Optimisation de Portefeuille avec le Modèle de Markowitz

ABDOULAYE TANGARA

Student in MSc in Quantitative and Computable Economics

Contacts

LinkedIn

Mon GitHub

Mon Portfolio

abdoulayetangara722@gmail.com

Introduction

Dans le cadre de mon apprentissage en finance quantitative et en gestion des risques, j'ai étudié les principes fondamentaux de la théorie moderne du portefeuille, notamment le modèle de Markowitz. Ce modèle, qui repose sur l'optimisation du couple rendement-risque, constitue une approche rigoureuse pour la sélection d'actifs financiers.

L'objectif de ce projet est de mettre en application ces concepts théoriques en construisant un portefeuille optimisé à partir de données réelles. À travers cette étude, je souhaite non seulement approfondir ma compréhension des mécanismes sous-jacents à la diversification et à la minimisation du risque, mais aussi développer des compétences pratiques en manipulation de données financières et en programmation.

Ainsi, ce travail consistera à collecter des données boursières, analyser les rendements des actifs, estimer la matrice de covariance, et résoudre le problème d'optimisation afin de déterminer un portefeuille efficient. Cette démarche me permettra d'évaluer la pertinence du modèle dans un contexte réel et d'en explorer les éventuelles limites.

0.1 Modélisation financière

1. Chargement des modules nécessaires

[92]:
```python
import numpy as np # Calcul mathématique
import pandas as pd
from tabulate import tabulate # Formatage de tableau
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.optimize as sco # Technique de modélisation
import yfinance as yf # Collecte de données sur Yahoo Finance
from datetime import datetime
```

2. Collecte des données

a. Selection des 40 meilleures capitalisation de CAC40

[93]:
```python
# Collectte des Tickers
url = "https://en.wikipedia.org/wiki/CAC_40"
cac40 = pd.read_html(url)[4]
tickers = cac40["Ticker"].to_list()
info_tickers = cac40[[ "Ticker", "Company", "Sector"]]
print(f"Les informations sur les entreprises du CAC40 sont : \n{info_tickers}")
```

```
Les informations sur les entreprises du CAC40 sont :
      Ticker                Company                  Sector
0      AC.PA                  Accor       Consumer Services
1      AI.PA            Air Liquide         Basic Materials
2     AIR.PA                 Airbus             Industrials
3      MT.AS          ArcelorMittal         Basic Materials
4      CS.PA                    Axa      Financial Services
5     BNP.PA            BNP Paribas      Financial Services
6      EN.PA               Bouygues             Industrials
7     CAP.PA              Capgemini              Technology
8      CA.PA              Carrefour      Consumer Defensive
9     ACA.PA        Crédit Agricole      Financial Services
10     BN.PA                 Danone      Consumer Defensive
11    DSY.PA      Dassault Systèmes              Technology
12   EDEN.PA                Edenred             Industrials
13   ENGI.PA                  Engie               Utilities
14     EL.PA       EssilorLuxottica              Healthcare
15    ERF.PA     Eurofins Scientific              Healthcare
16    RMS.PA                 Hermès       Consumer Cyclical
17    KER.PA                 Kering       Consumer Cyclical
18     OR.PA                L'Oréal      Consumer Defensive
19     LR.PA                Legrand             Industrials
20     MC.PA                   LVMH       Consumer Cyclical
21     ML.PA               Michelin             Industrials
22    ORA.PA                 Orange  Communication Services
23     RI.PA          Pernod Ricard      Consumer Defensive
```

24	PUB.PA	Publicis	Communication Services
25	RNO.PA	Renault	Consumer Cyclical
26	SAF.PA	Safran	Industrials
27	SGO.PA	Saint-Gobain	Industrials
28	SAN.PA	Sanofi	Healthcare
29	SU.PA	Schneider Electric	Industrials
30	GLE.PA	Société Générale	Financial Services
31	STLAP.PA	Stellantis	Consumer Cyclical
32	STMPA.PA	STMicroelectronics	Technology
33	TEP.PA	Teleperformance	Communication Services
34	HO.PA	Thales	Industrials
35	TTE.PA	TotalEnergies	Energy
36	URW.PA	Unibail-Rodamco-Westfield	Real Estate
37	VIE.PA	Veolia	Industrials
38	DG.PA	Vinci	Industrials
39	VIV.PA	Vivendi	Communication Services

b. Collecte du prix à la cloture

[94]:
```python
# Collecte des données
start = datetime(2020,1,1)
end = datetime.now()
cac40_data = yf.download(tickers=tickers, start=start, end=end)["Close"]
cac40_data.head(5)
```

```
[*********************100%***********************]  40 of 40 completed
```

[94]:
```
Ticker          AC.PA    ACA.PA      AI.PA      AIR.PA      BN.PA     BNP.PA  \
Date
2020-01-02  40.105446  9.573108  95.134499  128.991776  62.225372  37.128826
2020-01-03  39.577240  9.515021  94.834160  129.494125  62.763680  36.685329
2020-01-06  38.655277  9.420636  94.195923  128.933823  62.881443  36.505154
2020-01-07  38.751312  9.398853  93.782951  127.523323  62.292648  36.491302
2020-01-08  38.799335  9.442416  93.858040  129.822601  61.030991  36.574455

Ticker          CA.PA      CAP.PA      CS.PA      DG.PA  ...     SAN.PA  \
Date                                                     ...
2020-01-02  12.773987  102.722450  19.155062  83.949806  ...  74.554527
2020-01-03  12.825184  102.814957  19.162588  83.428490  ...  75.017288
2020-01-06  12.825184  101.103683  18.910547  83.008072  ...  75.488319
2020-01-07  12.927582  101.149918  19.027164  82.436295  ...  75.223885
2020-01-08  12.944649  100.872421  19.019640  84.017075  ...  75.480057

Ticker         SGO.PA  STLAP.PA   STMPA.PA      SU.PA      TEP.PA     TTE.PA  \
Date
2020-01-02  32.953613  3.626625  23.870199  83.409286  195.199768  35.638062
2020-01-03  32.378918  3.626625  23.667986  83.194168  195.378342  36.042274
2020-01-06  32.075977  3.626625  23.138391  83.086609  194.663971  36.564205
2020-01-07  31.946783  3.626625  23.716131  82.746025  195.199768  36.324986
2020-01-08  31.986879  3.626625  23.687244  82.961128  198.592957  36.419224

Ticker      URW.PA     VIE.PA    VIV.PA
Date
2020-01-02     NaN  19.276802  8.635487
2020-01-03     NaN  19.373466  8.655223
2020-01-06     NaN  19.260689  8.513765
2020-01-07     NaN  19.091524  8.503899
2020-01-08     NaN  18.994858  8.471001

[5 rows x 40 columns]
```

3. Analyse exploratoire des données

[95]:
```python
# Analyse de la dimension des données
print("Analyse de la dimension des données :\n")
print(f"Les nombres de lignes est de : {cac40_data.shape[0]} \n")
print(f"Le nombre de colonne est de :  {cac40_data.shape[1]} \n")
```

```
Analyse de la dimension des données :

Les nombres de lignes est de : 1329

Le nombre de colonne est de :  40
```

[ ]:
```python
# Cherche de valeurs manquantes
print(f"Données avant gestion des valeurs manquantes : \n")
print(cac40_data.isnull().sum())


# Gestion des valeurs manquantes ou missing values
cac40_data.drop(columns="URW.PA", axis=1, inplace=True)
print(f"Données après gestions des VM \n")
print(cac40_data.isnull().sum())
```

[96]:
```python
# Analyse descriptives des actifs
print(f"Analyse descriptive des actifs")
cac40_data.describe().T
```

```
Analyse descriptive des actifs
```

[96]:

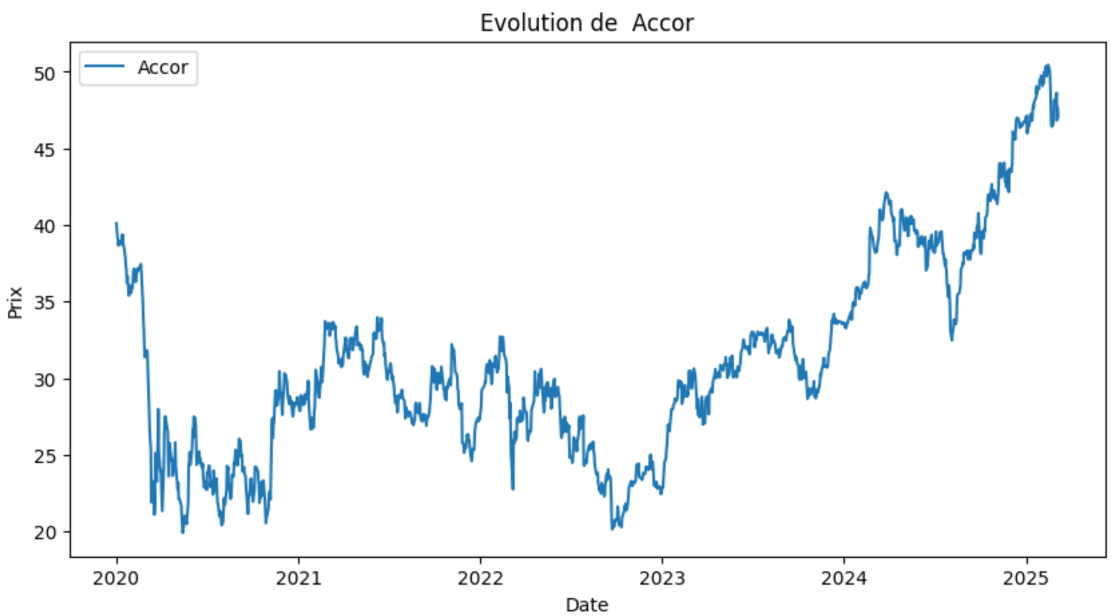
Ticker	count	mean	std	min	25%	\
AC.PA	1329.0	31.061872	6.597255	19.899063	26.487268	
ACA.PA	1329.0	9.643278	2.611507	4.369432	7.828506	
AI.PA	1329.0	128.000952	25.117673	74.500755	107.770012	
AIR.PA	1329.0	112.625243	28.375404	47.400997	95.563469	
BN.PA	1329.0	53.357230	5.905306	40.783173	48.980347	
BNP.PA	1329.0	46.268740	12.334437	17.095333	39.138805	
CA.PA	1329.0	14.486672	1.656564	10.516994	13.299759	
CAP.PA	1329.0	157.102119	35.553066	53.114117	141.722000	
CS.PA	1329.0	22.982204	6.913644	9.399976	18.289850	
DG.PA	1329.0	88.689711	14.666820	47.928078	78.039856	
DSY.PA	1329.0	37.028360	5.916763	21.150549	33.150002	
EDEN.PA	1329.0	43.873078	7.267253	28.170000	38.639992	
EL.PA	1329.0	157.413976	39.480780	85.370842	128.393906	
EN.PA	1329.0	28.323192	3.291434	17.722338	26.481192	
ENGI.PA	1329.0	11.382469	2.582780	6.434699	9.174883	
ERF.PA	1329.0	66.080221	17.681936	37.932003	53.554047	
GLE.PA	1329.0	20.831152	5.204630	8.961896	18.846771	

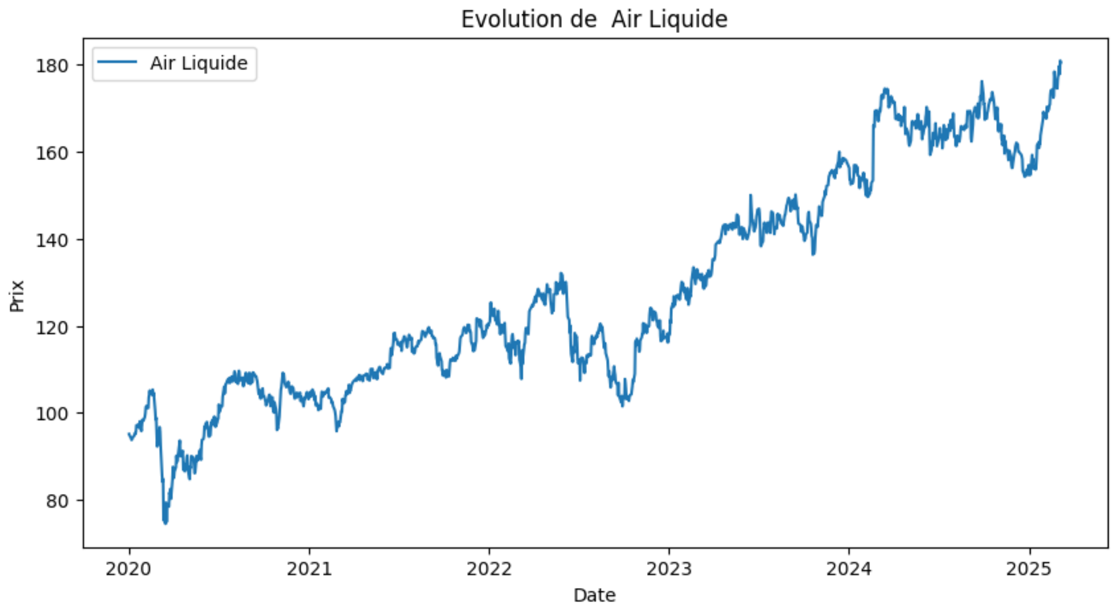
HO.PA	1329.0	107.452311	33.632115	50.129253	76.013649
KER.PA	1329.0	459.618631	117.762787	206.616669	387.354065
LR.PA	1329.0	80.135204	13.486446	44.424168	69.393066
MC.PA	1329.0	612.430652	143.557006	267.396973	534.093506
ML.PA	1329.0	27.449648	4.913246	15.127179	23.950975
MT.AS	1329.0	21.394802	5.723742	6.054872	20.020493
OR.PA	1329.0	344.700345	60.412341	191.168549	301.393097
ORA.PA	1329.0	8.800575	1.064217	6.425508	7.835805
PUB.PA	1329.0	58.060131	25.960650	16.253555	41.087036
RI.PA	1329.0	152.738126	25.900784	97.000000	129.691345
RMS.PA	1329.0	1456.961127	562.886453	509.738434	981.730957
RNO.PA	1329.0	32.914997	8.254341	13.896367	27.940767
SAF.PA	1329.0	134.580764	43.906599	51.654900	102.869095
SAN.PA	1329.0	83.218305	9.842733	59.539501	75.500229
SGO.PA	1329.0	52.018573	17.677772	15.982747	38.470360
STLAP.PA	1329.0	12.107839	5.191218	3.626625	10.693386
STMPA.PA	1329.0	33.573058	7.316938	13.472259	27.190655
SU.PA	1329.0	148.078227	48.016566	60.302776	115.885307
TEP.PA	1329.0	207.946092	83.684091	80.080002	121.440804
TTE.PA	1329.0	43.428731	13.069318	15.799302	30.806929
URW.PA	485.0	63.991642	13.313858	41.096058	48.699265
VIE.PA	1329.0	23.623148	4.600152	13.109076	20.505756
VIV.PA	1329.0	8.977548	1.795608	1.816810	8.260459

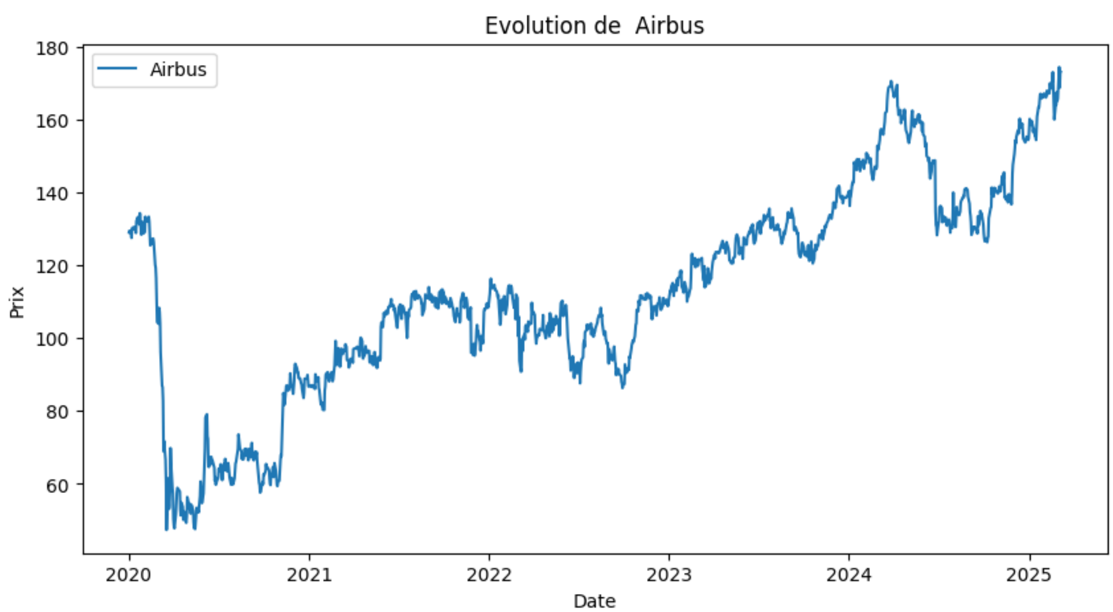
	50%	75%	max
Ticker			
AC.PA	30.011860	33.810417	50.459999
ACA.PA	9.359613	10.971296	16.580000
AI.PA	119.841942	150.041016	180.919998
AIR.PA	111.026878	131.880005	174.380005
BN.PA	52.054249	57.060951	72.339996
BNP.PA	46.177555	55.829861	77.169998
CA.PA	14.720167	15.606474	18.880699
CAP.PA	164.713089	181.584625	223.129745
CS.PA	21.862164	26.906042	38.650002
DG.PA	85.464355	101.150002	118.849998
DSY.PA	36.572365	40.459232	55.351994
EDEN.PA	43.794212	48.655895	60.507473
EL.PA	156.256454	173.330215	295.799988
EN.PA	28.190001	30.219999	36.660000
ENGI.PA	10.738558	13.665205	17.389999
ERF.PA	60.515282	76.673294	121.239822
GLE.PA	21.660000	23.844999	42.014999
HO.PA	111.804733	134.077118	247.199997
KER.PA	475.428375	529.785706	718.989868
LR.PA	81.554825	89.797768	110.699997
MC.PA	632.256958	702.400024	871.436646

