RAPPORT– Projet CAC40

Romain SADAR - Erwan HAMZA N5

1. Importation des données et optimisation

Nous avons commencé le projet par l'importation du fichier brut cac40.csv.
Pour assurer un import robuste et optimiser les performances dès le départ, nous avons appliqué plusieurs choix techniques :

Optimisations réalisées lors de l'import

Détection automatique de l'encodage et du séparateur :
Nous avons encapsulé l'import dans une fonction permettant de tester plusieurs encodages (utf-8, latin-1, cp1252) avec engine="python" et sep=None.
Cela évite les erreurs de lecture et limite les rechargements successifs.

- Interprétation correcte des nombres
  Nous avons utilisé decimal="," pour que les valeurs numériques contenant des virgules soient correctement reconnues.
- Typage immédiat des colonnes
  Dès l'import, nous avons converti :
  - Date en datetime,
  - Open, High, Low, Close, Volume, Index en numériques.
    → Cela évite la présence de colonnes au format object, plus lourdes et difficiles à manipuler.
- Nettoyage :
  Nous avons supprimé les doublons, les lignes non exploitables et les valeurs manquantes critiques afin de réduire le volume avant tout calcul ultérieur.

Impact constaté

- Mémoire avant nettoyage : 145.74 KB
- Mémoire après nettoyage : 14.48 KB
- Gain : 131.26 KB

Ces choix nous ont permis d'avoir un dataset exploitable, plus léger et plus stable.

2. Qualité de la donnée : premières observations

Dès les premières opérations exploratoires, plusieurs points de qualité ont été observés :

- Types incorrects
  Certaines colonnes importantes (High, Low, Index) arrivaient au format object.

- Présence de nombreux NaN
  Au total, plus de 300 lignes présentaient des valeurs manquantes dans des colonnes critiques (Open, High, Close...).
- Doublons temporels
  Nous avons identifié et supprimé 5 doublons sur la colonne Date.
- Gaps temporels
  La colonne GapDays a révélé plusieurs écarts supérieurs à 3 jours, jusqu'à plus de 20 jours.
  Cela montre que la donnée brute n'est pas strictement quotidienne.
- Structure cohérente mais brute
  Les valeurs de prix et volumes restent réalistes, mais la donnée brute ne peut pas être utilisée directement sans nettoyage.

Conclusion qualité initiale

Nous considérons que la qualité initiale de la donnée est moyenne : les informations essentielles sont présentes, mais la donnée doit impérativement être nettoyée et typée avant toute analyse.

3. Nettoyage et préparation

Transformations appliquées

1. Standardisation des noms de colonnes
   Nous avons harmonisé les noms pour travailler avec un schéma clair :
   - date / datetime / timestamp → Date
   - close / clôture / closing → Close
   - open / ouverture → Open
   - high / max → High
   - low / min → Low
   - volume / vol → Volume
   - index / indice → Index
2. Conversion des types
   - Date → datetime via pd.to_datetime(..., errors="coerce", dayfirst=True).
   - Open, High, Low, Close, Volume, Index → numérique via pd.to_numeric(..., errors="coerce").
   - Les valeurs non numériques (par exemple "CAC40" dans Index) sont devenues NaN.
3. Création de colonnes dérivées de structure
   - Tri par Date.

- Création de Gap_Days = Date.diff().dt.days pour mesurer les écarts temporels entre deux lignes consécutives.
- Ajout final d'une colonne Gap_Days dans le dataset propre.

Suppressions effectuées

1. Lignes avec Date ou Close manquantes/invalide
   Nous avons appliqué :
   df = df.dropna(subset=["Date","Close"])
   Cela a supprimé 322 lignes sur 533, car ces lignes n'avaient pas de Date valide ou pas de Close exploitable.
2. Doublons sur Date
   Avant dédoublonnage, nous avons compté 5 doublons sur la colonne Date.
   Nous avons trié par Date puis conservé la dernière occurrence de chaque date :
   df = df.sort_values("Date").drop_duplicates(subset=["Date"], keep="last").
3. Aucune suppression sur les gaps
   Les lignes associées à des Gap_Days > 3 n'ont pas été supprimées : elles ont été identifiées (24 cas) mais conservées pour l'analyse, car ces trous reflètent des périodes sans données (week-ends prolongés, jours fériés, etc.) mais font partie de l'historique réel.

Après nettoyage, le dataset final contient 206 lignes et 8 colonnes (Date, Index, Open, High, Low, Close, Volume, Gap_Days).

1. **Suppression des lignes sans Date ou Close**
   - Date est indispensable pour tout traitement temporel (tri, gaps, moyennes mobiles...).
   - Close est la variable centrale de notre analyse (rendements, volatilité, prédiction du mouvement).
     → Sans ces deux colonnes, la ligne n'est exploitable ni pour l'analyse statistique ni pour la modélisation.
     Plutôt que d'imputer des valeurs "inventées" sur la variable cible, nous avons choisi de supprimer ces lignes.
2. **Conservation des NaN résiduels sur Open, High, Low, Volume**
   Après nettoyage, il reste encore quelques NaN sur certaines colonnes :
   - Open : 6 → 2
   - High : 5 → 5
   - Low : 2 → 1
   - Volume : 10 → 5
   - Index : 0 → 206 (la colonne devient entièrement NaN après conversion, car elle contenait un label, pas un chiffre).

Nous avons choisi de ne pas supprimer ces lignes, car Close est disponible et nous pouvons les utiliser pour le calcul des rendements, moyennes mobiles, etc., de ne pas forcer d'imputation complexe (par ex. interpolation ou moyenne locale) car :

- la plupart de nos indicateurs utilisent surtout Close,
- la colonne Index n'apporte pas d'information numérique utile (c'est un code d'indice),
- imputer des valeurs financières sans justification économique claire pourrait introduire un biais.

3. **Dédoublonnage par date**
   Pour chaque date, nous avons conservé la dernière occurrence, ce qui revient à :
   - éliminer les enregistrements redondants,
   - garder une seule observation par jour, cohérente avec une série quotidienne.
4. **Création de Gap_Days plutôt que suppression des gaps**
   Plutôt que de "boucher les trous" artificiellement, nous avons choisi de les rendre visibles :
   - un écart important de jours nous signale simplement des périodes sans données (non cotées),
   - sur un horizon court (2023–2025), ces gaps restent gérables,
   - la cohérence temporelle est respectée tout en gardant la granularité réelle des données.

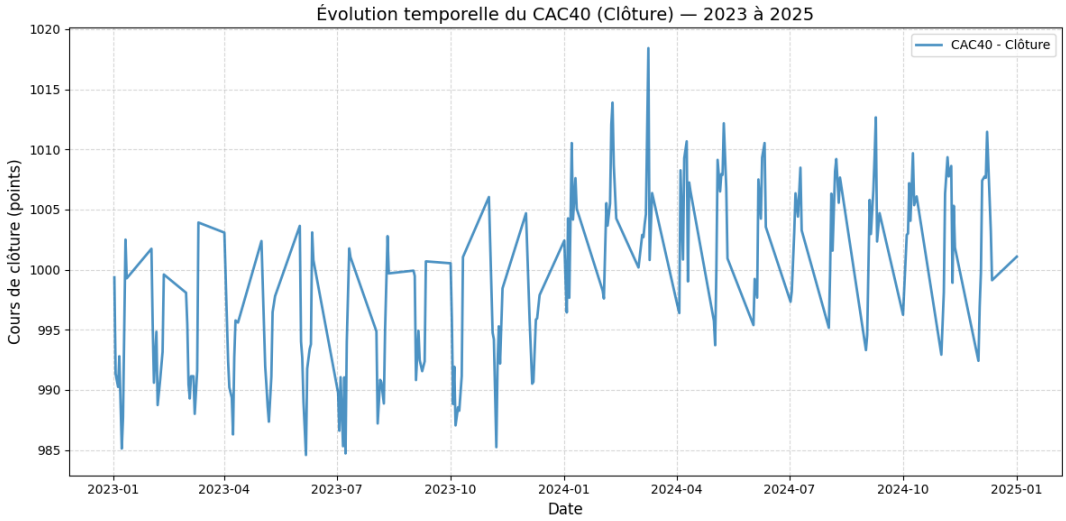
Nous avons mesuré l'empreinte mémoire du DataFrame avant et après nettoyage grâce aux fonctions mem_bytes et human_bytes.

