PROJET 7

ALGO INVEST & TRADE

ALGO INVEST & TRADE / PROJET /

CONTEXTE :

- AlgoInvest & Trade : une entreprise spécialisée dans l'aide à la décision en investissement.
- Projet : l'optimisation de stratégies d'investissement à court terme.
- Interlocutrice métier : Sienna, responsable des décisions d'achat.
- Supports :
  - Les décisions reposent sur des données d'actions issues de fichiers CSV.
  - Ces données peuvent être incomplètes ou incorrectes.

BESOINS :

- Disposer d'un outil d'aide à la décision fiable.
- Gérer des données historiques imparfaites.
- Respecter des contraintes strictes métier :
  - budget maximal de 500 €,
  - une action achetée au plus une fois,
  - aucune fraction d'action autorisée.

ALGO INVEST & TRADE / PROJET /

OBJECTIFS :

- Maximiser le profit après deux ans d'investissement.
- Proposer une solution **brute force** pour valider la logique.
- Proposer une solution **optimisée** pour traiter des volumes de données plus importants.
- Exploiter des fichiers CSV fournis comme supports de données.
- **Comparer les résultats avec les décisions d'investissement de Sienna.**

PLAN DE ROUTE :

- Présentation de la solution **brute force.**
- Limites de l'approche brute force.
- Présentation de la solution **optimisée.**
- un **schéma / pseudo-code.**
- **Comparaison des résultats** avec la stratégie de Sienna
- Conclusion et perspectives.

BRUTEFORCE.py

CONDITIONS :

- Chaque action ne peut être achetée qu'une seule fois (aucun doublon).
- Achat d'actions entières uniquement.
- Budget maximal strict : 500 €, vérifié pour chaque combinaison.

METHODE :

ENUMERATION EXHAUSTIVE

- Génération toutes combinaisons possibles d'actions, sans répétition.
- Pour chaque combinaison : calcul du coût total + validation si coût ≤ 500 €.
- Pour chaque combinaison valide : calcul du profit total + comparaison avec meilleur profit connu.
- La combinaison ayant le profit maximal sous contrainte budgétaire est conservée comme solution finale.
- Aucune décision locale : toutes les combinaisons admissibles sont évaluées avant de choisir la meilleure.

LIMITES :

- **Complexité exponentielle ($O(2^n)$).**
- **Temps d'exécution très élevé lorsque le nombre d'actions augmente.**
- **Impossible à utiliser sur de grands jeux de données , non adaptée à usage en prod.**

OPTIMIZED.py

+ Condition supplémentaire implicite :

- Budget maximal strict : 500 €, respecté globalement par la solution finale.

PROGRAMMATION DYNAMIQUE (SAC À DOS)

- Transformation du problème en **problème du sac à dos 0/1**.
- Le budget est découpé en **capacités intermédiaires** (de 0 à 500 €).
- Construction progressive d'une solution optimale :
  - pour chaque action,
  - pour chaque budget possible,
  - on décide **prendre ou ne pas prendre l'action**.
- À chaque étape :
  - le meilleur profit possible est mémorisé pour un budget donné.
- La solution finale est obtenue **SANS tester toutes les combinaisons explicitement**.
- Le résultat optimal est garanti par la **réutilisation de sous-solutions optimales**.

BRUTEFORCE.py

PSEUDO - CODE :

1. **La combinaison est reçue** depuis `iter_combinations`.
2. **Son coût total est calculé** (somme des coûts des actions).
3. **Le budget est vérifié** :
   - si le coût dépasse le budget → la combinaison est ignorée.
4. **Le profit total est calculé** (somme des profits des actions).
5. **Le profit est comparé** au meilleur profit connu :
   - si le profit est supérieur → la combinaison devient la meilleure solution.

Dans chaque combinaisons de 1; 2; 3; 4; 5 (…) actions : quel est le meilleur profit ?

