée renvoie un panier qui respecte les contraintes de diversité et de quantité (un dictionaire avec comme index l'id du cycle et en valeur le nombre de lot par an).

[7]:
```
Box_cat,_ = project.get_box_cat(category, criteria, cropping_cycles)
print(Box_cat)
```

```
{514: 10, 515: 10, 517: 10, 516: 5, 520: 1, 522: 1, 523: 2, 521: 2, 519: 6, 518:
3}
```

1.7 Implémenter une fonction qui renvoie N paniers (on pourra sauvegarder le résultat en json ou csv.)

[8]:
```
N = 10 #100000
categories = np.unique(cropping_cycles["Crop category"])
```

```
boxes = project.get_N_boxes(N, criteria, cropping_cycles, categories)
print(boxes[0])
```

```
{'Carrot': {336: 4, 335: 5, 334: 8, 339: 3, 333: 1, 338: 3, 337: 6, 329: 2, 328:
2, 326: 2, 330: 9, 332: 2}, 'Condiment crop': {401: 4, 301: 7, 399: 4, 293: 3,
403: 2, 296: 1, 493: 1, 492: 2, 491: 8, 295: 3, 496: 8, 292: 3, 488: 3, 298: 7,
487: 2, 299: 5, 297: 3, 400: 4, 402: 2}, 'Cooked green': {308: 5, 422: 3, 412:
2, 373: 4, 497: 4, 365: 3, 362: 2, 358: 2, 304: 2, 420: 1, 408: 3, 498: 4, 319:
2, 321: 4, 499: 3, 364: 1, 502: 4}, 'Fruit crop': {388: 9, 389: 3, 390: 6, 387:
8, 425: 4, 508: 2, 509: 1, 426: 6, 506: 2, 423: 3, 427: 7, 302: 10, 453: 2, 455:
3, 512: 9, 303: 8, 513: 6}, 'Potato': {515: 8, 516: 9, 514: 8, 517: 12, 523: 3,
521: 2, 518: 4, 519: 6}, 'Raw green': {451: 4, 557: 1, 468: 4, 556: 2, 558: 1,
560: 2, 445: 1, 473: 4, 446: 1, 458: 4, 462: 7, 563: 1, 564: 1, 447: 1, 472: 3,
546: 2, 547: 1, 347: 4, 548: 2, 463: 4, 348: 2, 466: 3, 439: 2, 467: 4, 440: 2,
450: 2}, 'Root crop': {494: 6, 543: 3, 486: 3, 344: 3, 318: 6, 544: 5, 541: 1,
479: 2, 495: 2, 314: 2, 480: 2, 528: 2, 527: 2, 478: 2, 317: 3, 312: 3, 539: 1,
485: 1}, 'Tomato': {574: 7, 576: 8, 573: 7, 570: 3, 572: 3}}
```

1.2 2 Estimation de la viabilité d'une microferme

On suppose que le rendement Y_c pour le système de production s, pour la ferme f et le légume c (en lots par m^2 s'écrit):

$$log(Y_c) = a_s + a_f + a_c + r$$

où les e_s, e_f et r suivent des lois normales $\mathcal{N}(m, \sigma)$ et décrivent les effets du système de production, de la ferme considérée et des effets aléatoires respectivement. Les paramètres de ces lois sont donnés dans le tableau suivante:

Paramètre	m	σ
a_s	0.74	0.12
a_f	0.	0.42
r	0.	0.14

de même la charge de travail en heures par m^2 W_c pour le système de production s, pour la ferme f et le légume c s'écrit:

$$log(W_c) = b_s + b_f + b_c + s$$

Paramètre	m	σ
b_s	2.72	0.19
b_f	0.	0.36
s	0.	0.21

On dispose d'une base de données **'Crop_properties.csv'** qui donne pour chaque légume: * son prix au kilo * la quantité par lot en kg * les paramètres a_c et b_c qui décrivent l'effet du légume sur le rendement et la charge de travail.

2.1 Calculer pour chaque microferme simulée son profit en euros par m^2 ainsi que la charge de travail associée en heures/m^2

[9]:
```
crop_properties = pd.read_csv("data/session2/crop_properties.csv")
workload, CA = project.compute_CA_workload(boxes[0], crop_properties)
print(workload, CA)
```

```
2264.3199301238615 1061.9994
```

2.2 Calculer la surface puis le chiffre d'affaire correspondant à 1800 heures de travail.

[10]:
```
ws = []
CAs = []
for box in boxes:
   workload, CA = project.compute_CA_workload(box, crop_properties)
   ws.append(workload)
   CAs.append(CA)

workload_total=1800
CA_total = project.compute_CA(np.array(ws), np.array(CAs), workload_total)
print(CA_total)
```

```
[34100.00430267 54632.70487362 27373.62404455 38495.59306203
 55208.78227224 48013.18595187 41815.67647702 49444.49527443
 52109.73420627 40606.36776608]
```