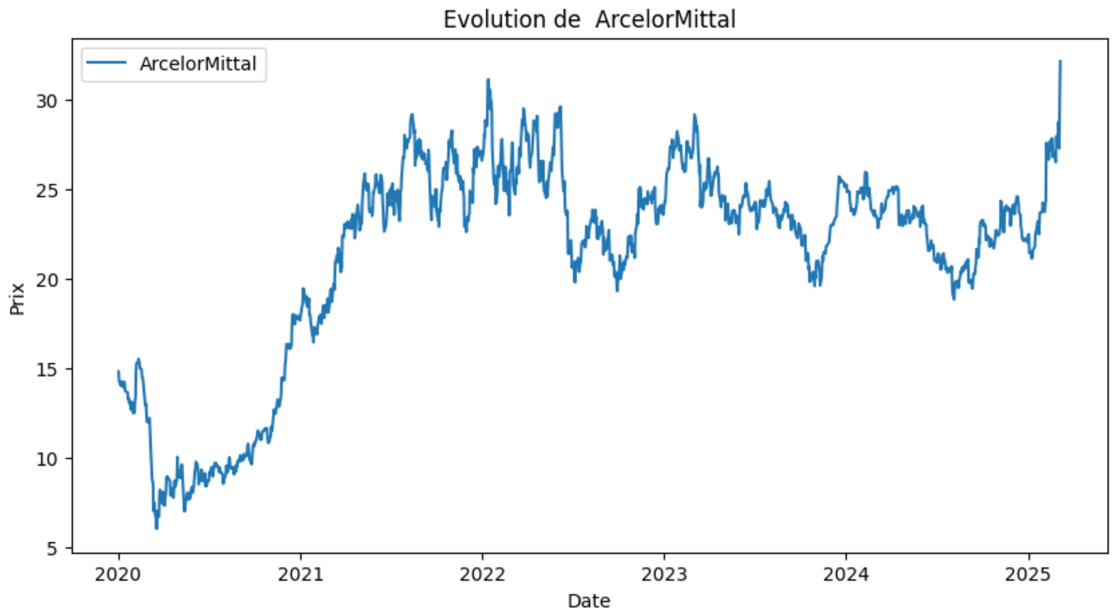
ML.PA	27.279108	31.010818	38.369999
MT.AS	23.286318	24.966825	32.169998
OR.PA	347.146057	394.089203	456.899994
ORA.PA	8.801795	9.759664	11.650000
PUB.PA	49.376736	72.202301	107.650002
RI.PA	153.759628	174.192245	203.255585
RMS.PA	1338.104614	1936.831543	2835.500000
RNO.PA	32.950146	37.243214	53.980000
SAF.PA	117.667038	156.113632	260.700012
SAN.PA	82.128159	91.395721	109.620003
SGO.PA	50.952217	59.587048	105.900002
STLAP.PA	12.612000	14.380169	25.422567
STMPA.PA	34.308689	39.552654	49.764664
SU.PA	139.356934	163.444397	271.700012
TEP.PA	210.064178	283.982208	364.515015
TTE.PA	43.937698	55.930000	66.739632
URW.PA	68.361061	75.300003	83.099998
VIE.PA	24.648605	27.638332	31.510000
VIV.PA	9.239752	10.185000	12.458085

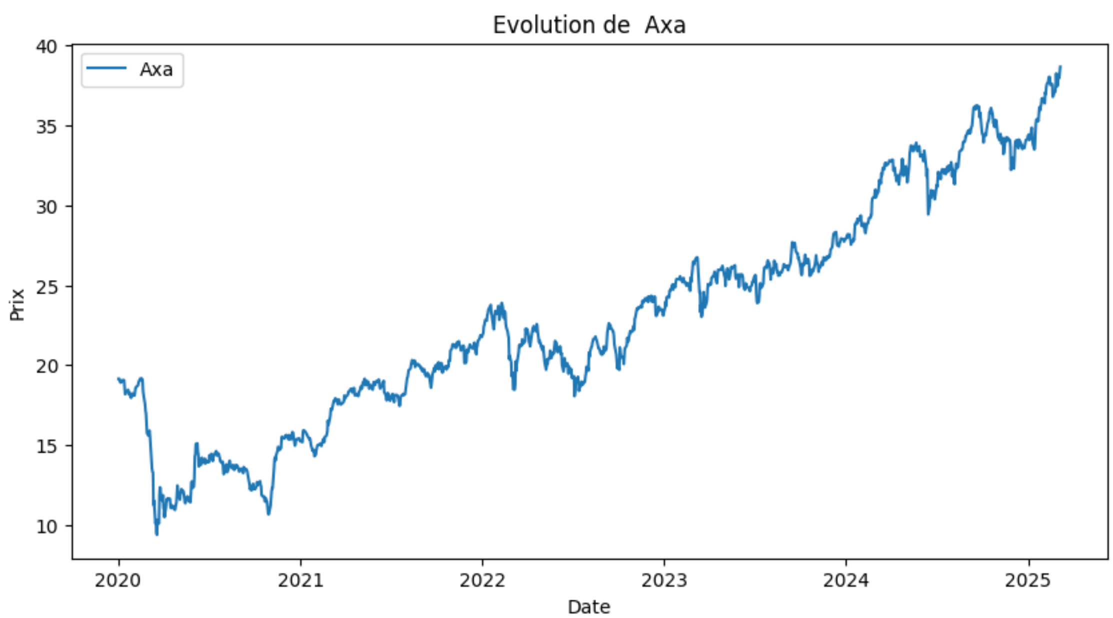
[97]:
```
col = info_tickers.iloc[:,0].to_list()
col.remove("URW.PA")
for i in col:
    plt.figure(figsize=(10,5))
    plt.plot(cac40_data[i],␣
 ↪label=info_tickers[info_tickers["Ticker"]==i]["Company"].values[0])
    plt.title(f"Evolution de ␣
 ↪{info_tickers[info_tickers['Ticker']==i]['Company'].values[0]}")
    plt.xlabel("Date")
    plt.ylabel("Prix")
    plt.legend()
    plt.show()
```

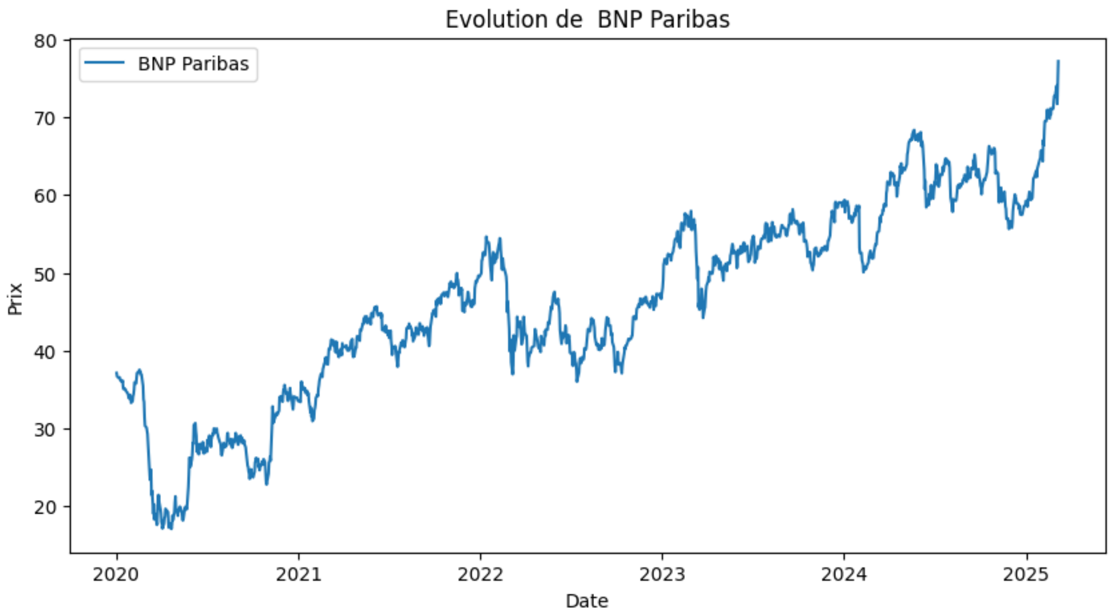


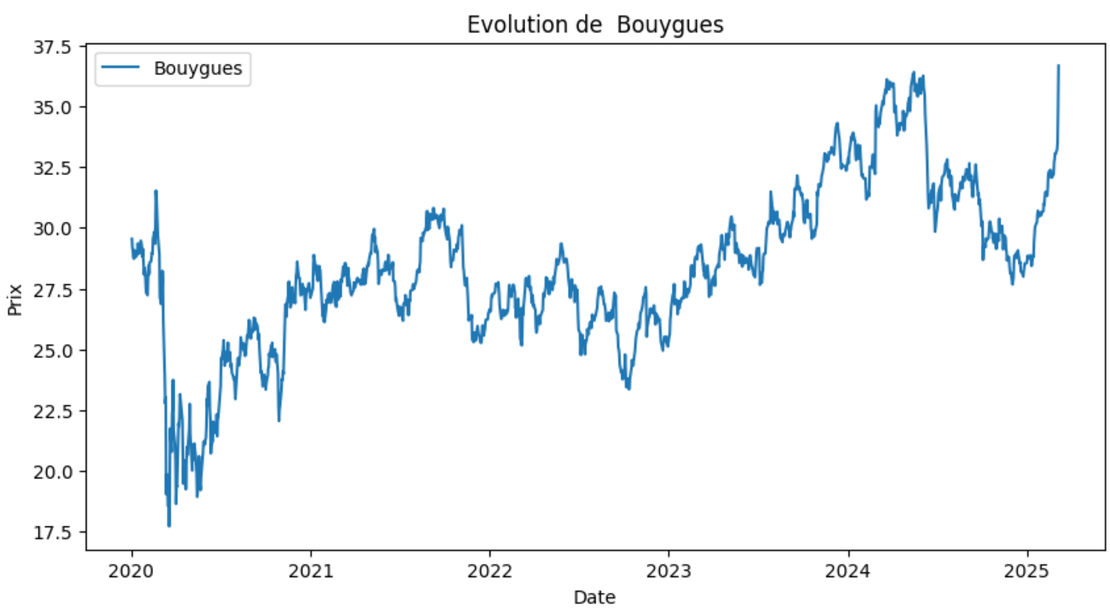

Evolution de Air Liquide
Air Liquide
Prix
180
160
140
120
100
80
2020
2021
2022
2023
2024
2025
Date

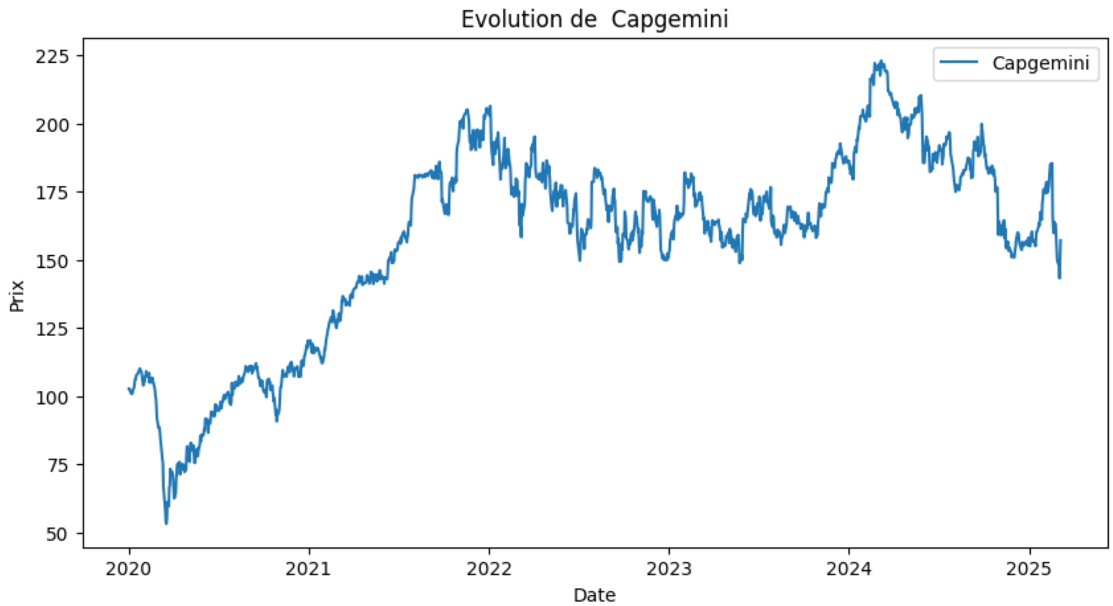

Evolution de Airbus
Airbus
Prix
180
160
140
120
100
80
60
2020
2021
2022
2023
2024
2025
Date

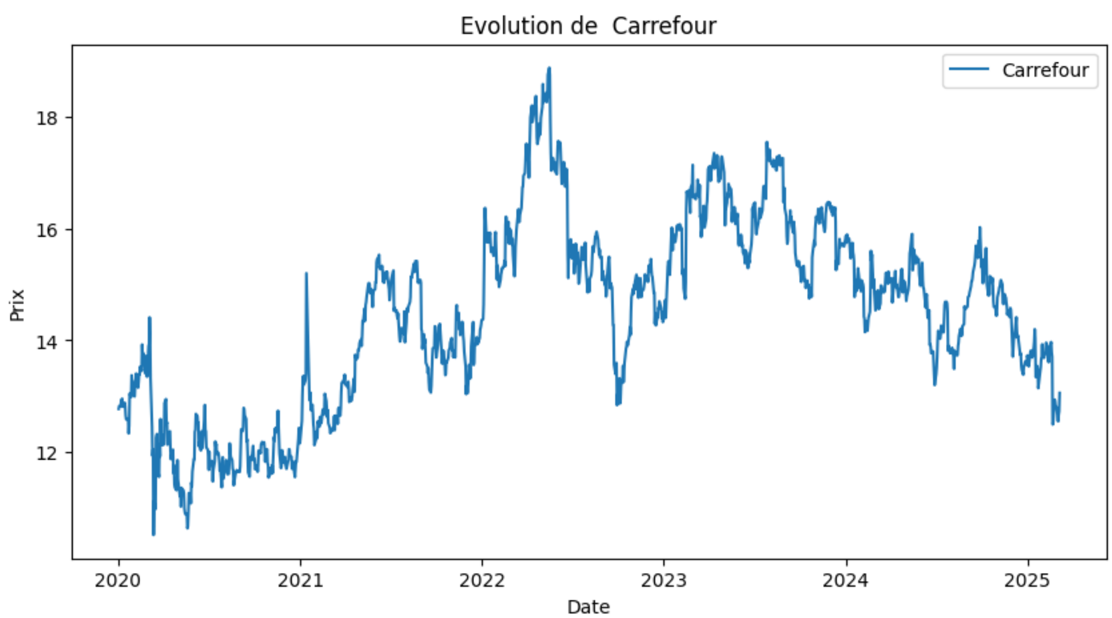

Evolution de ArcelorMittal
ArcelorMittal
Prix
Date
30
25
20
15
10
5
2020
2021
2022
2023
2024
2025

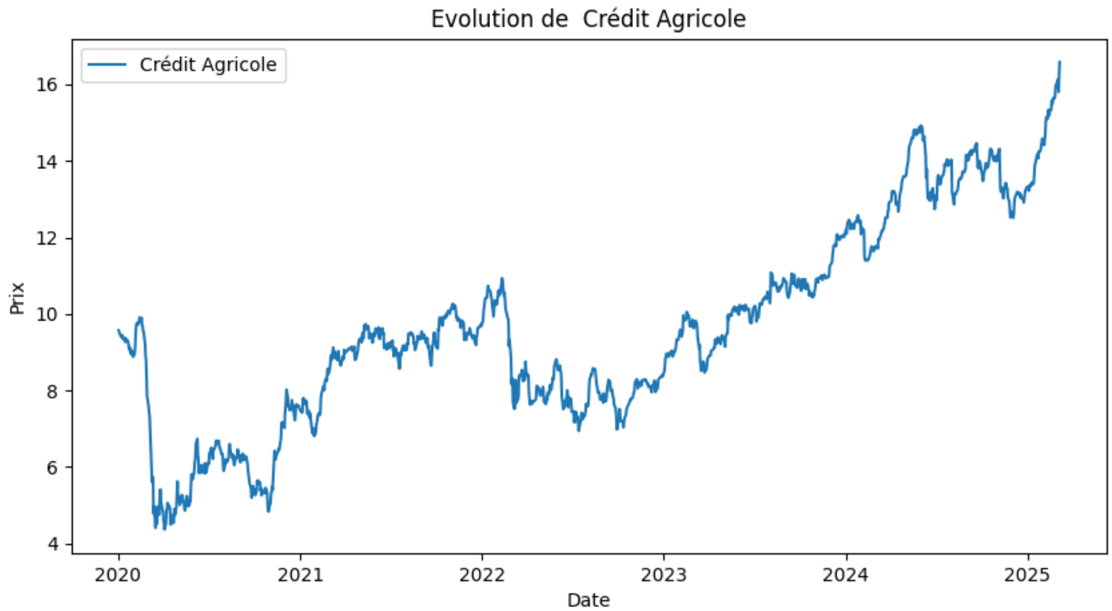

Evolution de Axa
Axa
Prix
Date
40
35
30
25
20
15
10
2020
2021
2022
2023
2024
2025

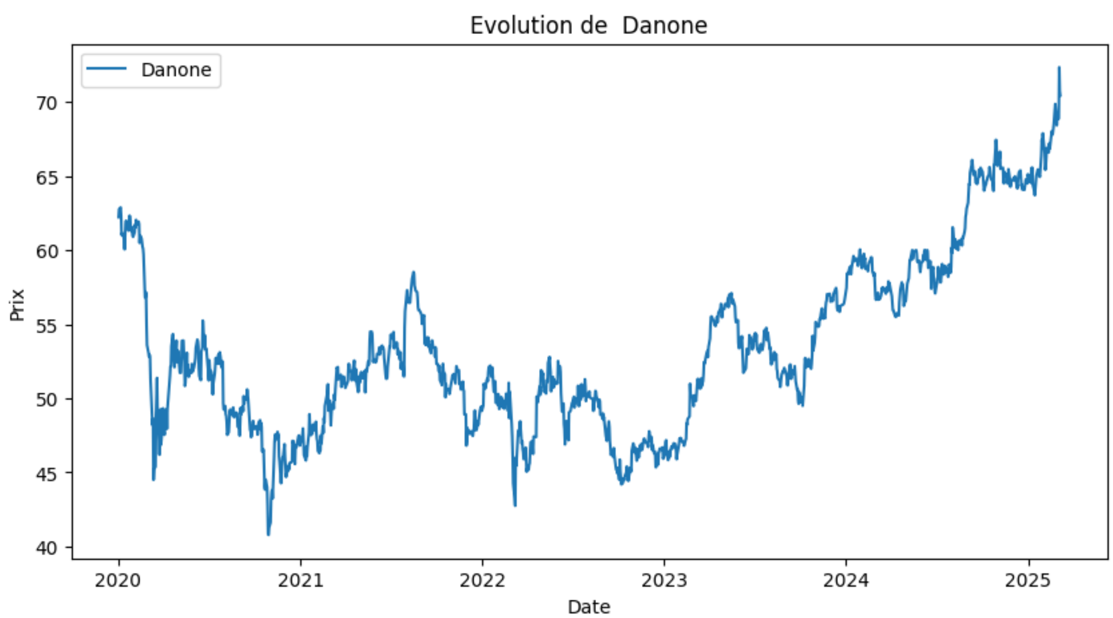

Evolution de BNP Paribas
BNP Paribas
Prix
Date
80
70
60
50
40
30
20
2020
2021
2022
2023
2024
2025

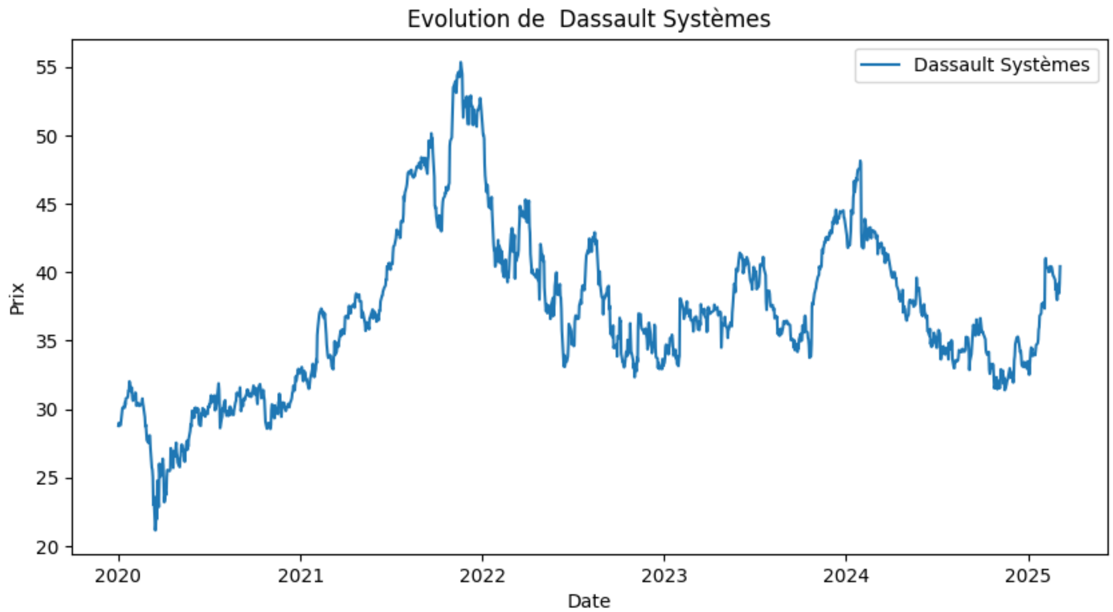

Evolution de Bouygues
Bouygues
Prix
Date
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
2020
2021
2022
2023
2024
2025

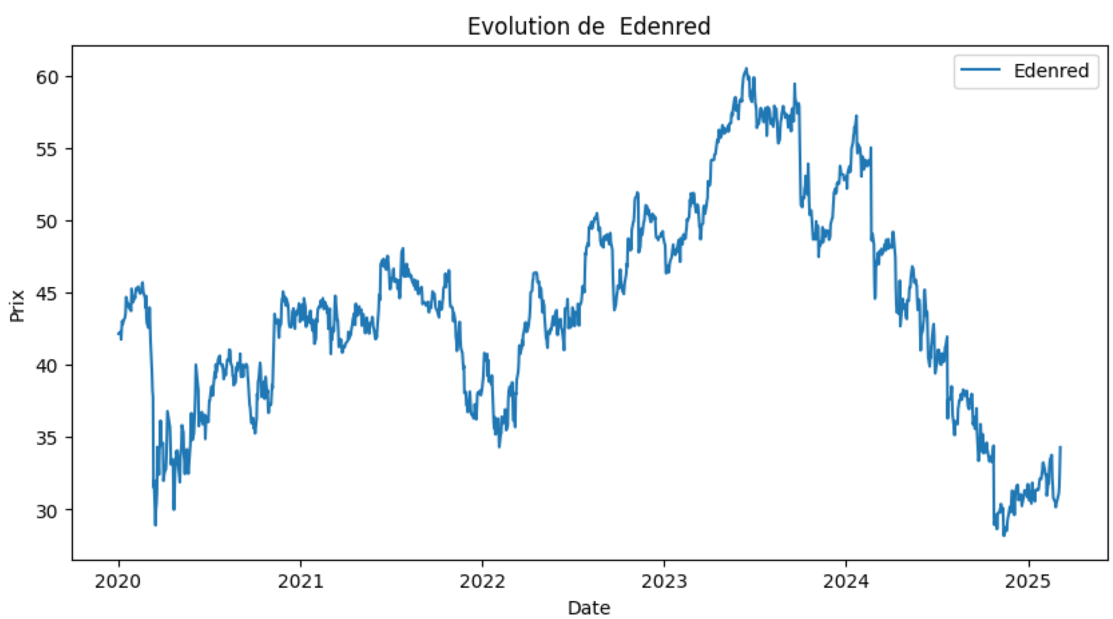

Evolution de Capgemini
Capgemini
225
200
175
150
125
100
75
50
Prix
2020
2021
2022
2023
2024
2025
Date

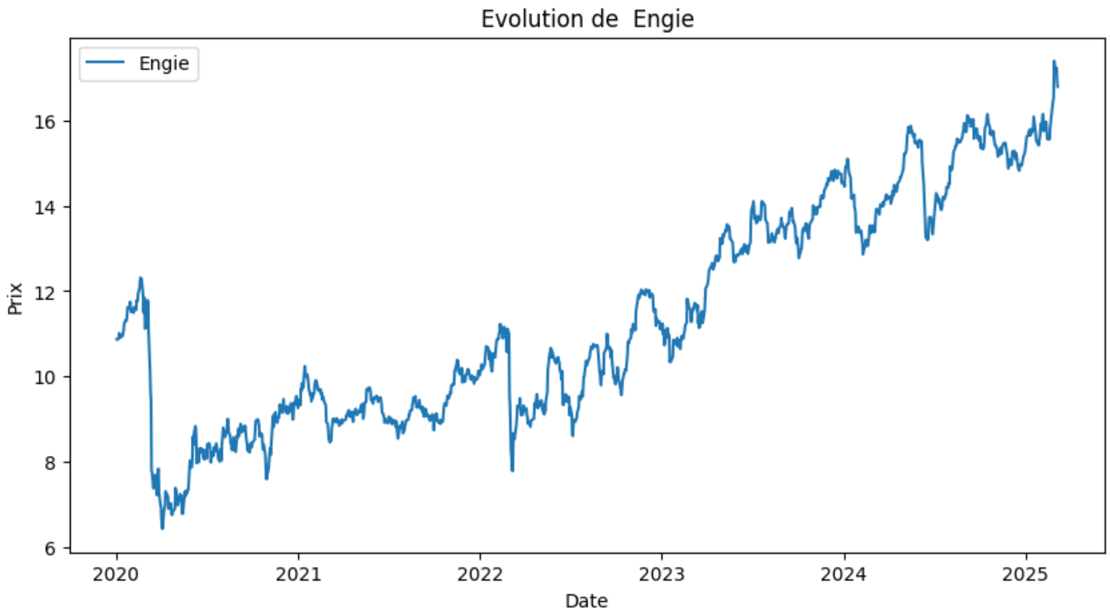

Evolution de Carrefour
Carrefour
18
16
14
12
Prix
2020
2021
2022
2023
2024
2025
Date

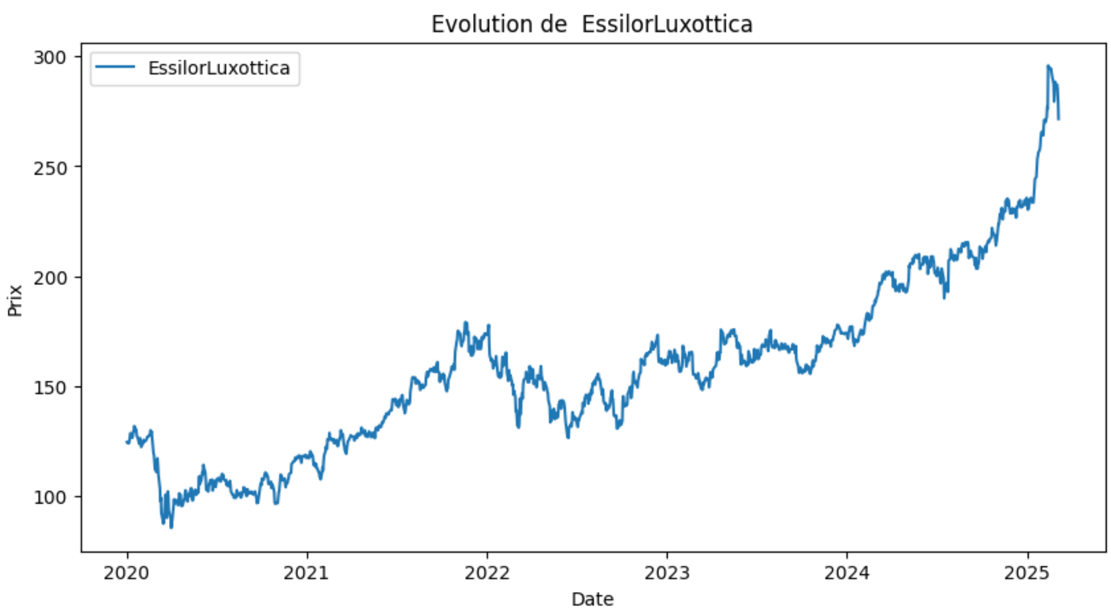

Evolution de Crédit Agricole
Crédit Agricole
Prix
Date
4
6
8
10
12
14
16
2020
2021
2022
2023
2024
2025

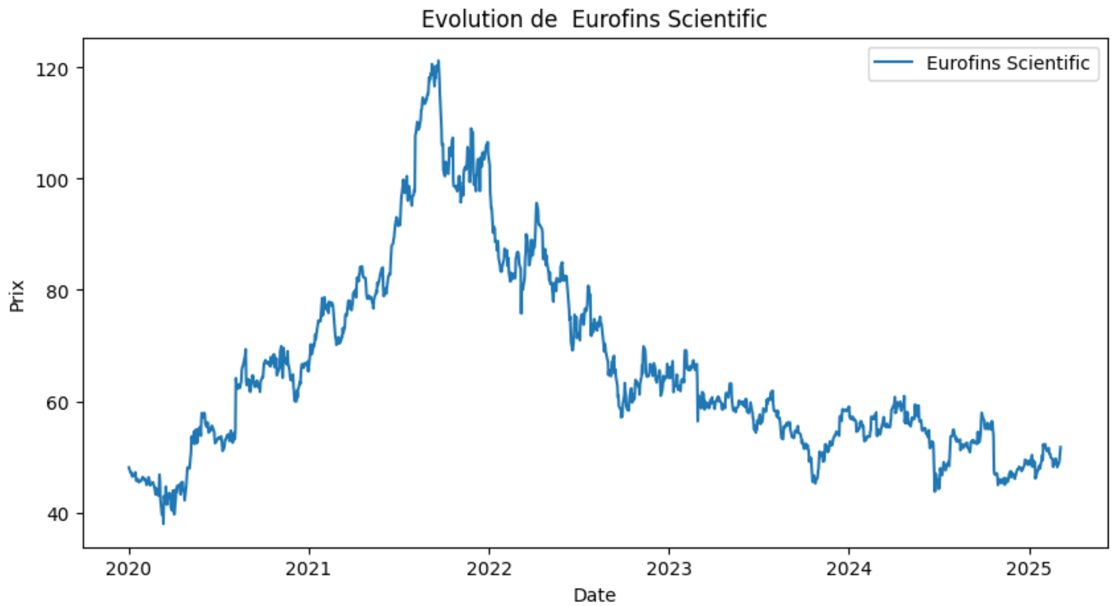

Evolution de Danone
Danone
Prix
Date
40
45
50
55
60
65
70
2020
2021
2022
2023
2024
2025

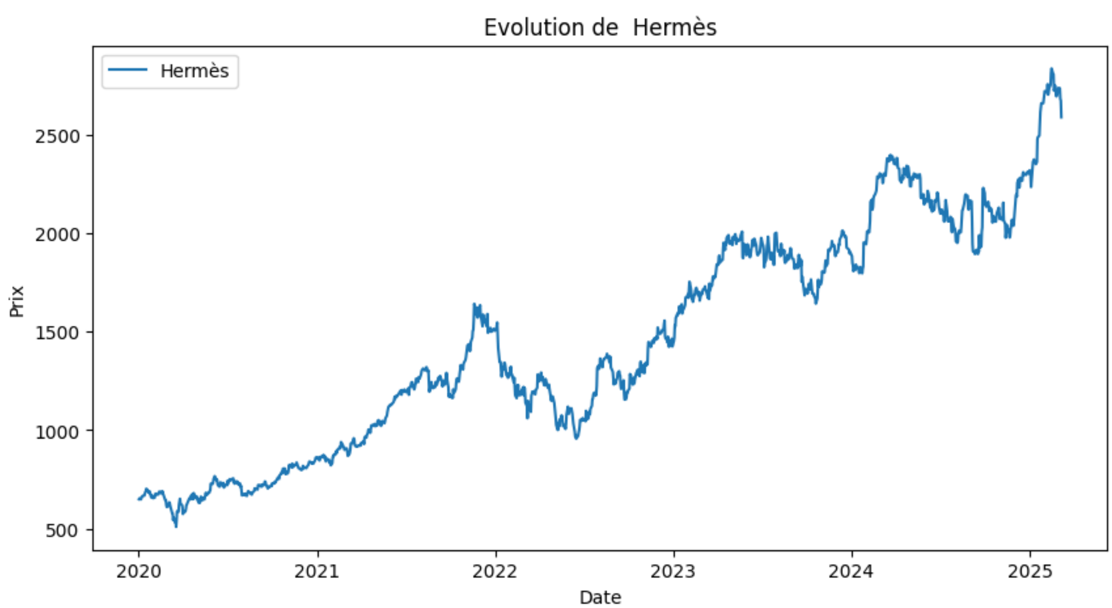

Evolution de Dassault Systèmes
Dassault Systèmes
55
50
45
40
35
30
25
20
Prix
2020
2021
2022
2023
2024
2025
Date

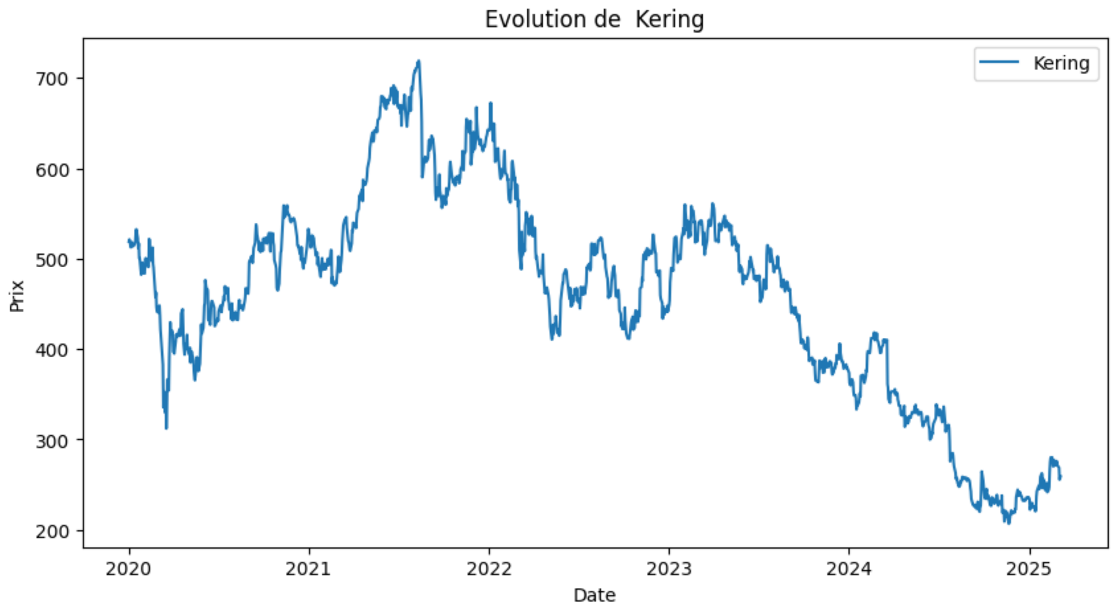

Evolution de Edenred
Edenred
60
55
50
45
40
35
30
Prix
2020
2021
2022
2023
2024
2025
Date

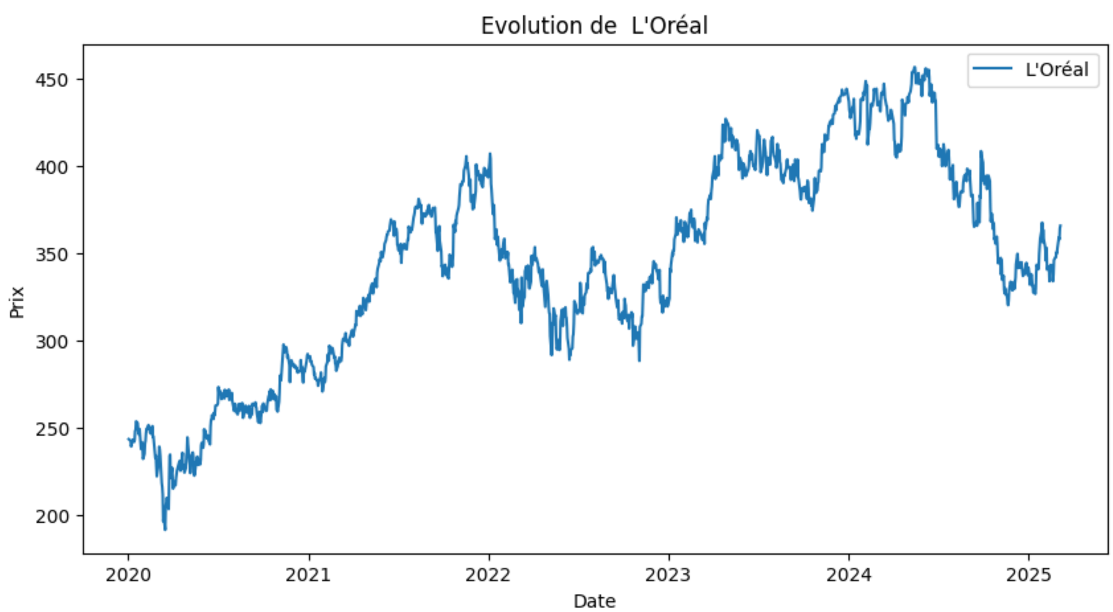

Evolution de Engie
Engie
16
14
12
10
8
6
Prix
2020
2021
2022
2023
2024
2025
Date

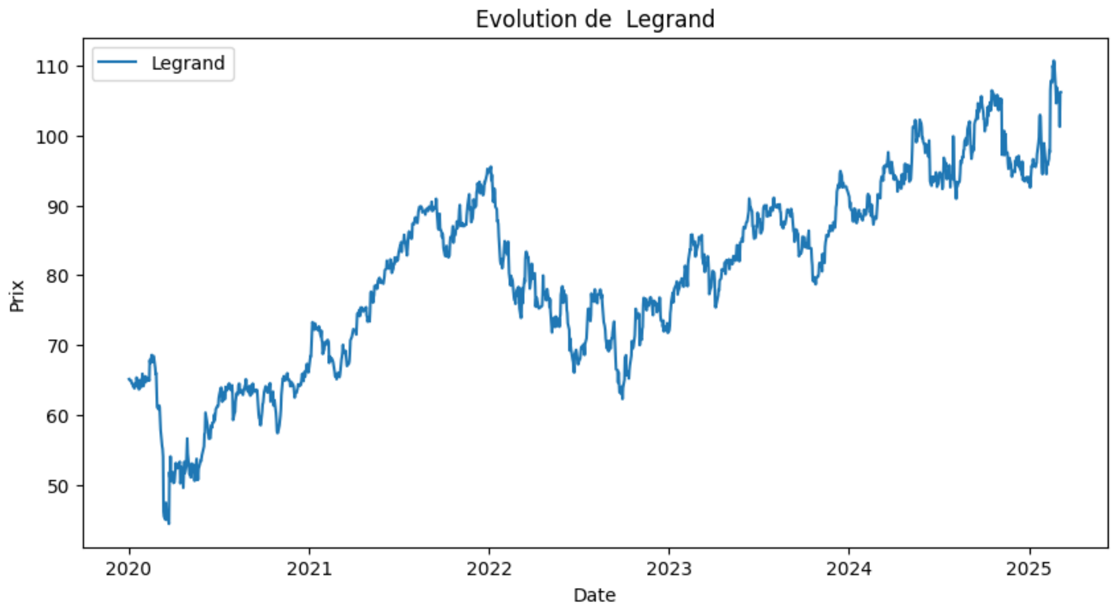

Evolution de EssilorLuxottica
EssilorLuxottica
300
250
200
150
100
Prix
2020
2021
2022
2023
2024
2025
Date


Evolution de Eurofins Scientific
Eurofins Scientific
Prix
Date
120
100
80
60
40
2020
2021
2022
2023
2024
2025

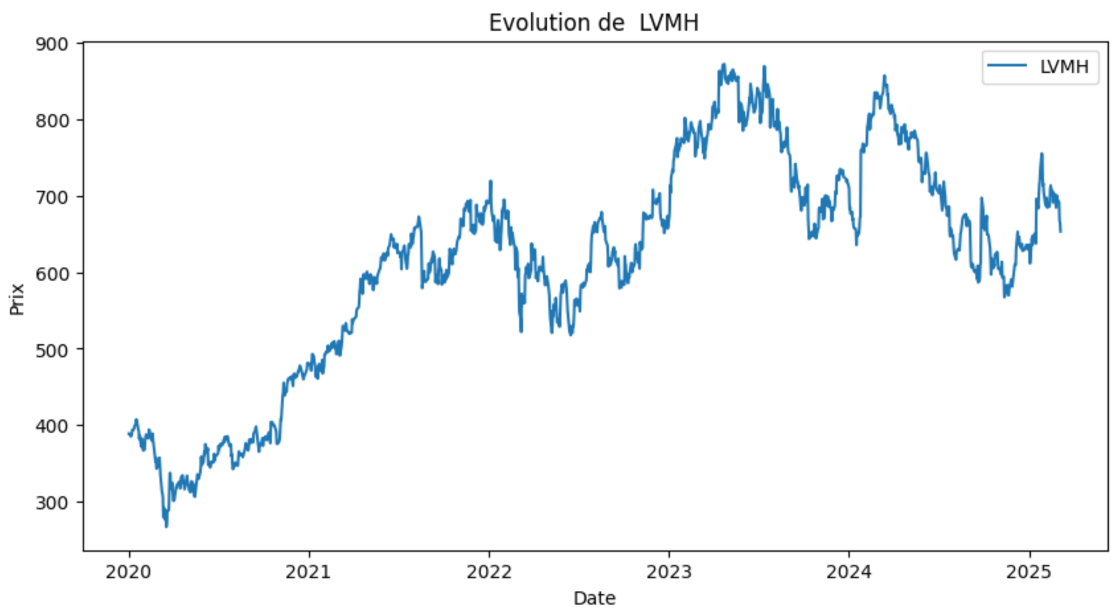

Evolution de Hermès
Hermès
Prix
Date
2500
2000
1500
1000
500
2020
2021
2022
2023
2024
2025


Evolution de Kering
Kering
700
600
500
400
300
200
Prix
2020
2021
2022
2023
2024
2025
Date

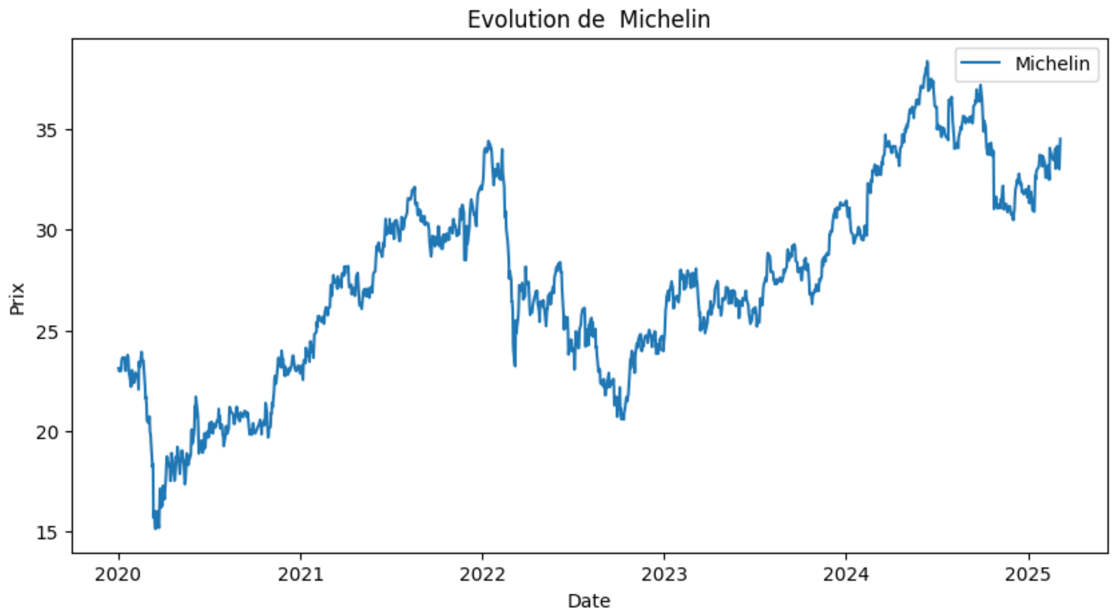

Evolution de L'Oréal
L'Oréal
450
400
350
300
250
200
Prix
2020
2021
2022
2023
2024
2025
Date

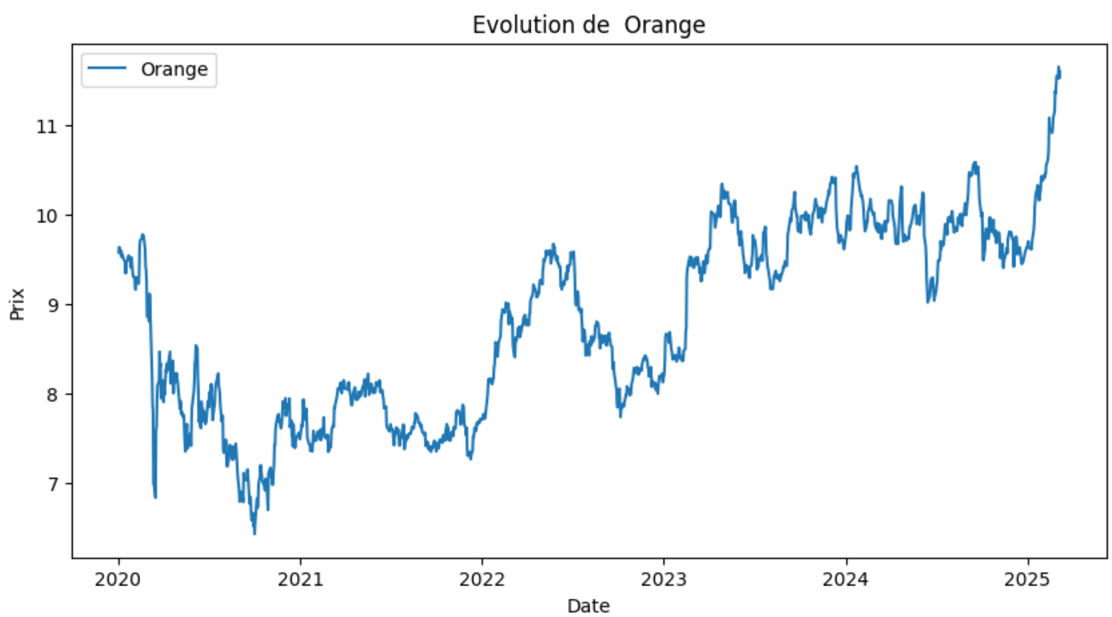


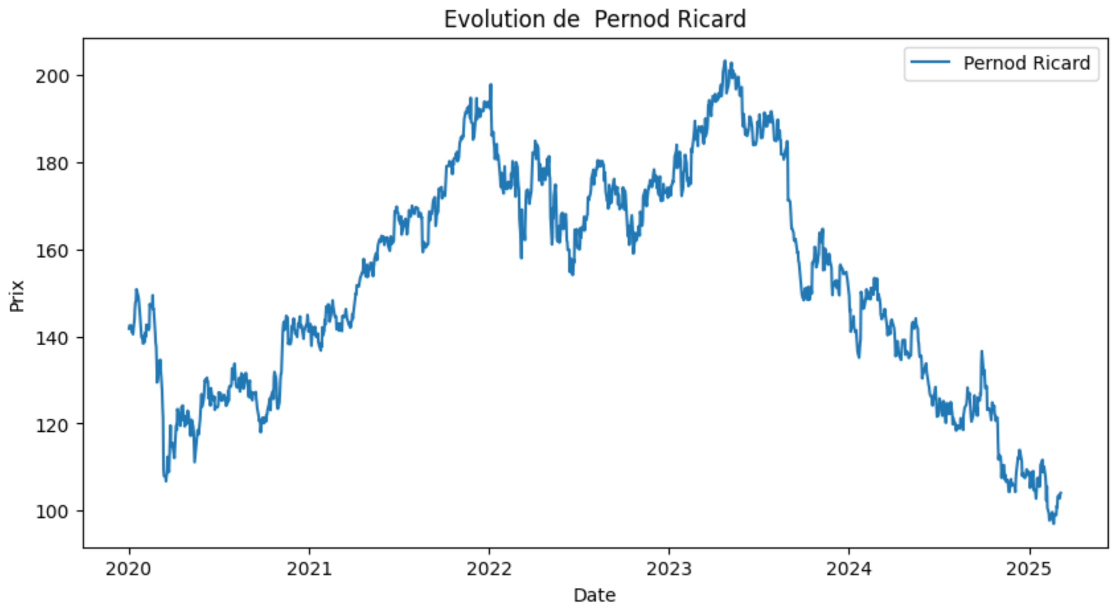
Evolution de Legrand
Legrand
110
100
90
80
70
60
50
Prix
2020
2021
2022
2023
2024
2025
Date

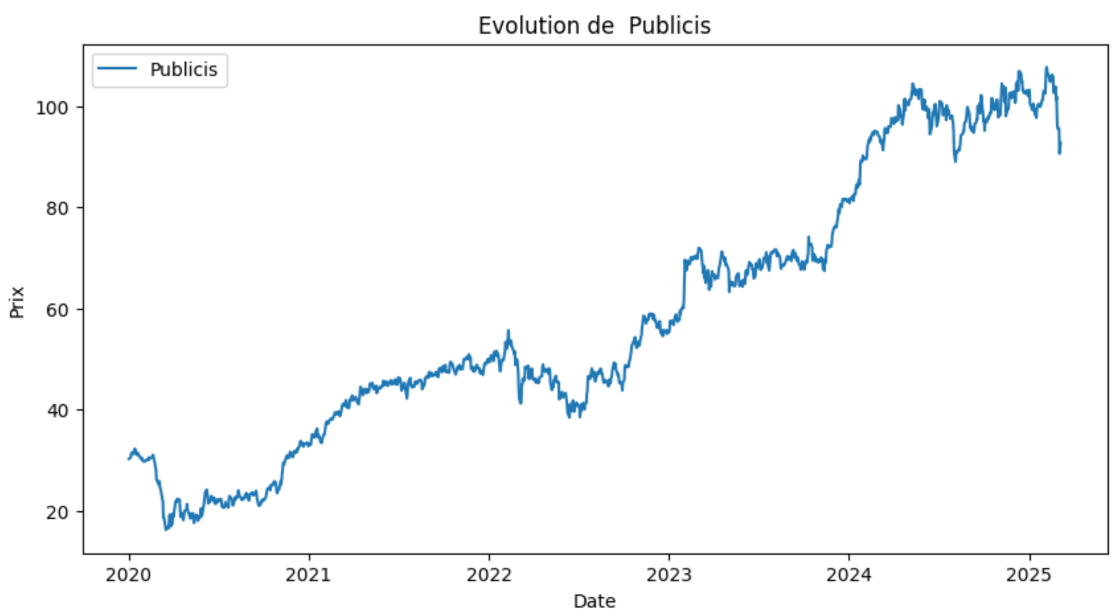


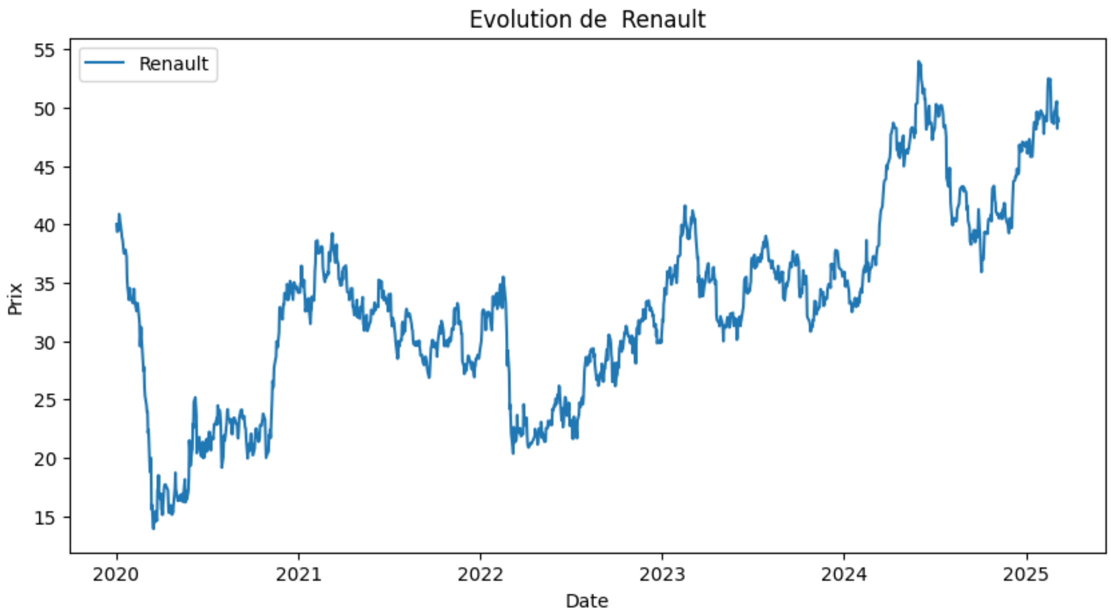
Evolution de LVMH
LVMH
900
800
700
600
500
400
300
Prix
2020
2021
2022
2023
2024
2025
Date

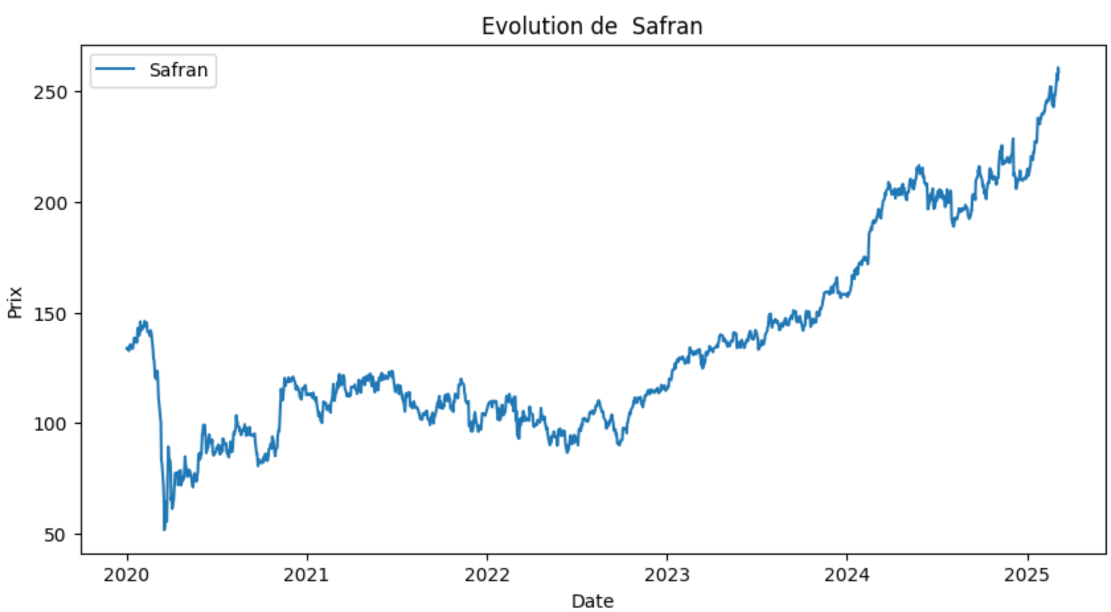

Evolution de Michelin
Michelin
35
30
25
20
15
Prix
2020
2021
2022
2023
2024
2025
Date

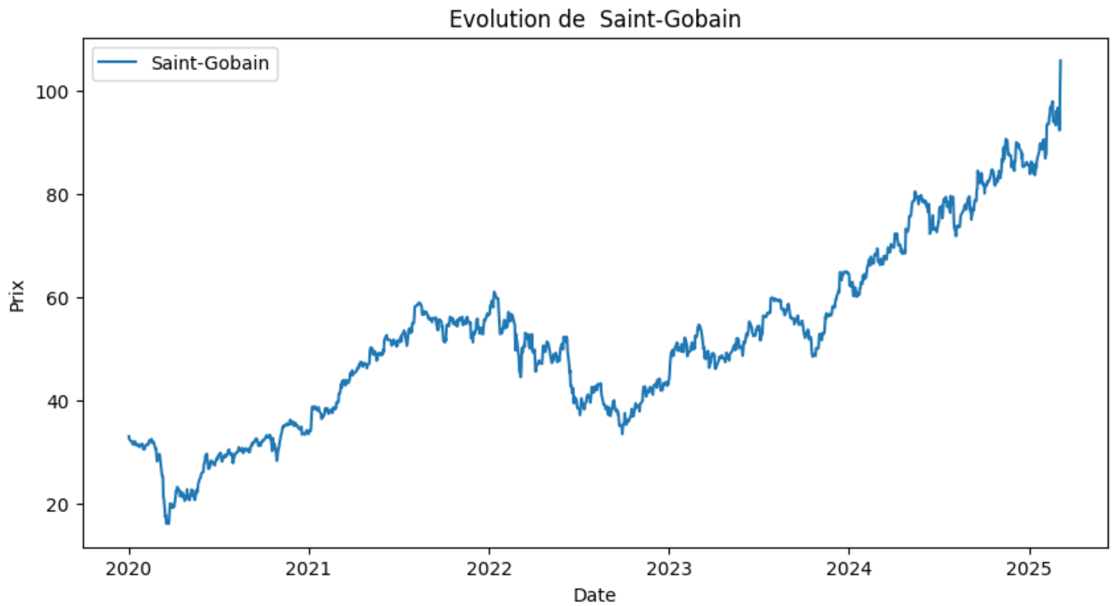

Evolution de Orange
Orange
11
10
9
8
7
Prix
2020
2021
2022
2023
2024
2025
Date

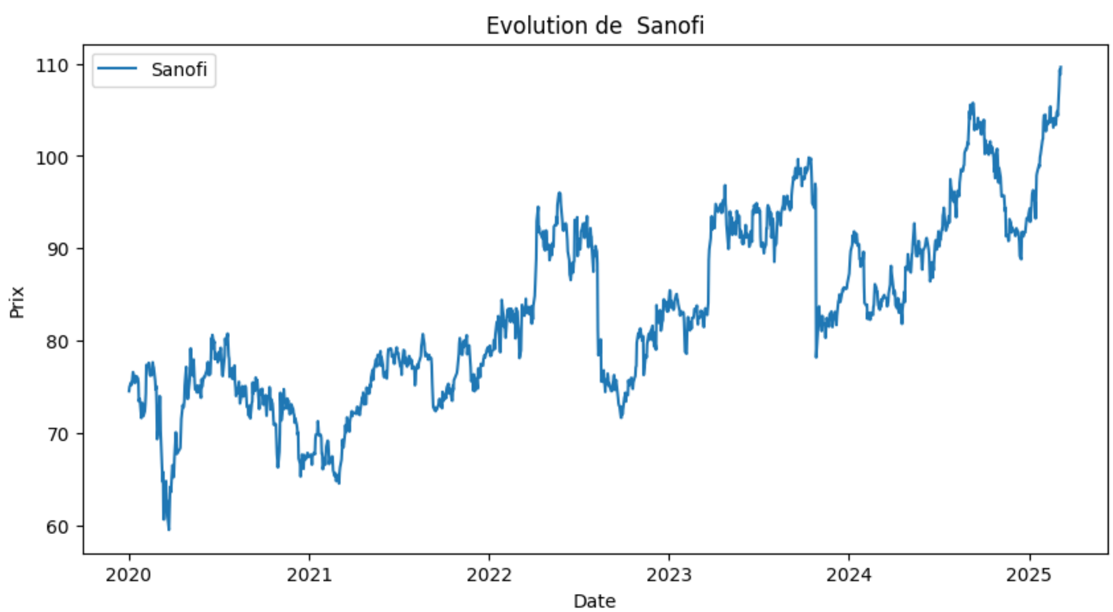

Evolution de Pernod Ricard
Pernod Ricard
Prix
200
180
160
140
120
100
2020
2021
2022
2023
2024
2025
Date