SCHÉMAS :

TOUTES LES COMBIS

- A seul
- A+B
- A+B+C
- A+B+C+D
- A+B+C+D+E

PAS DE DOUBLONS

	A	B	C	D	E
A	-	✓	✓	✓	✓
B	✗	-	✓	✓	✓
C	✗	✗	-	✓	✓
D	✗	✗	✗	-	✓
E	✗	✗	✗	✗	-

OPTIMIZED.py

Le fil d'Ariane

est reconstruit en partant du budget final, puis en suivant keep[C] pour retrouver la dernière action utilisée, en soustrayant son coût et en répétant jusqu'à atteindre un budget nul.

Les doublons sont évités car chaque action met à jour les budgets en descendant, empêchant sa réutilisation dans la même itération

PROGRESSION COMBINAISONS, ACTION APRES ACTION / BUDGET :

Initialisation (avant toute action)

> DP[C] = profit maximal atteignable avec un budget C
> keep[C] = index de la dernière action utilisée pour atteindre DP[C] (-1 = aucune)

Budget C	DP[C] (meilleur profit possible)	keep[C] (dernière action utilisée)
1	0	-1
2	0	-1
3	0	-1

Après action A (i = 0, coût = 1, profit = 1)

Budget C	DP[C] (meilleur profit possible)	keep[C] (dernière action utilisée)
1	1	0
2	1	0
3	1	0

Après action B (i = 1, coût = 2, profit = 3)

Budget C	DP[C] (meilleur profit possible)	keep[C] (dernière action utilisée)
1	1	0
2	3	1
3	4	1

Après action C (i = 2, coût = 3, profit = 4)

Budget C	DP[C] (meilleur profit possible)	keep[C] (dernière action utilisée)
1	1	0
2	3	1
3	4	1

Profit C = profit A+B en profit > donc on garde A+B
> pas strictement supérieur

ALGO INVEST & TRADE / SIENNA / / OPTIMIZED /

SOURCES DES DONNEES :

Décisions d'achat basées sur des **données historiques existantes**.

Notre solution : exploitation directe des **fichiers CSV fournis**, avec nettoyage automatique des données invalides (coût ≤ 0, profit ≤ 0, dépassement de budget)..

METHODES DE DECISION :

- Utilise une stratégie issue d'analyses passées.
- Les règles exactes de sélection ne sont pas connues.
- Ne garantit pas une optimalité mathématique pour un budget donné.

Notre algorithme :

- **Brute force** : teste toutes les combinaisons admissibles → optimalité garantie.
- **Optimisé (sac à dos)** : calcule le meilleur profit possible pour chaque budget jusqu'à 500 € → optimalité garantie avec forte réduction du temps de calcul.

GESTION DES CONTRAINTES :

Contrainte	Sienna	Notre solution
Budget max 500 €	Oui	Oui
Pas de doublons	Non précisé	Garanti
Actions entières uniquement	Non précisé	Garanti
Données invalides	Non précisé	Filtrées automatiquement
Optimalité garantie	Non	Oui

ALGO INVEST & TRADE / CONCLUSIONS /

RESULTATS OBTENUS :

- Les **paniers proposés par notre algorithme** :
  - respectent strictement le budget,
  - génèrent un **profit supérieur ou équivalent** à celui de Sienna,
  - sont **justifiés algorithmiquement** (traçabilité complète via DP et KEEP).

VALEUR AJOUTEE DE NOTRE SOLUTION :

- Résultat **mathématiquement optimal**, pas heuristique.
- Algorithme **reproductible**, **explicable** et **vérifiable**.
- Adapté à l'augmentation du volume de données (version optimisée).
- Solution plus **robuste face aux données manquantes ou incorrectes**.

CONCLUSION DE LA COMPARAISON :

La stratégie de Sienna repose sur des décisions historiques, tandis que notre algorithme calcule systématiquement la meilleure solution possible sous contraintes, garantissant un choix optimal et traçable.