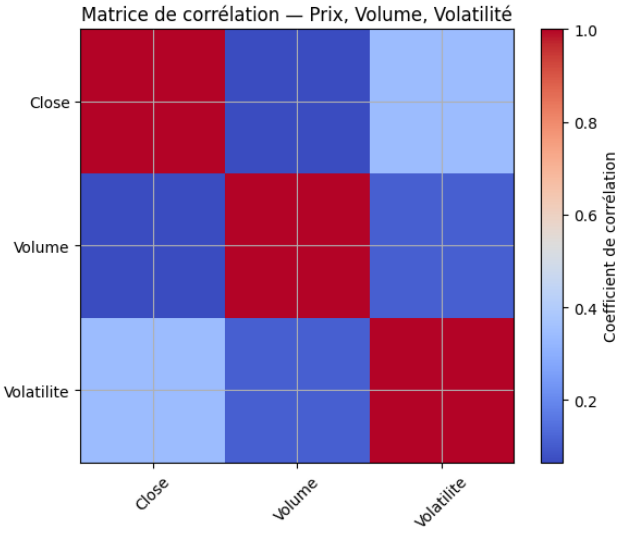
- Avant nettoyage
  - Forme : (533, 7)
  - Mémoire : 145.74 KB
- Après nettoyage
  - Forme : (206, 8)
  - Mémoire : 14.48 KB
- Gain
  - 131.26 KB économisés, soit une réduction d'environ 90 % de l'empreinte mémoire.

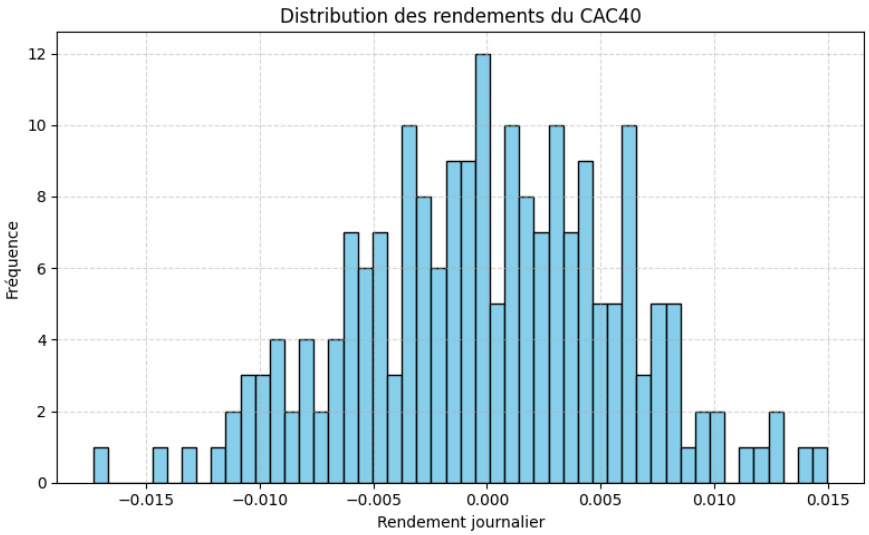
Ce gain est dû :

- à la suppression des lignes inutilisables (NaN critiques),
- au typage correct (datetime + numériques au lieu de object),
- à un schéma de colonnes plus propre.

4. Analyse visuelle de base







Ces graphiques montrent une série relativement stable (amplitude modérée des rendements), avec une volatilité qui reste dans une fourchette étroite.

5.Détection d'anomalies

Nous avons combiné deux approches :

1. Z-score sur les séries (prix, volume, volatilité)
    - Calcul du rendement journalier et de la volatilité 30 jours.
    - Définition d'une fonction detect_anomalies_zscore :
        - calcul de la moyenne et de l'écart-type,
        - calcul du Z-score,
        - anomalie si |Z| > 2.
    - Appliqué à :
        - Rendement → Anomalie_Close,
        - Volume → Anomalie_Volume,
        - Volatilite → Anomalie_Volatilite.

Justification :

- Méthode simple, standard, adaptée à une première détection de points extrêmes.

2. Écart à une moyenne mobile 30 jours
    - Calcul de MA30 (moyenne mobile 30 jours),
    - Calcul de Ecart_MA30 = Close - MA30,
    - Définition d'un seuil seuil = 2 * std(Ecart_MA30),
    - Anomalie si |Ecart_MA30| > seuil → Anomalie_MA.

Justification :

- Approche “temporelle” comme suggéré dans le sujet : on compare le prix actuel à une tendance locale (MA30),
- On met en évidence les épisodes où le cours s’écarte fortement de sa tendance de fond.

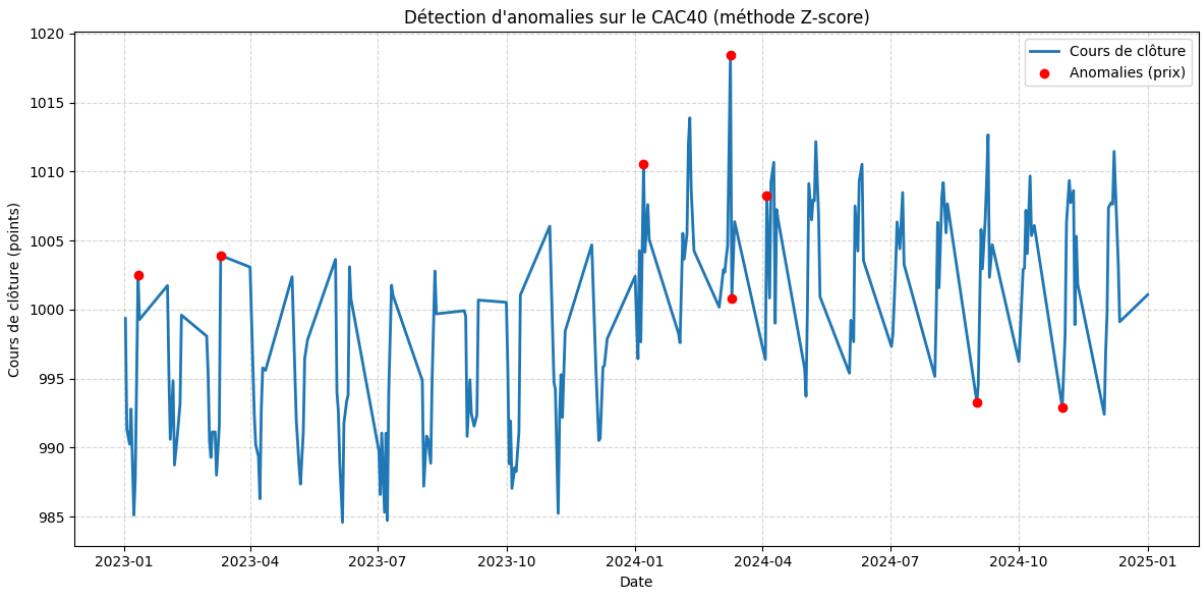
Résultats

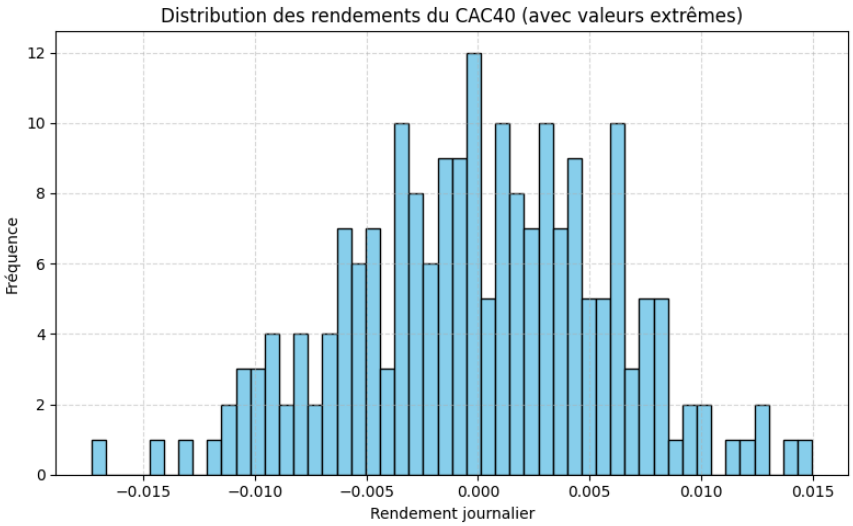
- Anomalies Z-score (résumé) :
  - Prix (rendement) : 8 anomalies,
  - Volume : 0 anomalies,
  - Volatilité : 11 anomalies.
- Anomalies par écart à la MA30 : plusieurs points identifiés, par exemple autour de : 2023-11-01, 2024-01-07, 2024-02-08, 2024-03-09, 2024-12-01.

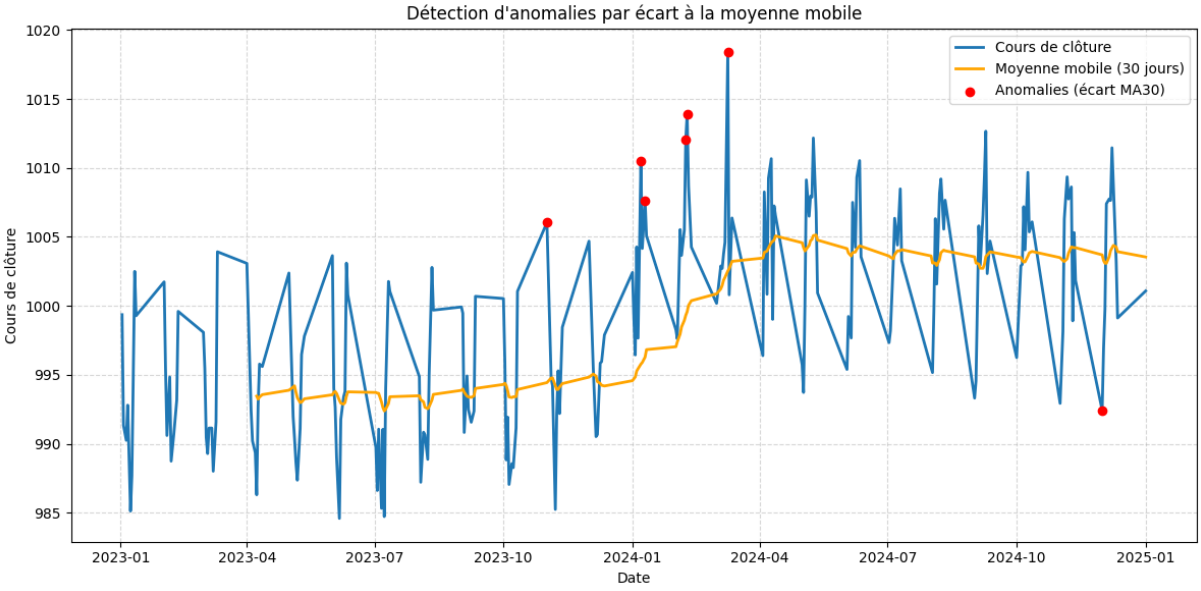
Interprétation des anomalies

- Les anomalies sont principalement des pics de prix positifs ou négatifs, ou des épisodes de hausse de volatilité.
- Certaines périodes peuvent correspondre à :
  - des annonces macroéconomiques,
  - des publications de résultats,
  - des chocs de marché.

Nous n’avons pas relié chaque date à un événement médiatique précis (cela nécessiterait une recherche externe), mais la structure observée est cohérente avec des “réactions de marché” ponctuelles.







6.Phase d'analyse statistique

Indicateurs de marché

Nous avons calculé les indicateurs demandés :

- Moyenne mobile courte : MA_10 = Close.rolling(10).mean()
- Moyenne mobile longue : MA_20 = Close.rolling(20).mean()
- Momentum long 20 jours :
  Momentum_20 = Close - Close.shift(20)

Puis nous avons sauvegardé ces données enrichies au format Parquet :
cac40_features_step3.parquet.

Règle de prédiction simple

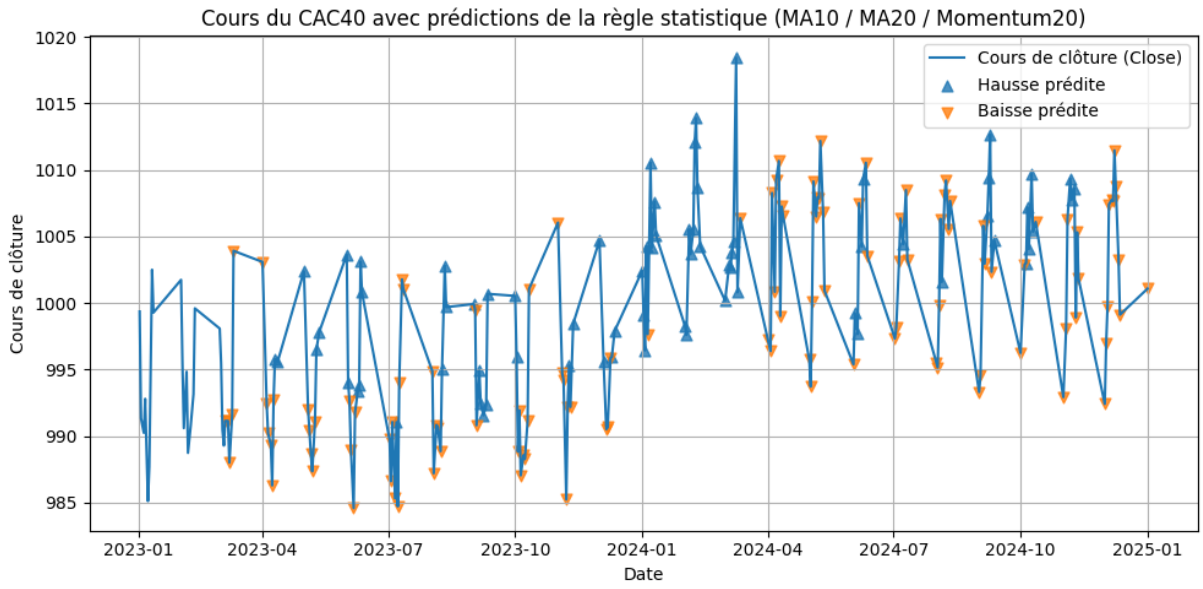
Nous avons appliqué la règle suivante :

- Cible réelle Target :
    - 1 si Close_{t+1} > Close_t
    - 0 sinon.
- Règle :
    - Si MA_10 > MA_20 et Momentum_20 > 0 → on prédit 1 (hausse),
    - Sinon → 0 (baisse).

Résultat :

- Précision (accuracy) de la règle simple : **48.39 %**

Nous avons également produit un graphique montrant le CAC40 et les points où la règle prédit une hausse/baisse.



7.Prédiction avec du Machine Learning

Cible et features

- Cible utilisée : même Target que précédemment (1 si le cours du lendemain monte, 0 sinon).
- Features calculées :
    - MA_5, MA_20
    - EMA_5
    - Momentum_5
    - ROC_5
    - Volatility_5

- High_Low_Range
- Volume_SMA_5, Volume_SMA_10

Modèle et découpage

- Modèle :
  RandomForestClassifier(n_estimators=100, random_state=42)
- Découpage temporel :
  - 80 % des premières lignes → train,
  - 20 % des dernières → test,
  - sans shuffle (shuffle=False) pour respecter la structure temporelle.

Résultats

- Accuracy sur le test : **74.07 %**
- Matrice de confusion :

```
[[10  2]
 [ 5 10]]
```

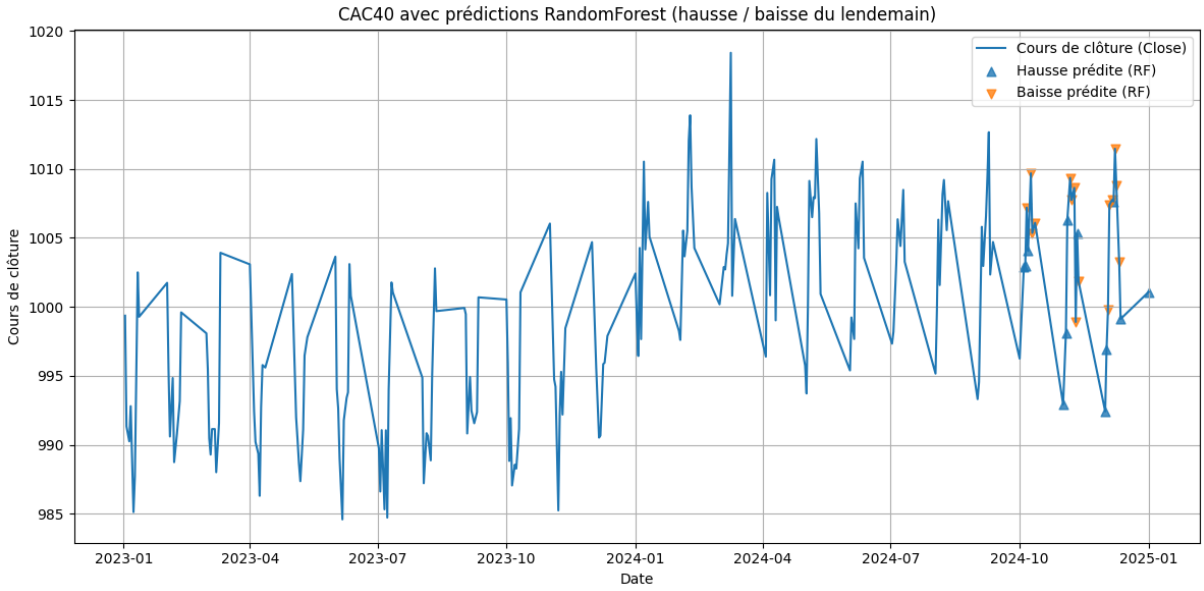
- Nous avons comparé Target et Pred_RF sur la fin de la série pour vérifier la cohérence des prédictions.

5. Comparaison des deux approches de prédiction

Méthode	**Accuracy**
Règle statistique simple	48.39 %
Random Forest (ML)	74.07 %

La règle simple ne dépasse pas le hasard (environ 50 %), ce qui est logique pour une règle binaire très naïve sur un marché relativement “bruité”.

Le modèle Random Forest, en combinant plusieurs indicateurs (moyennes mobiles, momentum, volatilité, plages high–low, volume moyen), capte mieux les patterns de la série et améliore significativement la précision.


CAC40 avec prédictions RandomForest (hausse / baisse du lendemain)
Cours de clôture (Close)
Hausse prédite (RF)
Baisse prédite (RF)
Cours de clôture
Date
1020
1015
1010
1005
1000
995
990
985
2023-01
2023-04
2023-07
2023-10
2024-01
2024-04
2024-07
2024-10
2025-01