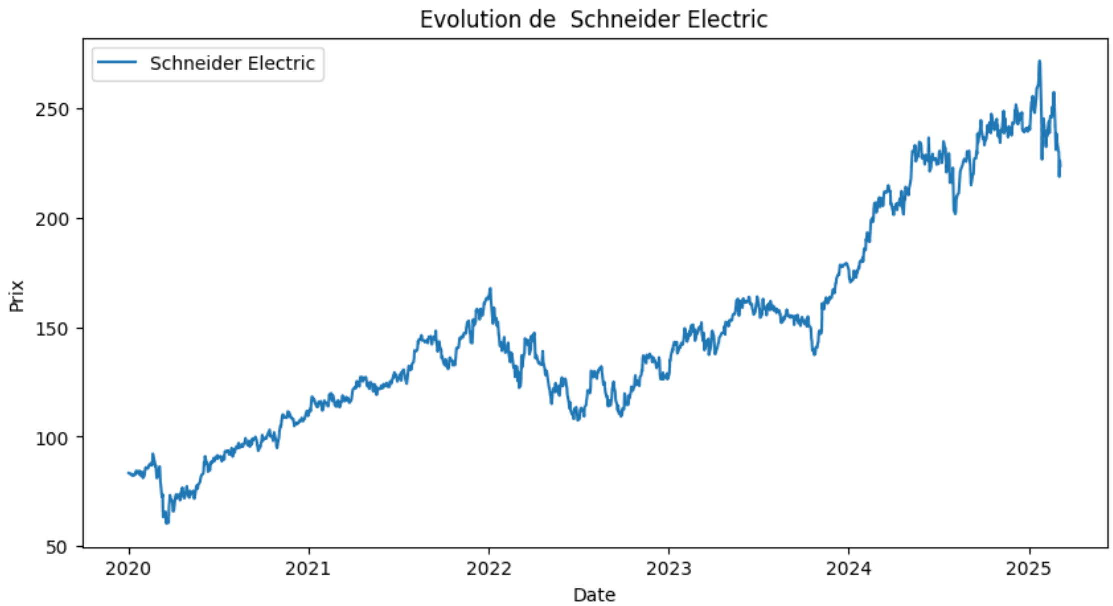

Evolution de Publicis
Publicis
Prix
100
80
60
40
20
2020
2021
2022
2023
2024
2025
Date

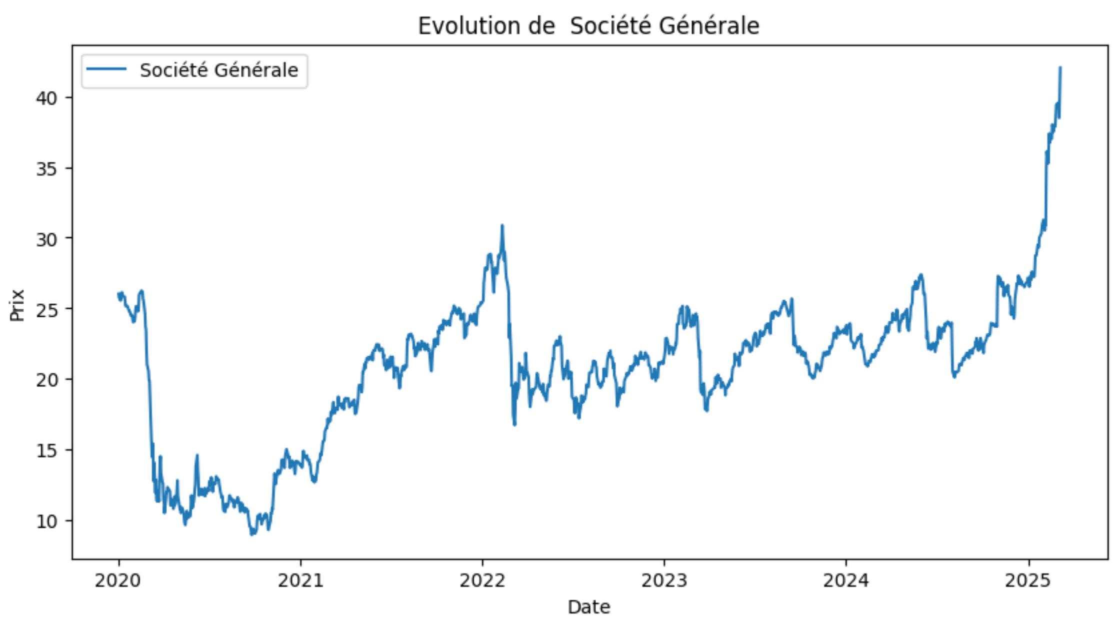

Evolution de Renault
Renault
Prix
Date
55
50
45
40
35
30
25
20
15
2020
2021
2022
2023
2024
2025

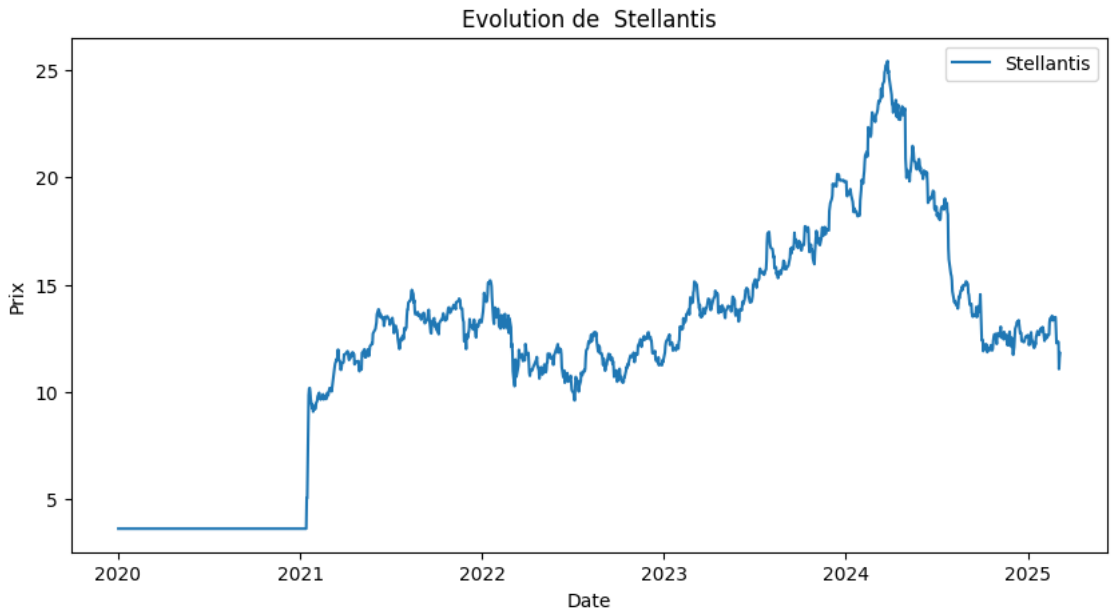

Evolution de Safran
Safran
Prix
Date
250
200
150
100
50
2020
2021
2022
2023
2024
2025

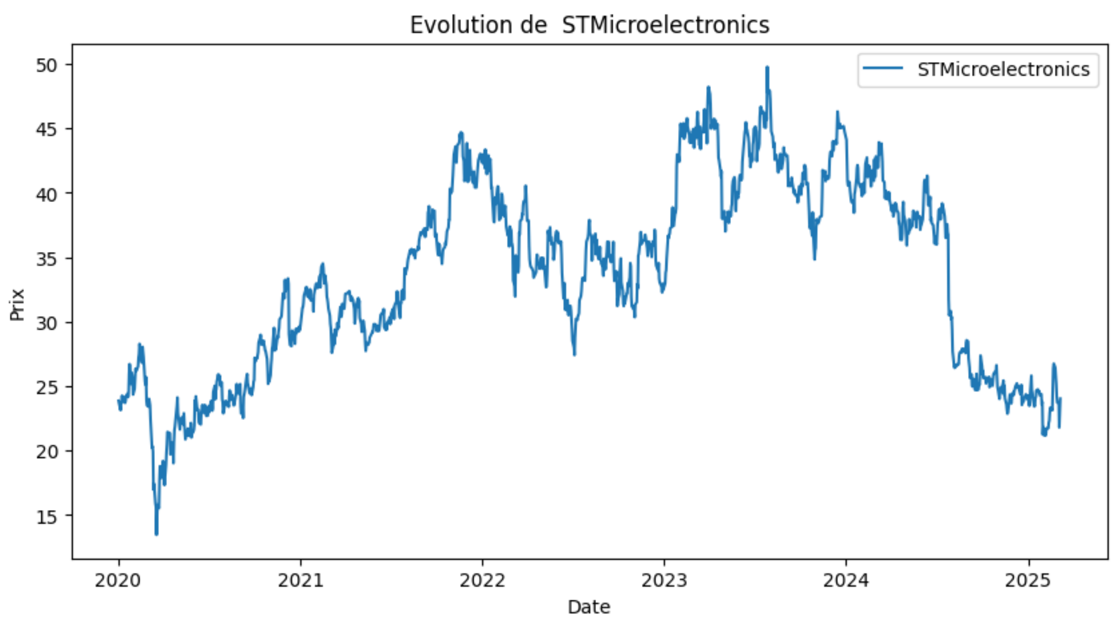

Evolution de Saint-Gobain
Saint-Gobain
Prix
Date
100
80
60
40
20
2020
2021
2022
2023
2024
2025

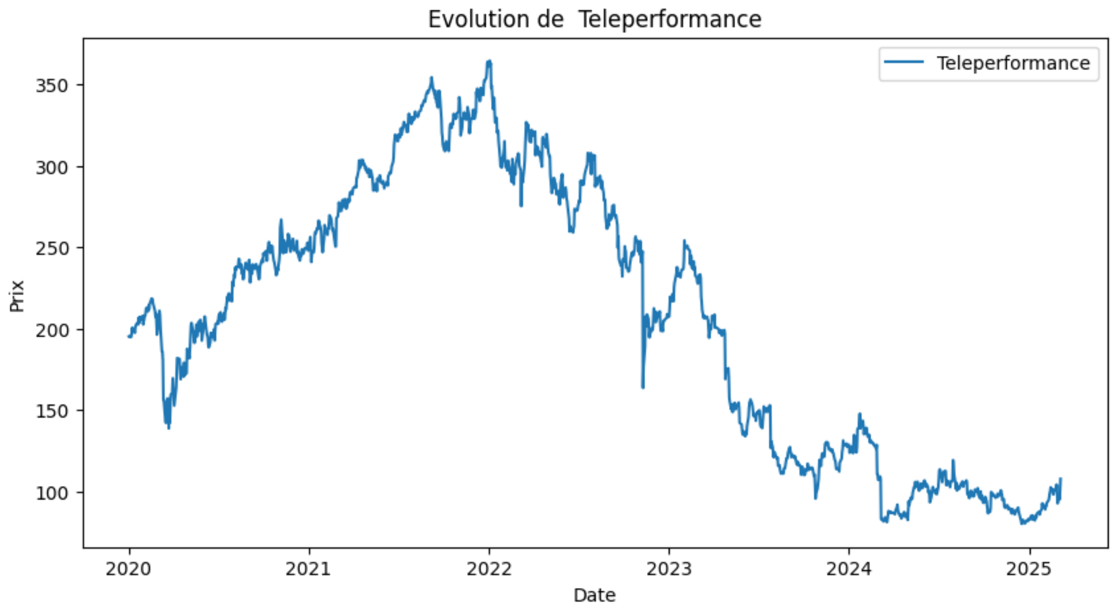

Evolution de Sanofi
Sanofi
Prix
Date
110
100
90
80
70
60
2020
2021
2022
2023
2024
2025

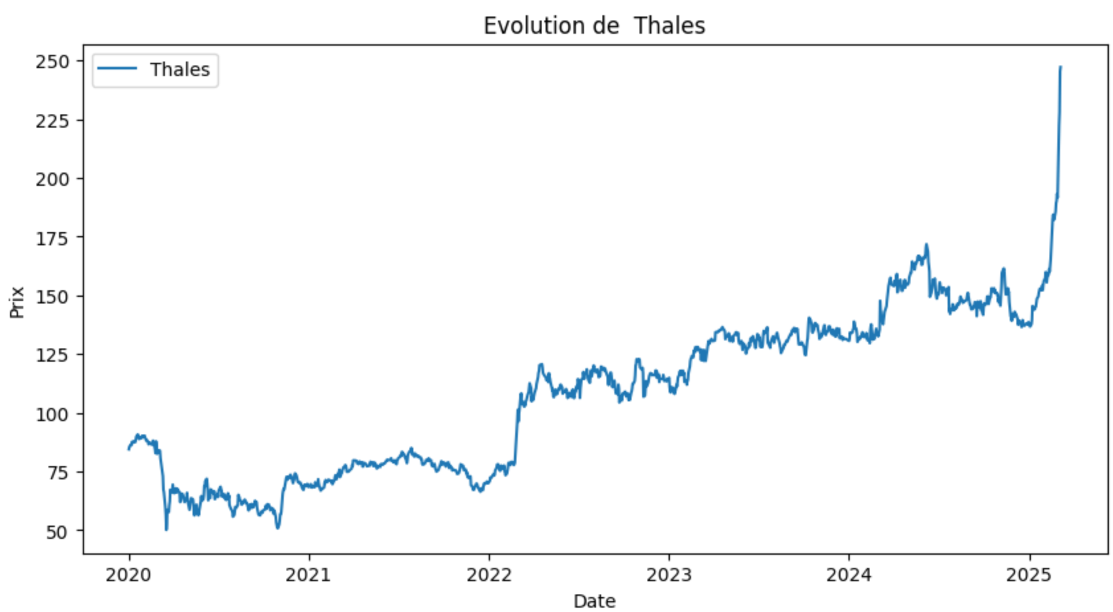

Evolution de Schneider Electric
Schneider Electric
Prix
Date
250
200
150
100
50
2020
2021
2022
2023
2024
2025

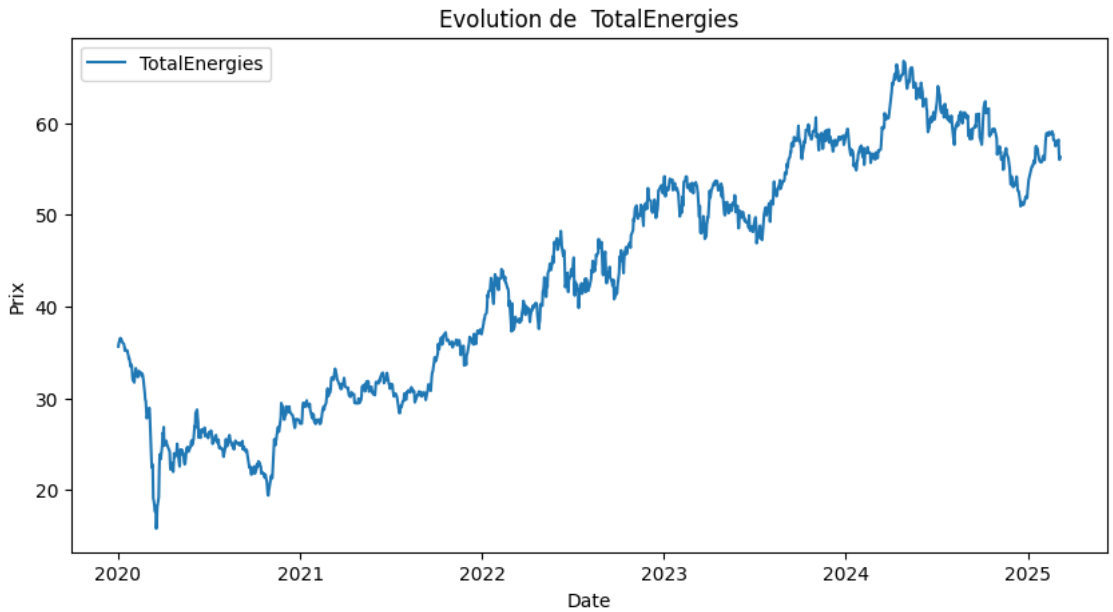

Evolution de Société Générale
Société Générale
Prix
Date
40
35
30
25
20
15
10
2020
2021
2022
2023
2024
2025

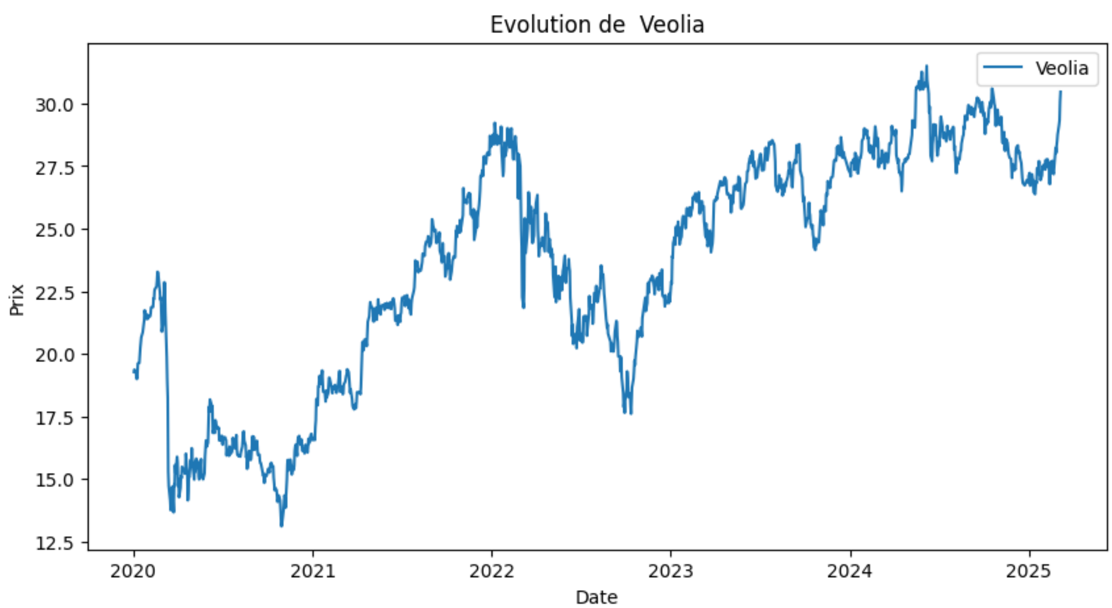

Evolution de Stellantis
Stellantis
Prix
Date
25
20
15
10
5
2020
2021
2022
2023
2024
2025

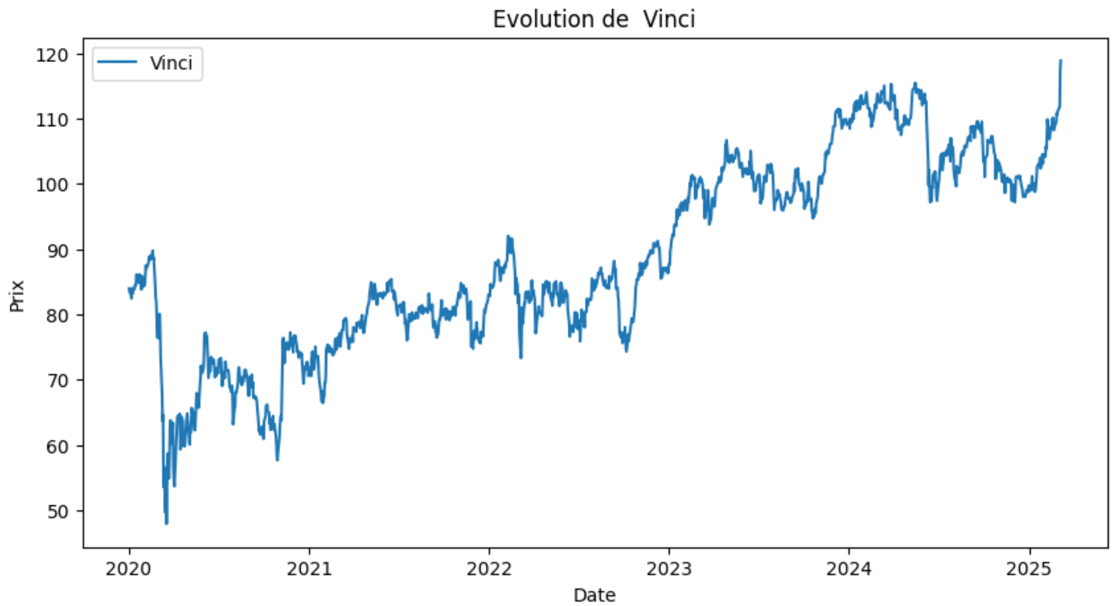

Evolution de STMicroelectronics
STMicroelectronics
Prix
Date
50
45
40
35
30
25
20
15
2020
2021
2022
2023
2024
2025

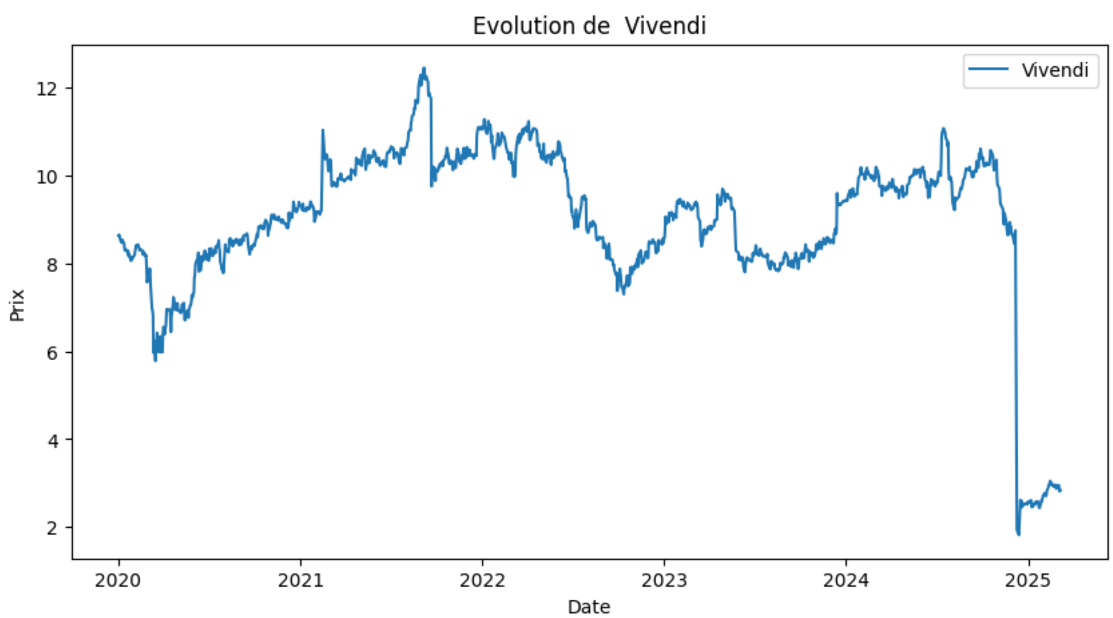

Evolution de Teleperformance
Teleperformance
350
300
250
200
150
100
Prix
2020
2021
2022
2023
2024
2025
Date


Evolution de Thales
Thales
250
225
200
175
150
125
100
75
50
Prix
2020
2021
2022
2023
2024
2025
Date


Evolution de TotalEnergies
TotalEnergies
60
50
40
30
20
Prix
2020
2021
2022
2023
2024
2025
Date


Evolution de Veolia
Veolia
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
Prix
2020
2021
2022
2023
2024
2025
Date


Evolution de Vinci
Vinci
Prix
Date
120
110
100
90
80
70
60
50
2020
2021
2022
2023
2024
2025


Evolution de Vivendi
Vivendi
Prix
Date
12
10
8
6
4
2
2020
2021
2022
2023
2024
2025

[98]:
```python
# Les rendements des actifs
returns = cac40_data.pct_change()
returns.dropna(inplace=True)
print(f"Description du rendements des actifs \n {returns.describe().T}")
```

```
Description du rendements des actifs
           count      mean       std       min       25%       50%       75%  \
Ticker
AC.PA      484.0  0.001022  0.012798 -0.047001 -0.006098  0.001435  0.008572
ACA.PA     484.0  0.001253  0.011374 -0.056192 -0.004189  0.001859  0.007254
AI.PA      484.0  0.000597  0.010741 -0.038347 -0.005315  0.001182  0.006693
AIR.PA     484.0  0.000786  0.013593 -0.094099 -0.007079  0.001809  0.007722
BN.PA      484.0  0.000553  0.008580 -0.027574 -0.004607  0.000866  0.005501
BNP.PA     484.0  0.000916  0.014186 -0.092086 -0.006244  0.001706  0.009155
CA.PA      484.0 -0.000465  0.013463 -0.088289 -0.007193  0.000297  0.007914
CAP.PA     484.0  0.000065  0.016365 -0.102184 -0.008084  0.000000  0.009180
CS.PA      484.0  0.000921  0.010761 -0.049111 -0.004761  0.001577  0.007619
DG.PA      484.0  0.000415  0.011151 -0.053725 -0.005171  0.000737  0.006973
DSY.PA     484.0  0.000288  0.015764 -0.103555 -0.007223  0.000000  0.008196
EDEN.PA    484.0 -0.000758  0.019721 -0.146469 -0.007476  0.001163  0.008718
EL.PA      484.0  0.001101  0.012335 -0.044539 -0.005686  0.000734  0.008024
EN.PA      484.0  0.000561  0.011788 -0.047493 -0.005771  0.000971  0.007331
ENGI.PA    484.0  0.000655  0.009679 -0.033369 -0.004890  0.001024  0.005852
ERF.PA     484.0 -0.000133  0.018880 -0.161547 -0.008470  0.000685  0.010309
GLE.PA     484.0  0.001686  0.018017 -0.120514 -0.005980  0.001619  0.010429
HO.PA      484.0  0.001387  0.016612 -0.067039 -0.006637  0.001591  0.009190
KER.PA     484.0 -0.001297  0.020661 -0.119145 -0.011328 -0.001262  0.008623
LR.PA      484.0  0.000702  0.013934 -0.072771 -0.006617  0.000445  0.007920
MC.PA      484.0 -0.000419  0.017767 -0.064622 -0.009903  0.000005  0.008202
ML.PA      484.0  0.000568  0.011921 -0.082224 -0.005605 -0.000296  0.007656
MT.AS      484.0  0.000622  0.018764 -0.068966 -0.008656  0.000422  0.008497
OR.PA      484.0 -0.000110  0.013349 -0.075803 -0.007908  0.000000  0.008062
ORA.PA     484.0  0.000377  0.008954 -0.040991 -0.004701  0.000815  0.005858
PUB.PA     484.0  0.000720  0.013087 -0.057214 -0.006306  0.000969  0.008574
RI.PA      484.0 -0.001196  0.014422 -0.067422 -0.009146 -0.001684  0.006670
RMS.PA     484.0  0.000709  0.015856 -0.065414 -0.006718  0.000313  0.008544
RNO.PA     484.0  0.000787  0.018663 -0.079738 -0.008487  0.001226  0.011459
SAF.PA     484.0  0.001431  0.012166 -0.073053 -0.005258  0.001577  0.008923
SAN.PA     484.0  0.000413  0.014144 -0.189329 -0.005875  0.000645  0.006972
SGO.PA     484.0  0.001737  0.015503 -0.043144 -0.006701  0.001528  0.009219
STLAP.PA   484.0 -0.000258  0.019636 -0.147394 -0.007929  0.000576  0.010453
STMPA.PA   484.0 -0.001052  0.022699 -0.137033 -0.012987  0.000303  0.010677
SU.PA      484.0  0.001026  0.016282 -0.094772 -0.006931  0.001875  0.010055
TEP.PA     484.0 -0.000799  0.029488 -0.231292 -0.015064  0.000000  0.013883
TTE.PA     484.0  0.000176  0.012344 -0.050742 -0.006244  0.001149  0.008051
URW.PA     484.0  0.001000  0.016607 -0.055467 -0.008151  0.001267  0.010592
VIE.PA     484.0  0.000337  0.011415 -0.046448 -0.006479  0.000375  0.007328
VIV.PA     484.0 -0.000620  0.043042 -0.778406 -0.006982  0.000000  0.007095
```

Ticker	max
AC.PA	0.065595
ACA.PA	0.061371
AI.PA	0.082596
AIR.PA	0.052384
BN.PA	0.050232
BNP.PA	0.042613
CA.PA	0.049347
CAP.PA	0.068480
CS.PA	0.030602
DG.PA	0.050559
DSY.PA	0.088597
EDEN.PA	0.048930
EL.PA	0.073892
EN.PA	0.080047
ENGI.PA	0.052663
ERF.PA	0.060662
GLE.PA	0.131779
HO.PA	0.160449
KER.PA	0.096091
LR.PA	0.090164
MC.PA	0.128119
ML.PA	0.068809
MT.AS	0.133415
OR.PA	0.069624
ORA.PA	0.030698
PUB.PA	0.046632
RI.PA	0.078539
RMS.PA	0.091043
RNO.PA	0.065269
SAF.PA	0.041155
SAN.PA	0.044698
SGO.PA	0.088439
STLAP.PA	0.057846
STMPA.PA	0.089143
SU.PA	0.082571
TEP.PA	0.138512
TTE.PA	0.039231
URW.PA	0.082573
VIE.PA	0.029670
VIV.PA	0.435483

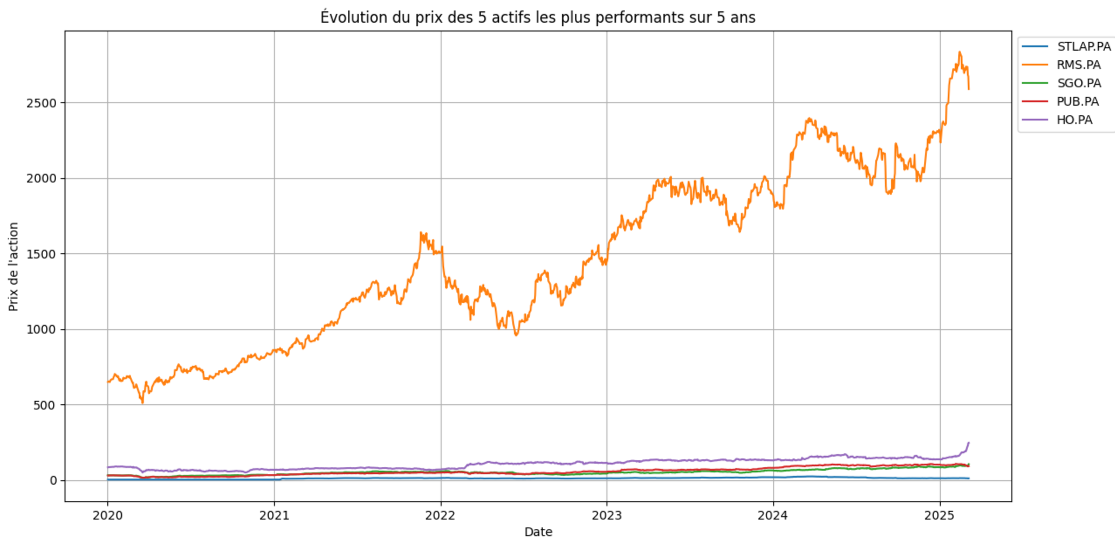
Analyse de la performance des actions

- Performance du point de vue du rendement

[99]:
```python
# Les 5 actifs les plus performants
mean_returns = returns.mean()
top_5_performing = mean_returns.nlargest(5)
print(f"Les 5 actifs les plus performants : \n")
print(top_5_performing)
```

```
Les 5 actifs les plus performants :

Ticker
SGO.PA    0.001737
GLE.PA    0.001686
SAF.PA    0.001431
HO.PA     0.001387
ACA.PA    0.001253
dtype: float64
```

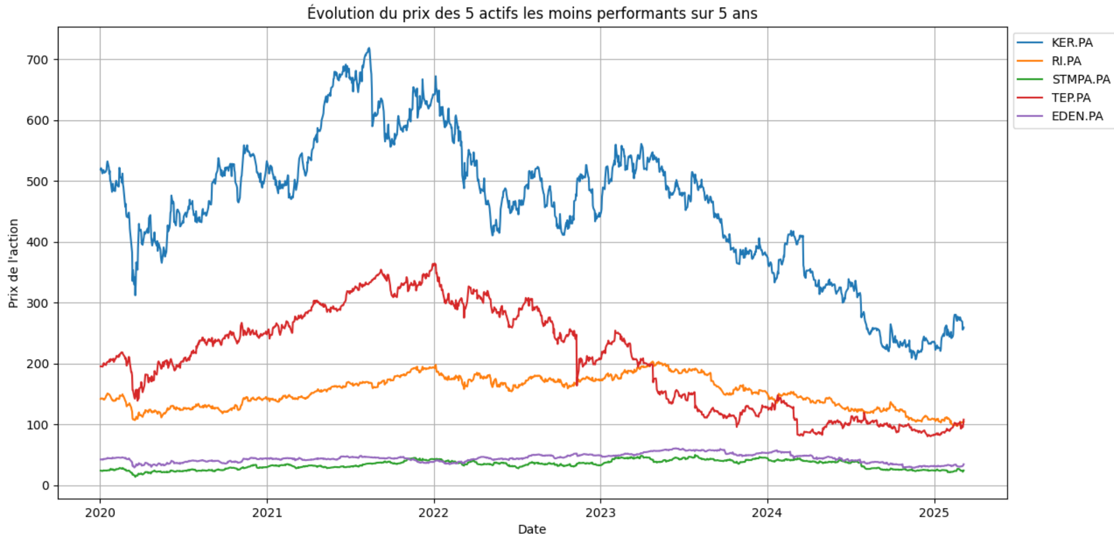
[100]:
```python
# Visualisation des 5 actifs les plus performants
ticker0 = ['STLAP.PA', 'RMS.PA', 'SGO.PA', 'PUB.PA', 'HO.PA']
top_5_performing_data = cac40_data[ticker0]
plt.figure(figsize=(14, 7))

for i in ticker0:
    plt.plot(cac40_data.index, cac40_data[i],label=i)

plt.title("Évolution du prix des 5 actifs les plus performants sur 5 ans")
plt.xlabel("Date")
plt.ylabel("Prix de l'action")
plt.legend(loc="upper left", bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



[101]:
```
# Les 5 actifs les moins performants
bottom_5_performing = mean_returns.nsmallest(5)
print(f"Les 5 actifs les moins performants : \n")
print(bottom_5_performing)
```

```
Les 5 actifs les moins performants :

Ticker
KER.PA      -0.001297
RI.PA       -0.001196
STMPA.PA    -0.001052
TEP.PA      -0.000799
EDEN.PA     -0.000758
dtype: float64
```

[102]:
```
# Les 5 actifs les moins performants
ticker1 = bottom_5_performing.index
plt.figure(figsize=(14, 7))

for j in ticker1:
    plt.plot(cac40_data.index, cac40_data[j],label=j)

plt.title("Évolution du prix des 5 actifs les moins performants sur 5 ans")
plt.xlabel("Date")
plt.ylabel("Prix de l'action")
plt.legend(loc="upper left", bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



- Performence du point de vue du risque

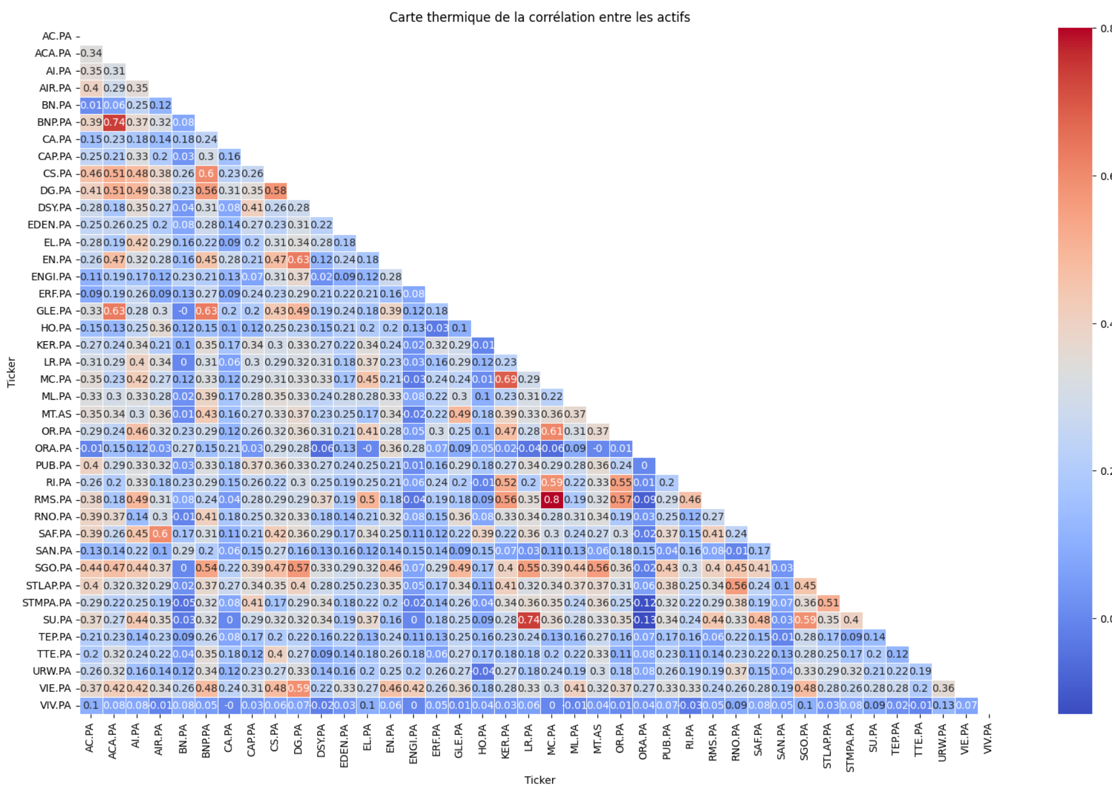
[103]:
```python
# Volatilité des actifs
volatility = returns.std()

# Les 5 actifs les plus volatils
top_5_least_volatile = volatility.nsmallest(5)
print(f"Les 5 actifs les moins volatils : \n {top_5_least_volatile}")

# Les 5 actifs les plus volatils
top_5_most_volatile = volatility.nlargest(5)
print(f"\n Les 5 actifs les plus volatils : \n {top_5_most_volatile}")
```

```
Les 5 actifs les moins volatils :
 Ticker
BN.PA      0.008580
ORA.PA     0.008954
ENGI.PA    0.009679
AI.PA      0.010741
CS.PA      0.010761
dtype: float64

 Les 5 actifs les plus volatils :
 Ticker
VIV.PA     0.043042
TEP.PA     0.029488
STMPA.PA   0.022699
KER.PA     0.020661
EDEN.PA    0.019721
dtype: float64
```

3. Analyse de corrélation des actifs

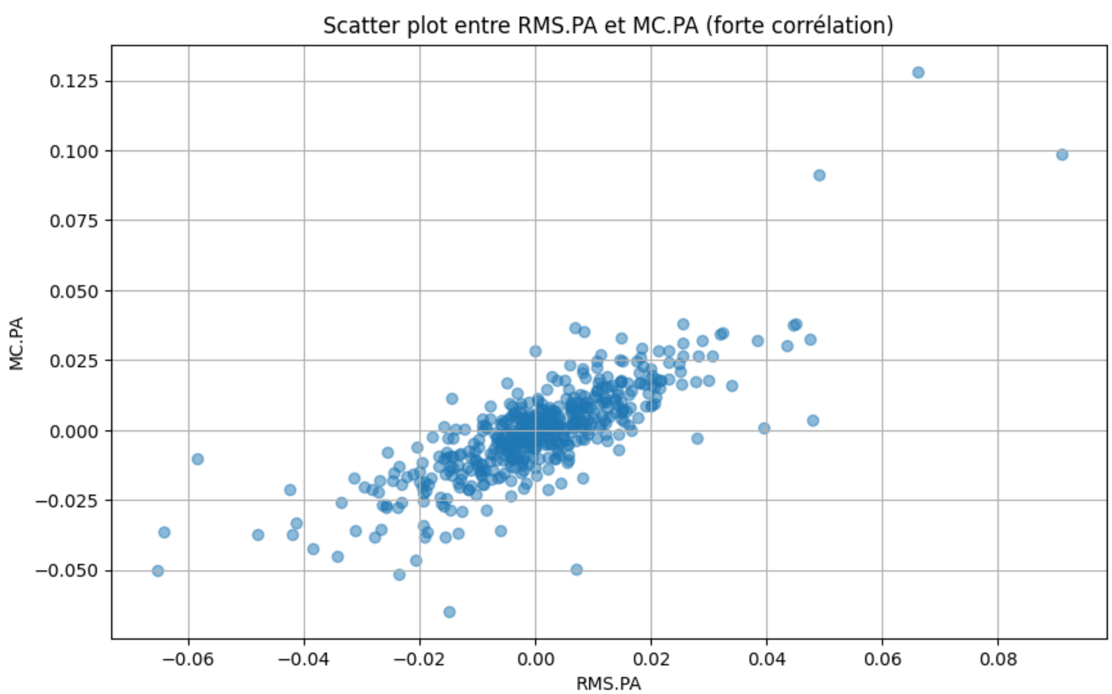
[105]:
```python
# Corrélation entre les actifs
r2 = np.round(returns.corr(),2)
# Carte thermique
mask =np.triu(np.ones(r2.shape, dtype=bool))
plt.figure(figsize=(20,12))
sns.heatmap(r2,cmap="coolwarm", linewidths=0.5, annot=True, mask=mask)
plt.title("Carte thermique de la corrélation entre les actifs")
```

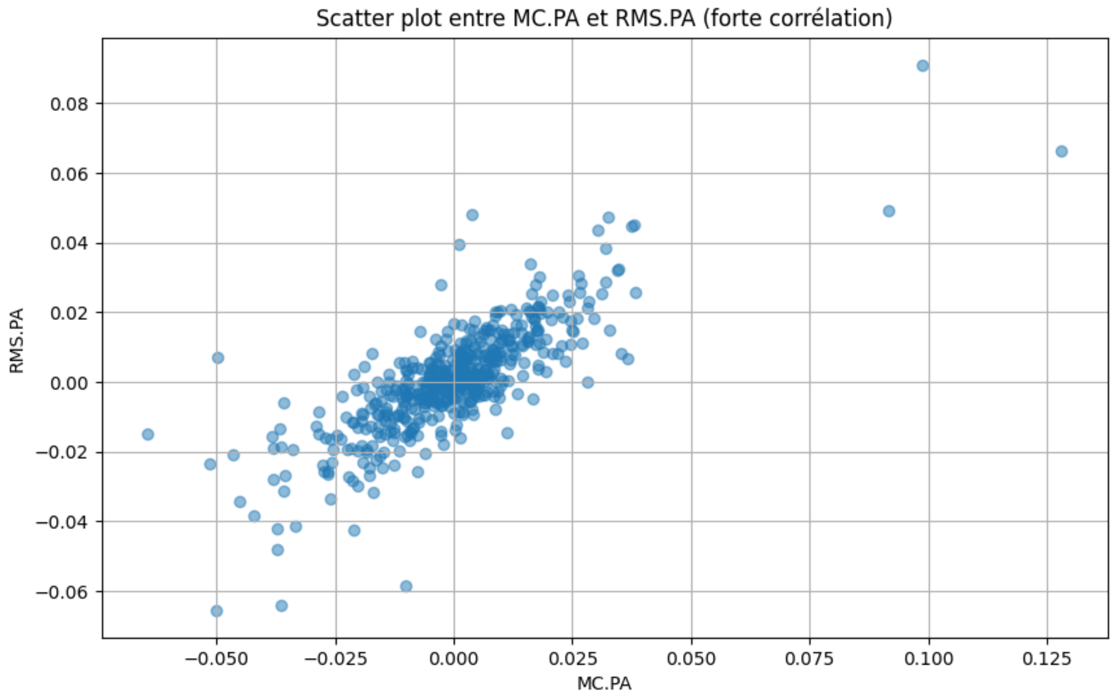
[105]:
```
Text(0.5, 1.0, 'Carte thermique de la corrélation entre les actifs')
```

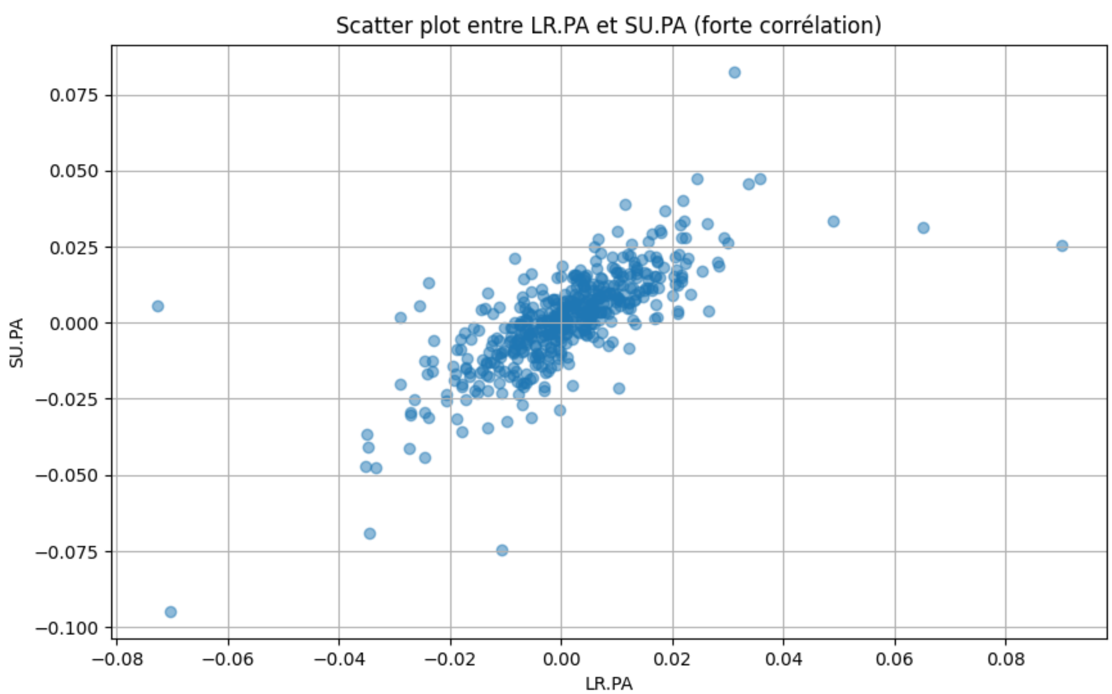


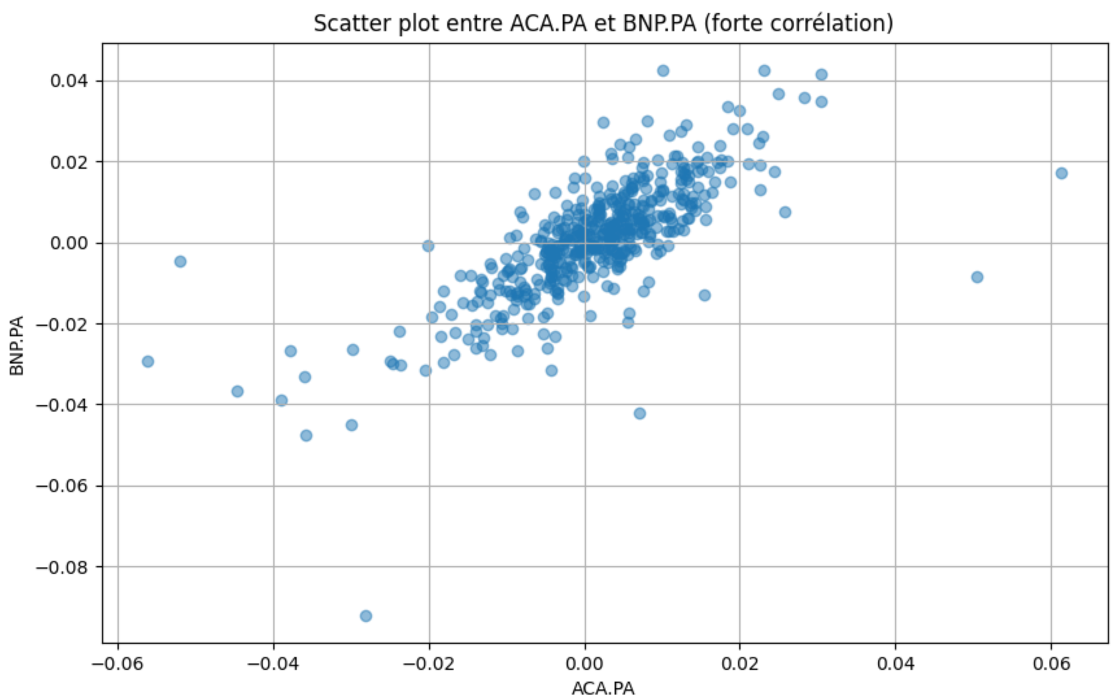
[106]:

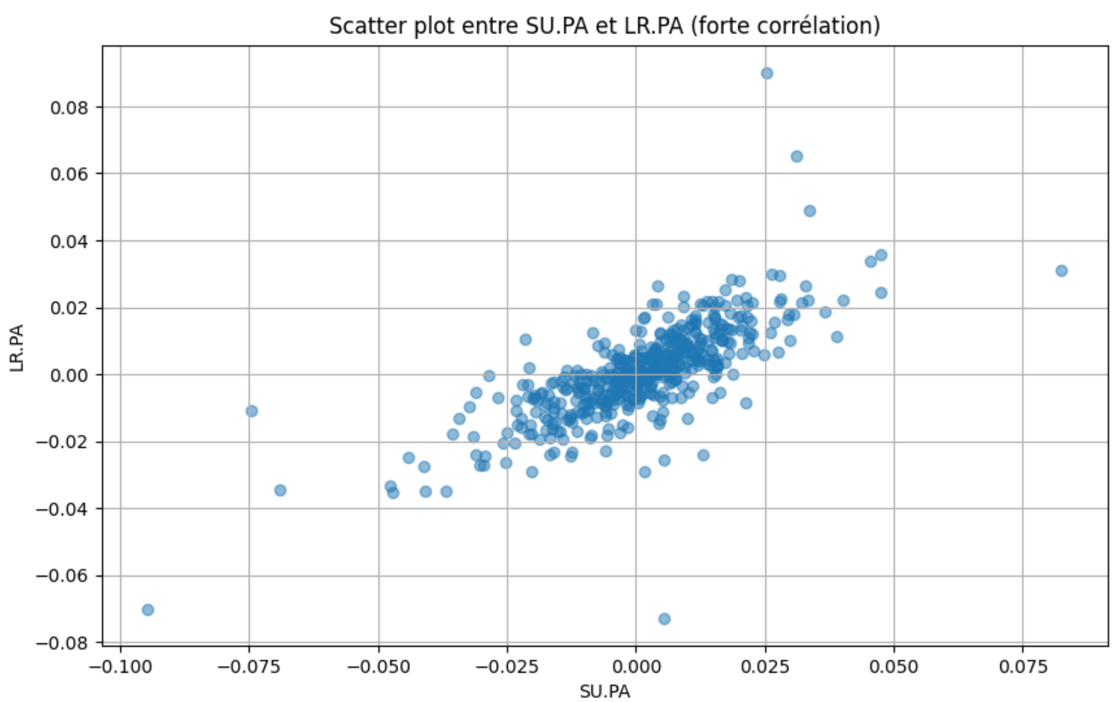
```
# Trouver les paires d'actifs les plus corrélés
correlation_pairs = r2.unstack().sort_values(kind="quicksort", ascending=False)
# Supprimer les paires avec une corrélation de 1 (corrélation d'un actif avec␣
↪lui-même)
correlation_pairs = correlation_pairs[correlation_pairs < 1]

# Sélectionner les 5 paires les plus corrélées
top_5_correlated_pairs = correlation_pairs.head(10).index

# Tracer les scatter plots pour les paires les plus corrélées
for pair in top_5_correlated_pairs:
    asset1, asset2 = pair
    plt.figure(figsize=(10, 6))
    plt.scatter(returns[asset1], returns[asset2], alpha=0.5)
    plt.title(f"Scatter plot entre {asset1} et {asset2} (forte corrélation)")
    plt.xlabel(asset1)
    plt.ylabel(asset2)
    plt.grid(True)
    plt.show()
```

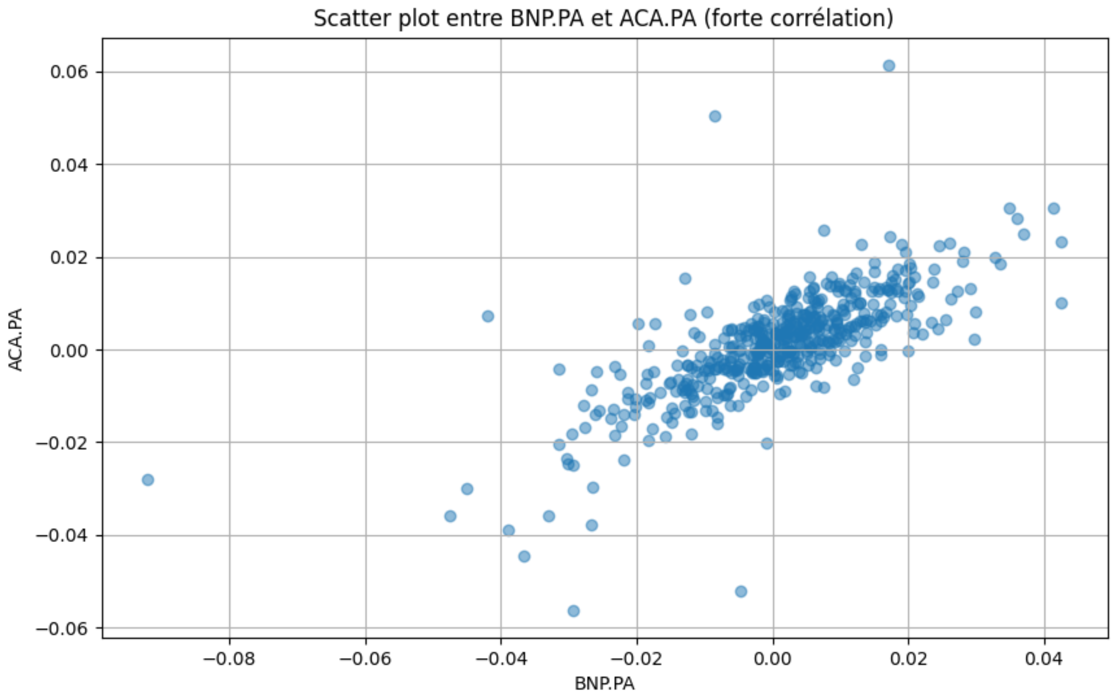


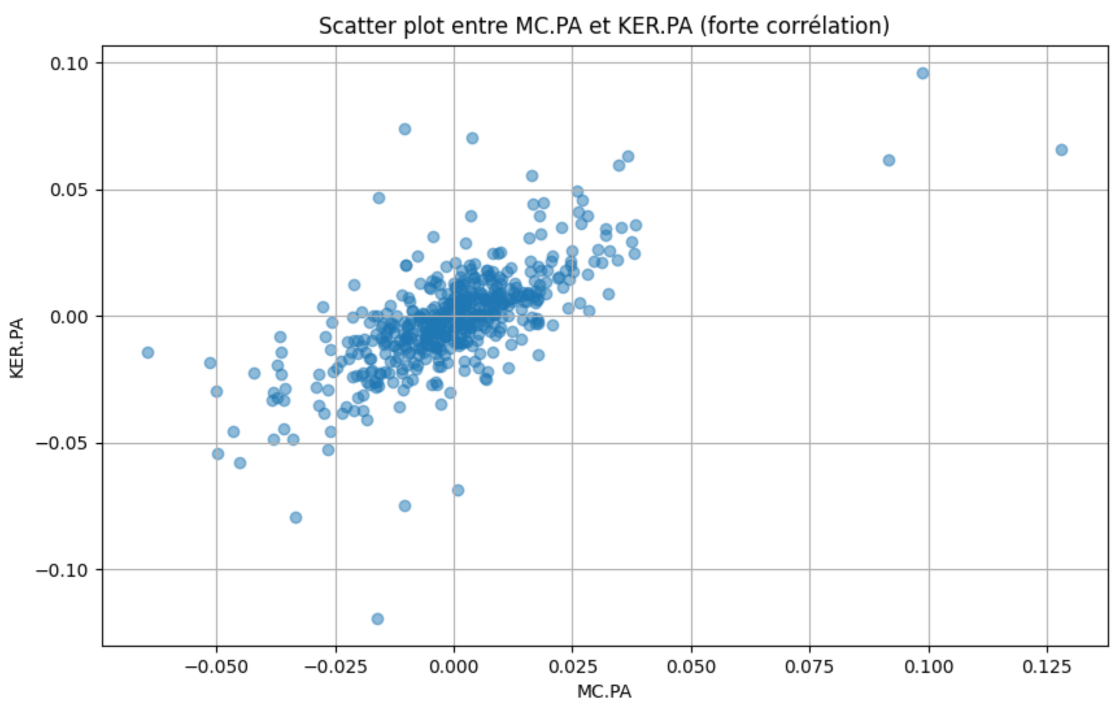

Scatter plot entre MC.PA et RMS.PA (forte corrélation)
RMS.PA
MC.PA
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100
0.125

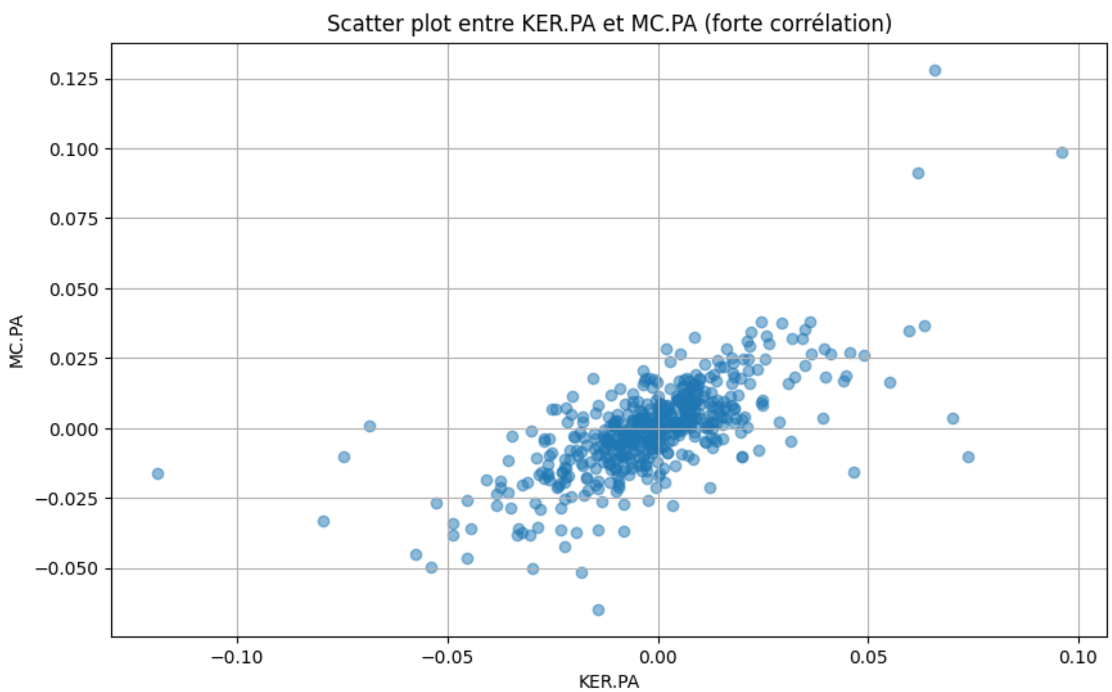

Scatter plot entre LR.PA et SU.PA (forte corrélation)
SU.PA
LR.PA
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08