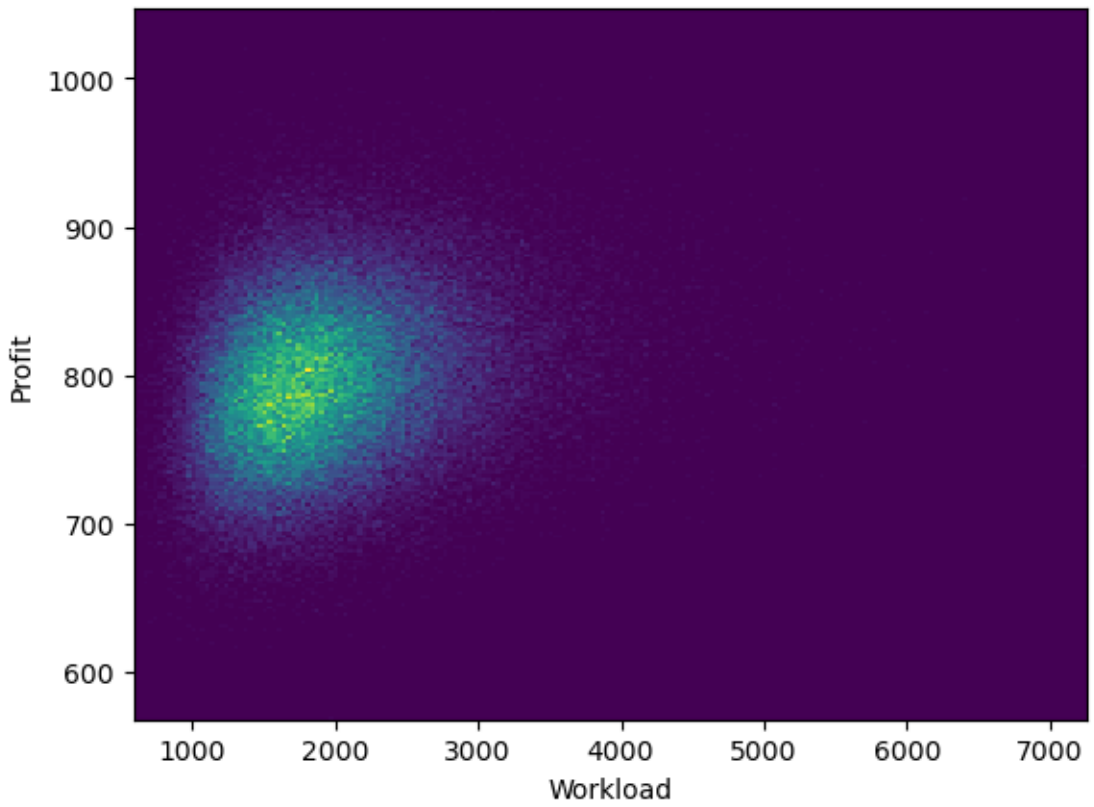
2.3 Calculer la probabilité qu'une microferme fasse un chiffre d'affaire de plus de 40000 euros par an avec une charge de travail inférieure à 1800 heures par an.

[11]:
```
workload_max=1800
CA_min=40000
proba_viable = project.compute_probability_viable(boxes, CA_min, workload_max,␣
 ↪crop_properties)
print(proba_viable)
```

```
0.8
```

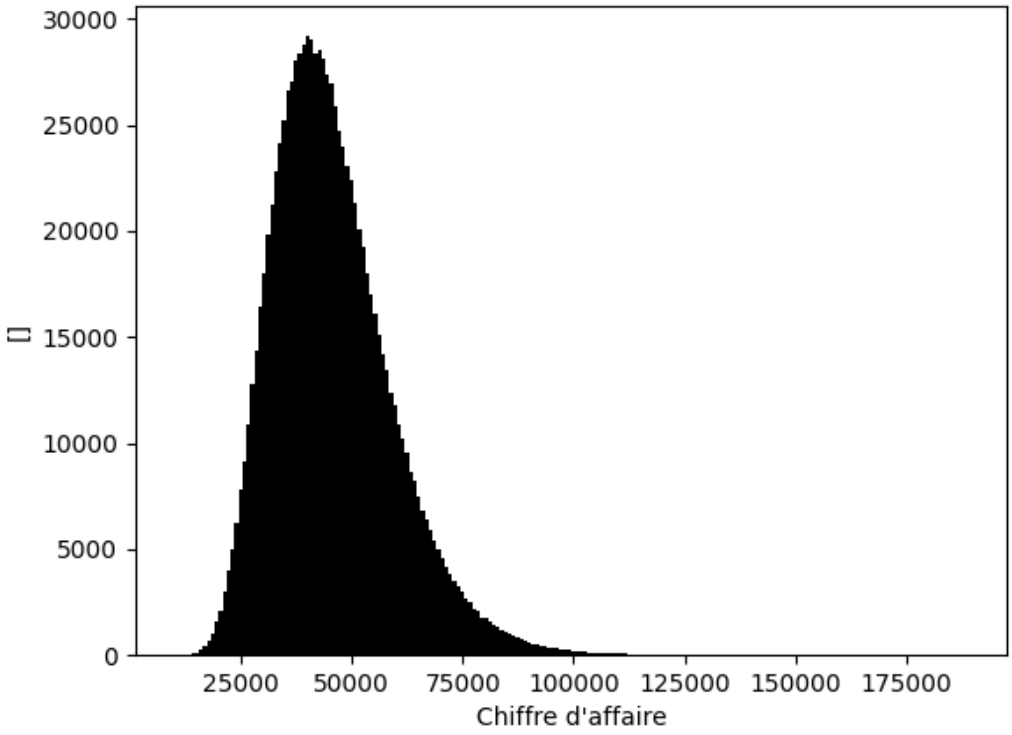
2.4 Faire la figure représentant la distribution jointe de chiffre d'affaire et charge de travail par m^2

[12]:
```
project.figure_distribution(ws,CAs)
```



Faire la figure qui montre la distribution du chiffre d'affaire annuel.

[13]:
```
project.figure_CAtot(CA_total)
```


30000
25000
20000
15000
10000
5000
0
[]
25000
50000
75000
100000
125000
150000
175000
Chiffre d'affaire

[ ]:

[ ]: