OPENCLASSROOMS

Développeur d'application Python

PROJET 7 – Résoudre des problèmes en utilisant des algorithmes en Python

par **Nicolas MARIE**

OBJECTIFS DE L'APPLICATION

- Implémenter des algorithmes permettant d'optimiser des stratégies d'investissement

- **Contraintes** :
  - Chaque action ne peut être achetée qu'une fois
  - On ne peut pas acheter une fraction d'action
  - On ne peut dépenser que 500€ par client

DÉFINITION – PROBLÈME DU SAC À DOS

- En algorithmique, le **problème du sac à dos**, (de l'anglais *Knapsack Problem*) est un problème d'optimisation combinatoire.

- Il consiste à trouver la combinaison d'éléments la plus précieuse à inclure dans un sac à dos, étant donné un ensemble d'éléments décrits par leurs poids et valeurs.

RÉSOLUTION – ALGORITHMES POSSIBLES

- Pour résoudre le problème du sac à dos, il existe plusieurs algorithmes et approches :
  - L'algorithme « Glouton (Greedy) » ; simple, rapide mais non optimal
  - La méthode brute-force ; optimale mais couteuse en temps
  - La méthode de programmation dynamique ; rapide et optimale

- **Problèmes du « sac à dos »**

Pseudo code : **On teste toutes les combinaisons**.

```
1   Algorithme arbre(actions, profit, coût, chemin)
2   Variable
3       action <- [ ] : tableau
4   DEBUT
5       SI actions == [ ]
6           retourner chemin, profit
7       FIN SI
8
9       action <- actions[0]
10      coût_action <- action[1]
11      profit_action <- action[3]
12
13      meilleur_chemin_d, meilleur_profit_d <- arbre(action[1:], profit, coût, chemin)
14
15      SI coût + coût_action <= COÛT_MAXIMUM
16          nouveau_chemin <- chemin + [action]
17          meilleur_chemin_g, meilleur_profit_g <- arbre(action[1:], profit + profit_action, coût + coût_action, nouveau_chemin)
18      SINON
19          meilleur_chemin_g, meilleur_profit_g <- [ ], 0.0
20      FIN SI
21
22      SI meilleur_profit_g > meilleur_profit_d
23          retourner meilleur_chemin_g, meilleur_profit_g
24      SINON
25          retourner meilleur_chemin_d, meilleur_profit_d
26      FIN SI
27  FIN
28
```

Algorithme récursif

Branche droite

Branche gauche

ALGORITHME FORCE-BRUTE APPLIQUÉ

ALGORITHME – PROGRAMMATION DYNAMIQUE

- **Problèmes du « sac à dos »**

  - Principe appliqué:

Étant donné une action i et un coût max c, la solution optimale est :

Si coût de l'action i > coût max c :

- La solution optimale du problème à $i-1$ actions avec le $coût\ \max c$

OU la meilleure solution entre :

- La solution optimale du problème à $i-1$ actions avec le même $coût\ \max c$

- La solution optimale du problème à $i-1$ actions avec $coût\ \max c - coût\ de\ l'action\ i$, à laquelle on ajoute le $bénéfice\ de\ l'action\ i$

- **Problèmes du « sac à dos »**

❶ Il s'agit donc de construire un tableau **pour stocker les bénéfices maximaux** réalisables pour différents coûts maximaux (allant de 0 à 500€ dans notre cas).

❷ Il suffira ensuite de **remonter le tableau** de la dernière à la première ligne pour **reconstruire le chemin des actions à choisir**.

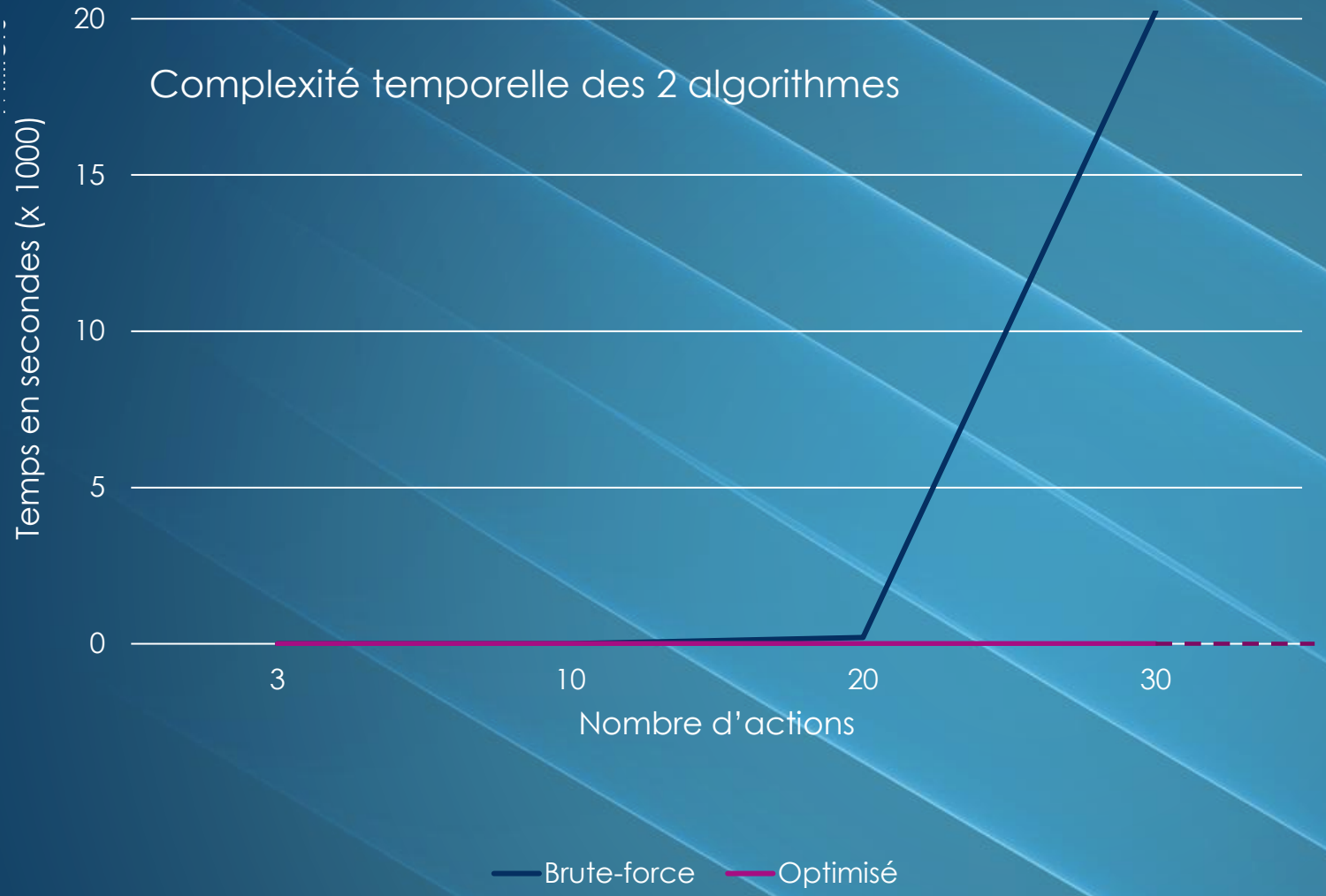
ALGORITHME – PROGRAMMATION DYNAMIQUE

- **Problèmes du « sac à dos »**

Pseudo code :

```
Algorithme bénéfice_maximum(bénéfices_actions, coûts_actions, coût_max)
Variable
n <- taille(actions)
tableau_coûts <- []
DEBUT
    POUR i allant de 0 à coût_max + 1:
        POUR j allant de 0 à n + 1:
            tableau_coût[i][j] <- 0
        FIN POUR
    FIN POUR

    POUR i allant de 1 à n:
        POUR c allant de 0 à coût_max:

            Si coûts_actions[i - 1] > c:
                tableau_coûts[i][c] <- tableau_coûts[i-1][c]
            SINON
                tableau_coûts[i][c] <- meilleur solution entre tableau_coûts[i-1][c] et tableau_coûts[i-1][c - coûts_actions[i - 1]] + bénéfice_actions[i - 1]
            FIN Si

        FIN POUR
    FIN POUR

retourner tableau_coûts[n][coût_max]
FIN
```

ALGORITHME – PROG. DYNAMIQUE APPLIQUÉ



COMPLEXITÉ - GRAPHIQUE

- **Algorithme brute-force**

  - La croissance n'est pas linéaire, mais exponentielle, ce qui rend l'algorithme inexploitable au-delà d'un petit nombre d'actions.
  - Avec 20 actions, le calcul nécessite environ 14 secondes, tandis que le calcul pour 30 actions atteint ~5h37, rendant cette approche impraticable à grande échelle.

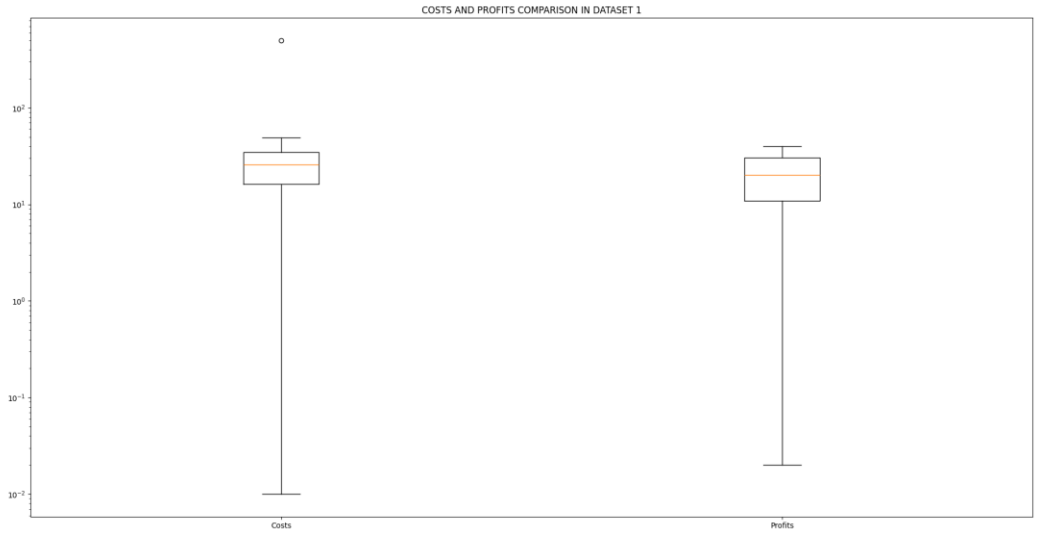
- **Algorithme optimisé**

  - La croissance est linéaire, quasi constante, proche de 0,1s

- Les deux algorithmes :

Algorithmes	Force-brute	Programmation dynamique
Complexité spatiale	$O(n)$ linéaire	$O(n * C)$ linéaire
Complexité temporelle	$O(2^n)$ exponentielle	
Avantages	• Optimal • Peu cher en mémoire (linéaire)	• Optimal • Rapide
Limites	Long à calculer	Gourmand en mémoire (on stocke toutes les solutions optimales intermédiaires)

COMPLEXITÉS

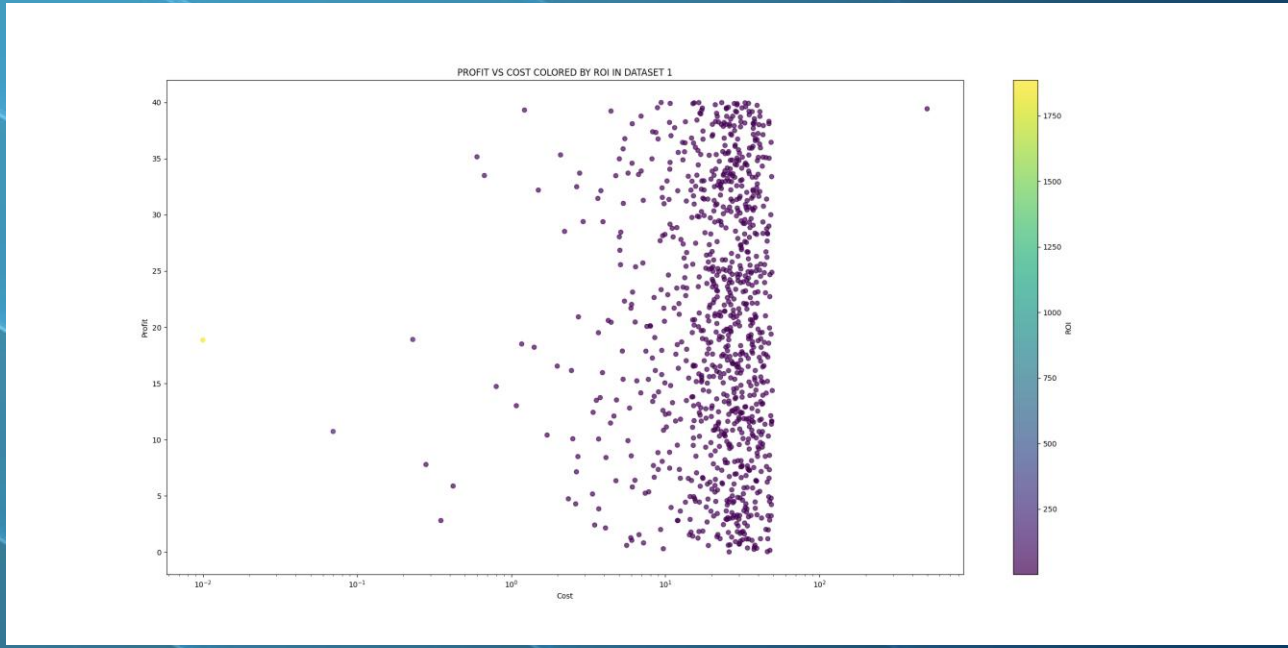


- ANALYSE BOX PLOT:

  - La boîte des coûts est au dessus de la boîte des profits : globalement, retour sur investissement faible.

  - Une valeur aberrante (*Outlier*) côté coût.

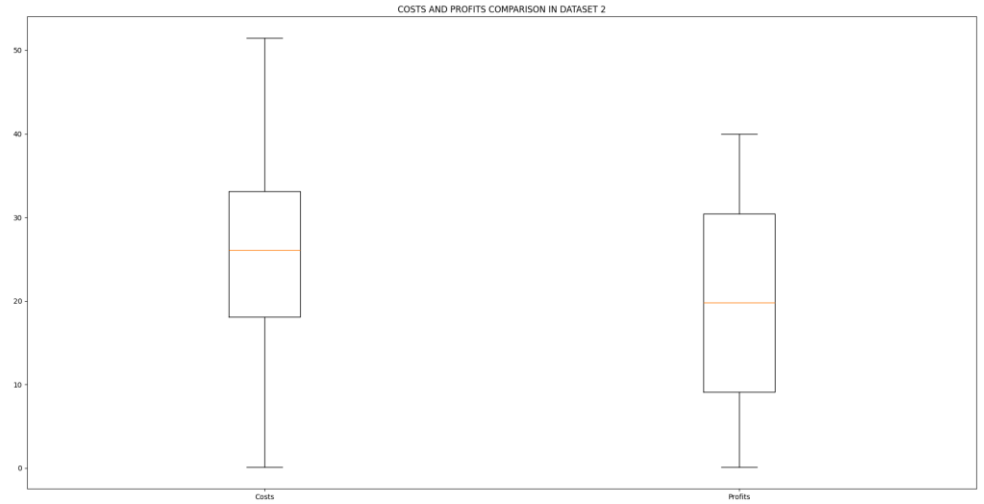
  - Les boîtes sont serrées : la plupart des valeurs (75%) sont peu dispersées. Les 25% inférieur sont dispersées.



- ANALYSE SCATTER PLOT:

  - Quelques valeurs seulement avec un bon retour sur investissement.

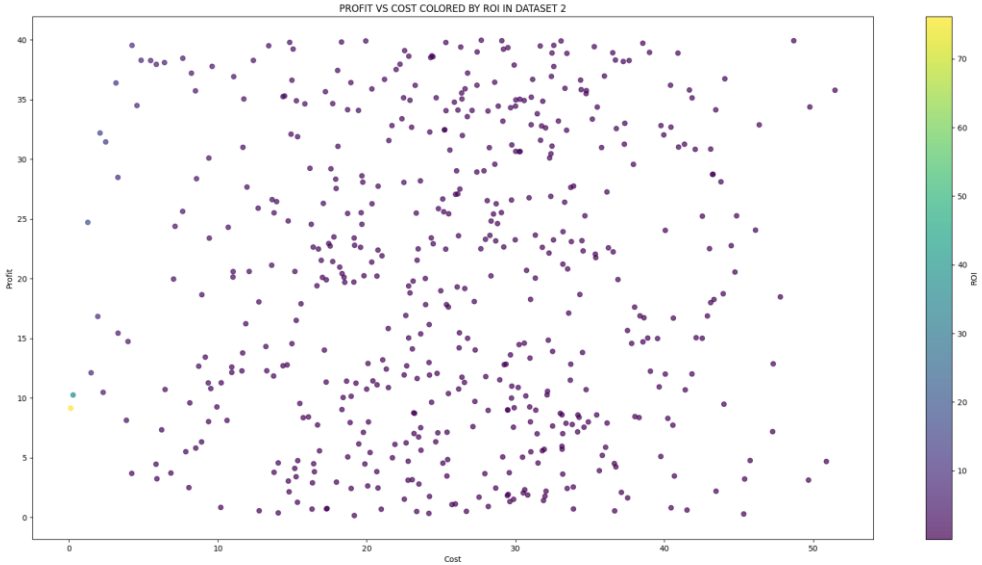
DATASET 1



- ANALYSE BOX PLOT :

  - La boîte des coûts est au dessus de la boîte des profits : coûts légèrement supérieurs aux profits.

  - La boîte des coûts montre que 50% des valeurs sont moins dispersées que les valeurs extrêmes.



- ANALYSE SCATTER PLOT :

  - Le *scatter-plot* montre bien la dispersion des valeurs.

  - Moins de 10% des valeurs ont un bon retour sur investissement (*ROI*).

DATASET 2

COMPARAISON

-

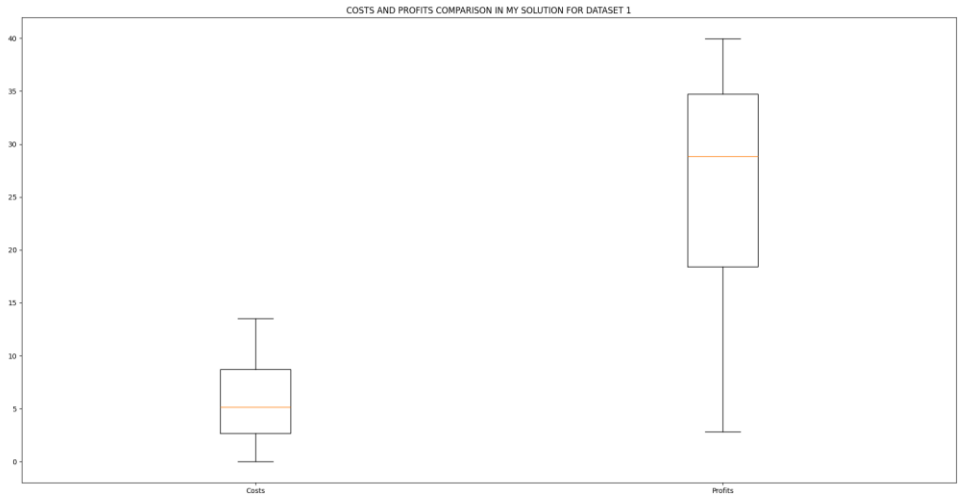
MA SOLUTION

VS

LA SOLUTION DE SIENNA

-

DATASET 1



MA SOLUTION

- OBSERVATIONS :

  - Boîte des profits au dessus de la boîte des coûts : profits avantageux par rapport aux coûts.

  - La boîte des coûts présente des valeurs peu dispersées.

  - La boîte des profits présente 75% des valeurs dispersées.

1 seule action

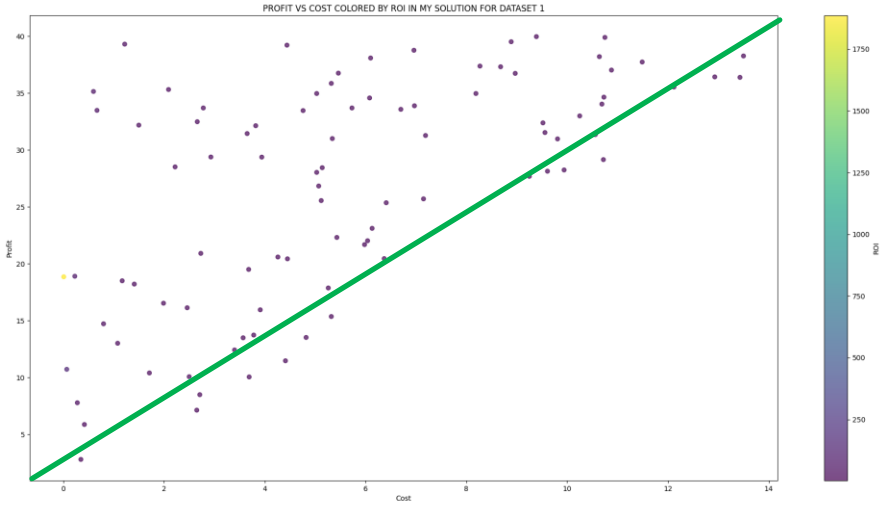
Nom	Coût	bénéfice	Profit
SHARE-GRUT	498,76€	7,90%	~~196,61€~~ 39,42€

SOLUTION DE SIENNA

- OBSERVATIONS :

  - Une seule action sélectionnée.

  - Profit faible et ne correspondant pas au *dataset*.



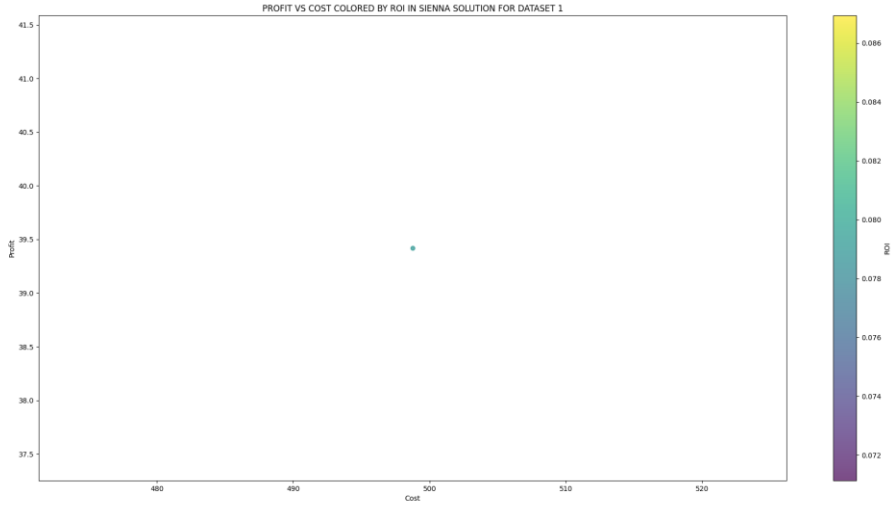
MA SOLUTION

- OBSERVATIONS :

  - Les valeurs sont présentes dans la partie supérieure gauche : ceci témoigne de coûts faibles et de profits élevés.

  - Retours sur investissement (*ROI*) élevés, voire très élevés compte tenu de coûts parfois très faibles.

  - Début de corrélation positive.



SOLUTION DE SIENNA

- OBSERVATIONS :

  - Retour sur investissement (*ROI*) très faible (**inférieur à 1**) compte tenu du coût exorbitant de cette seule action

COMPARAISON

-

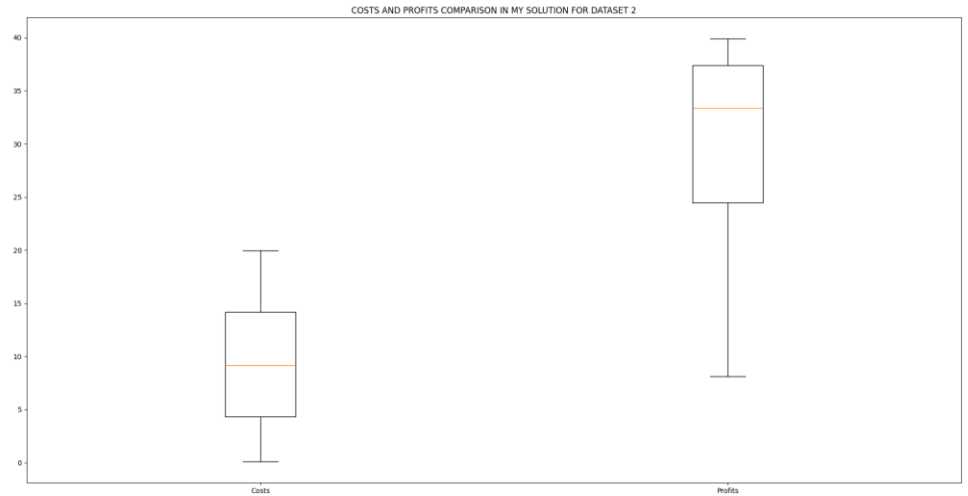
MA SOLUTION

VS

LA SOLUTION DE SIENNA

-

DATASET 2



MA SOLUTION

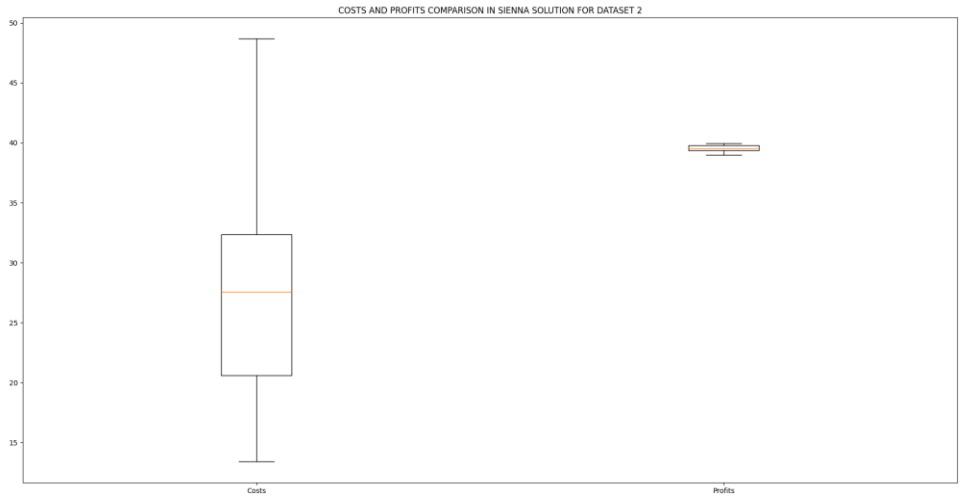
- OBSERVATIONS :

- La boîte des profits au dessus de la boîte des coûts : confirmation d'une solution très avantageuse en termes de profits.

- Valeurs des coûts peu dispersées.

- Valeurs des profits plus dispersées.

- Solution présentant de bons profits : 75% des valeurs de profits hautes voire très concentrées vers le max(25%) contre 25% vers le min



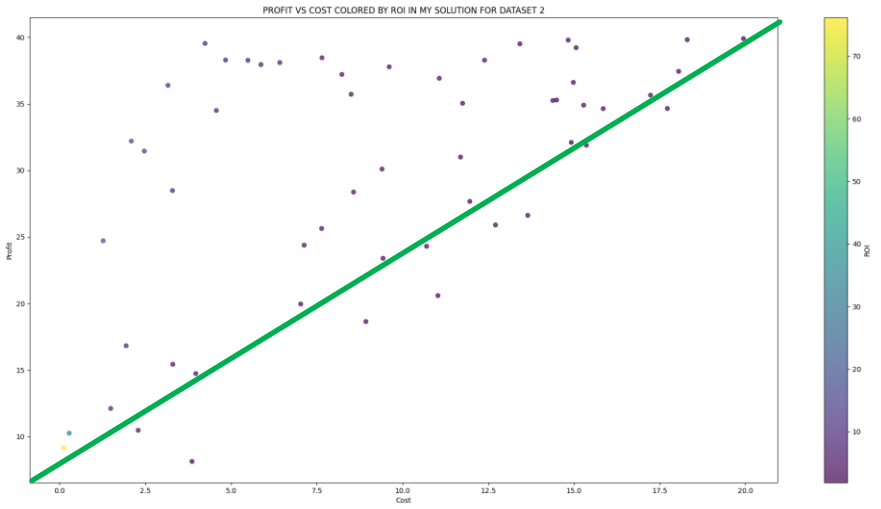
SOLUTION DE SIENNA

- OBSERVATIONS :

- La boîte des profits au dessus de la boîte des coûts : donc bons profits.

- Valeurs des profits extrêmement concentrées autour de la moyenne. Coûts dispersés avec 25% des valeurs dispersées proches du max : coûts parfois élevés des actions de cette solution.

- Solution présentant de bons profits mais compte-tenu de certains coûts élevés, le retour sur investissement (ROI) sera plus faible.



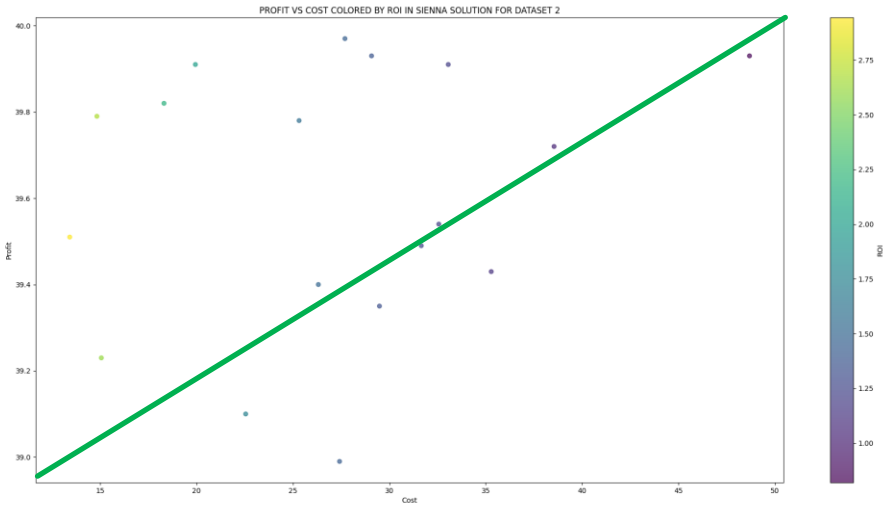
MA SOLUTION

- OBSERVATIONS :

  - Les valeurs de ma solution sont présentes dans la zone supérieure gauche : ceci témoigne d'une solution avec des coûts faibles et de bons profits. Corrélation positive.

  - Certaines valeurs ont un coût plus élevé mais compte tenu du profit élevé, les actions correspondantes sont sélectionnées.

  - Retours sur investissement (*ROI*) élevés de manière générale entre 10 et 70€.



SOLUTION DE SIENNA

- OBSERVATIONS :

  - Les valeurs de ma solution sont présentes dans la zone supérieure gauche : ceci témoigne d'une solution avec de bons profits. Corrélation positive.

  - Coûts plus élevés dans la solution de Sienna.

  - Retours sur investissement (*ROI*) plus faibles, inférieurs à 3€.

DATASET 1

- BILAN :

  - La solution de Sienna est mauvaise avec le choix d'une seule action au coût très élevé.

  - Ma solution est largement meilleure avec un retour sur investissement élevé.

DATASET 2

- BILAN :

  - La solution de Sienna est bonne mais ne l'emporte pas en raison de coûts élevés qui engendrent un *ROI* faible.

  - Ma solution est plus optimale avec un ROI entre 30 et 70€ contre une fourchette entre 1 et 3€ pour la solution de Sienna.

	Dataset 1		Dataset 2		
Solutions	Coût total	Profit Total	Coût total	Profit Total	
Solution de Sienna	498,76	39,42€	489,26€	712,80€	❌
Ma solution	499,92	2387,60€	499,84€	1609,58€	✅

LIMITES DU PROJET

- On ne tient pas compte du risque ni de la fiscalité des actions.
- On a vu que la programmation dynamique consomme plus de mémoire mais je n'en ai pas tenu compte dans la stratégie d'investissement.
- Modèle valable pour un marché avec peu de mouvement.

AMÉLIORATIONS POSSIBLES

- Tenir compte du risque des actions.
- Optimiser la mémoire, réduire le tableau de programmation dynamique
  - Algorithmes basés sur la technique *« diviser pour mieux régner »*
- Ajouter des *Datasets* plus volumineux

CONCLUSION

- Problème correctement modélisé
- Brute-force non exploitable
- Programmation dynamique retenue
- Ma solution plus performante que celle de Sienna