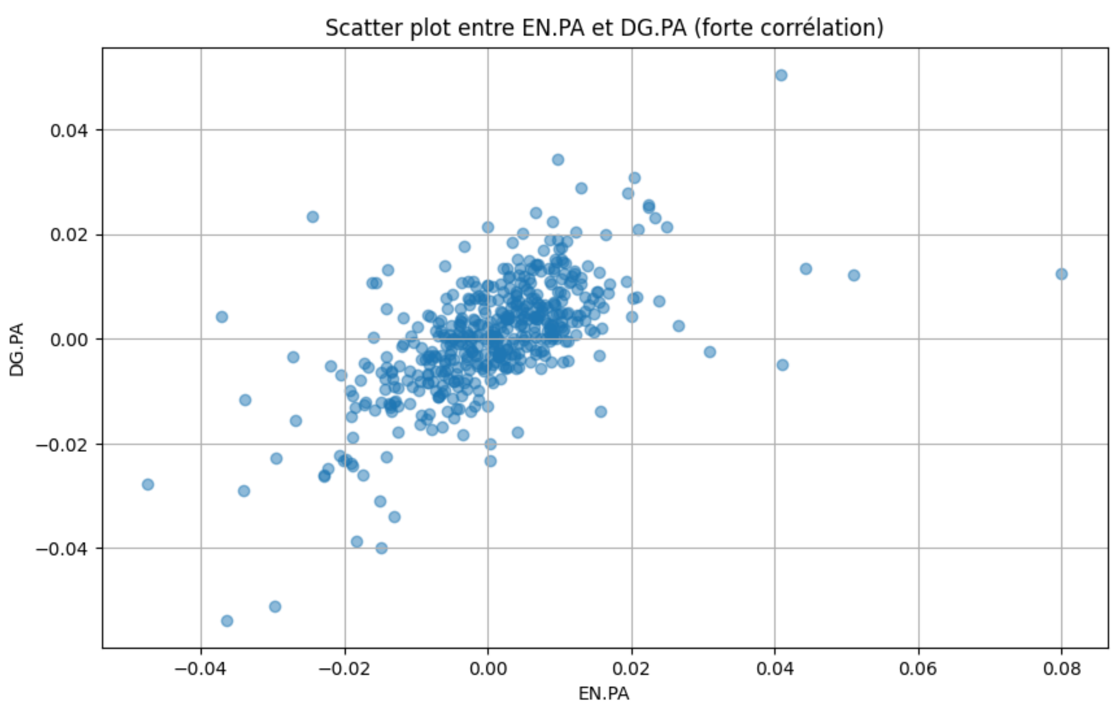

Scatter plot entre ACA.PA et BNP.PA (forte corrélation)
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
BNP.PA
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
ACA.PA

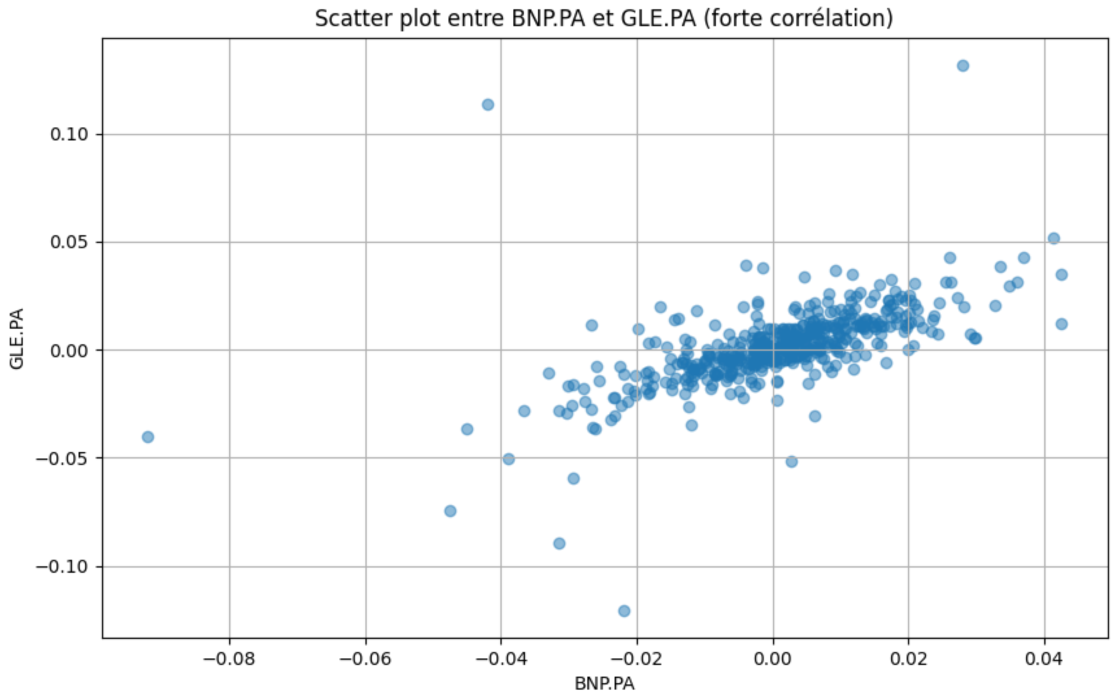

Scatter plot entre SU.PA et LR.PA (forte corrélation)
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
LR.PA
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
SU.PA


Scatter plot entre BNP.PA et ACA.PA (forte corrélation)
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
ACA.PA
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
BNP.PA

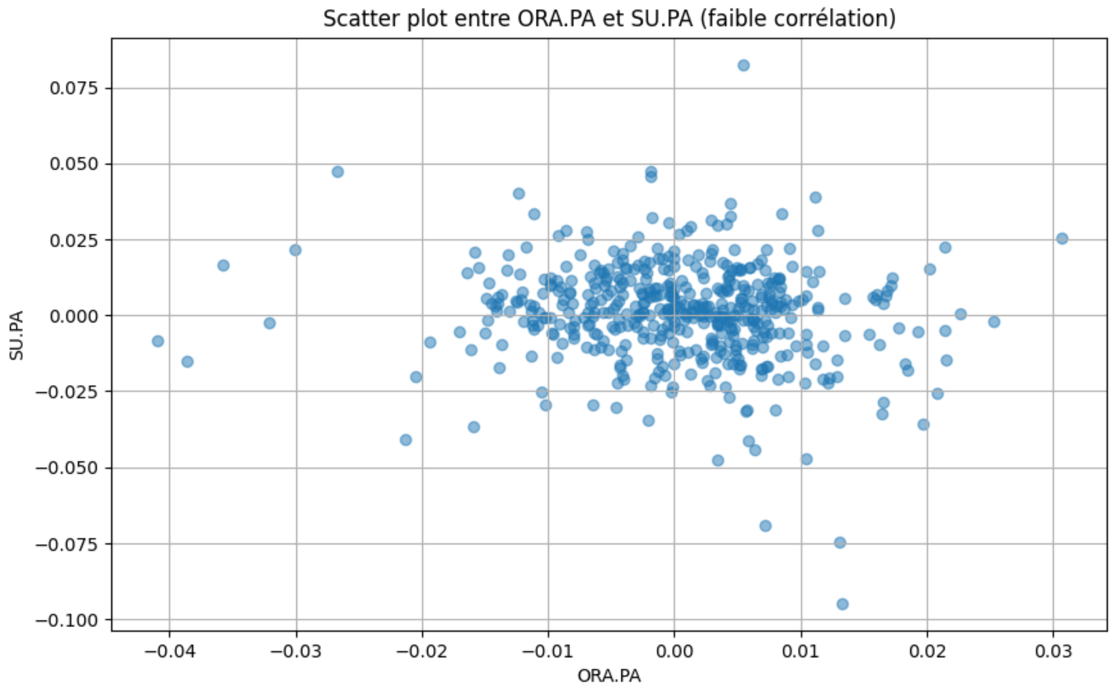

Scatter plot entre MC.PA et KER.PA (forte corrélation)
0.10
0.05
0.00
−0.05
−0.10
KER.PA
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100
0.125
MC.PA

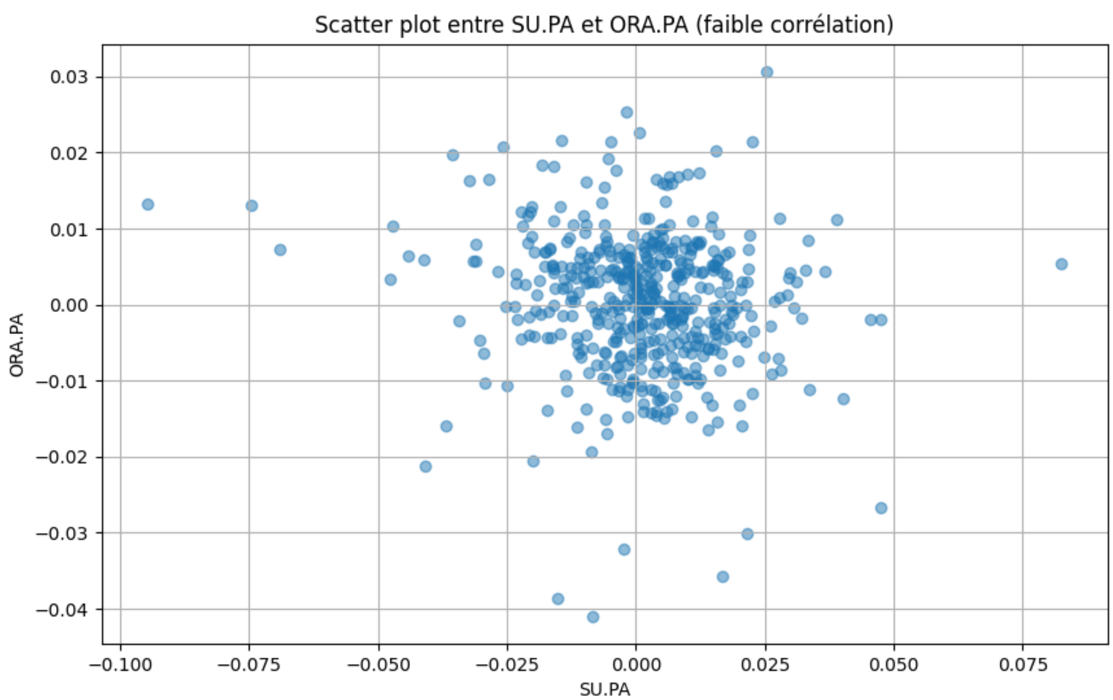

Scatter plot entre KER.PA et MC.PA (forte corrélation)
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
MC.PA
−0.10
−0.05
0.00
0.05
0.10
KER.PA

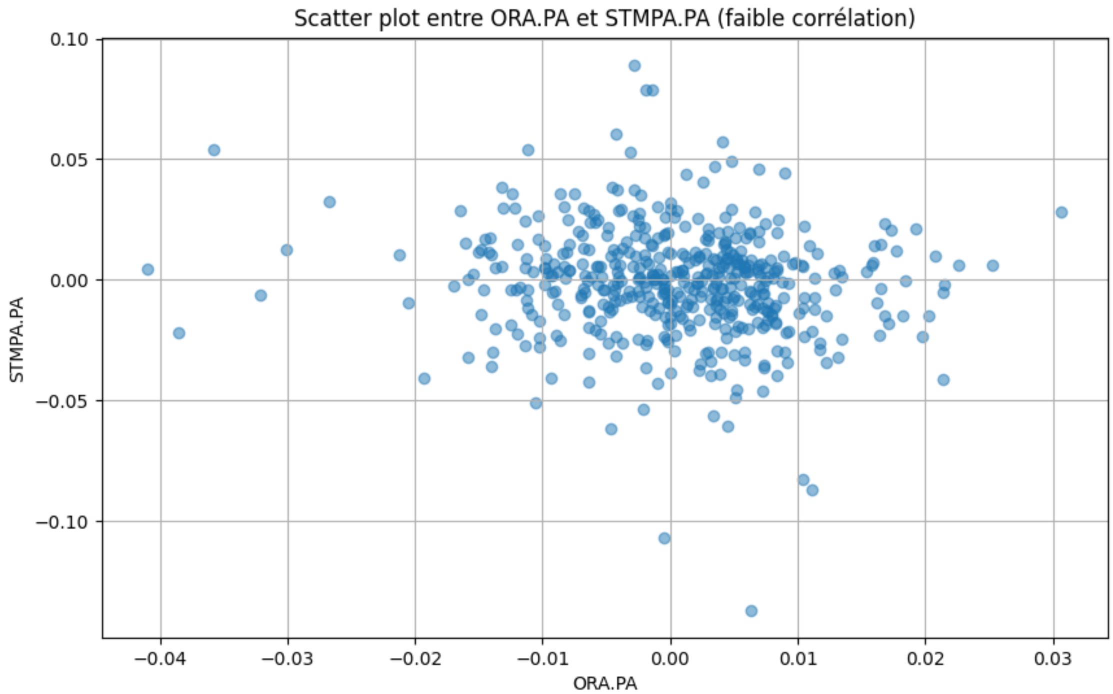

Scatter plot entre EN.PA et DG.PA (forte corrélation)
0.04
0.02
0.00
−0.02
−0.04
DG.PA
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
EN.PA

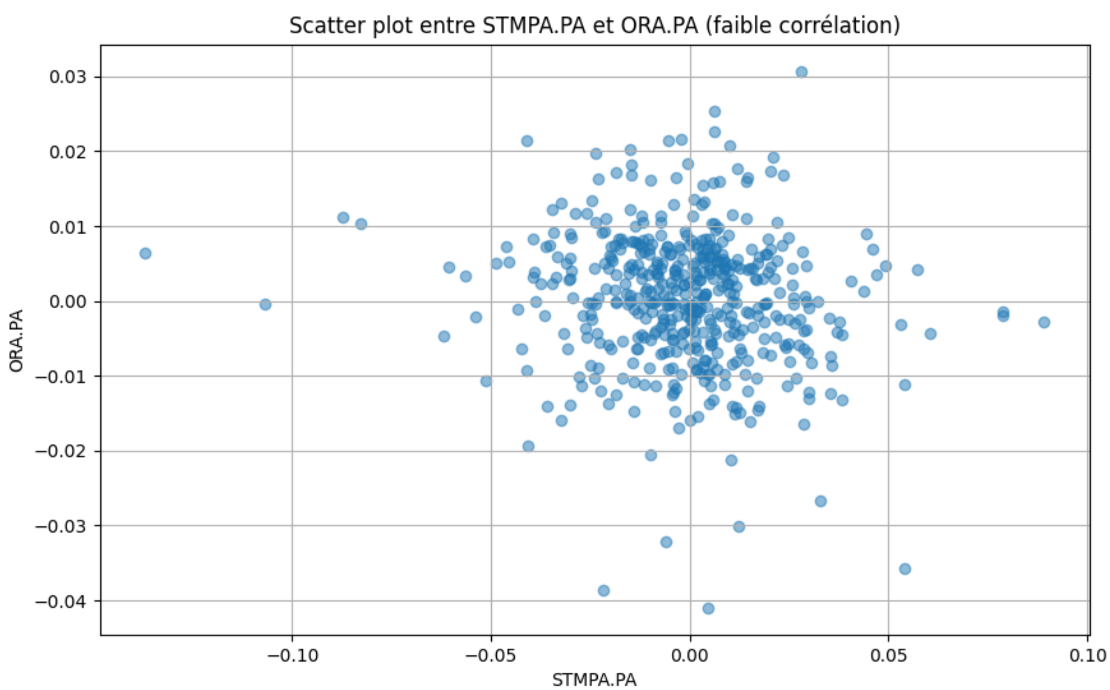


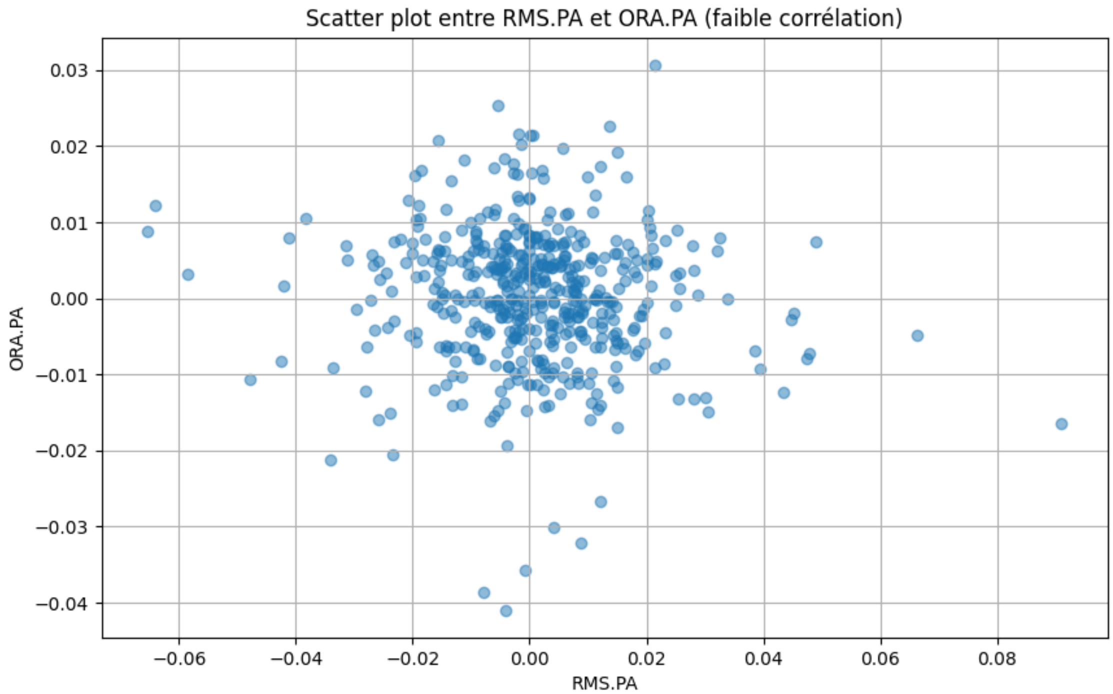
[107]:
```python
# Trouver les paires d'actifs les moins corrélés
correlation_pairs = r2.unstack().sort_values(kind="quicksort", ascending=True)
# Supprimer les paires avec une corrélation de 1 (corrélation d'un actif avec␣
↪lui-même)
correlation_pairs = correlation_pairs[correlation_pairs < 1]

# Sélectionner les 5 paires les moins corrélées
bottom_5_correlated_pairs = correlation_pairs.head(10).index

# Tracer les scatter plots pour les paires les moins corrélées
for pair in bottom_5_correlated_pairs:
    asset1, asset2 = pair
    plt.figure(figsize=(10, 6))
    plt.scatter(returns[asset1], returns[asset2], alpha=0.5)
    plt.title(f"Scatter plot entre {asset1} et {asset2} (faible corrélation)")
    plt.xlabel(asset1)
    plt.ylabel(asset2)
    plt.grid(True)
    plt.show()
```

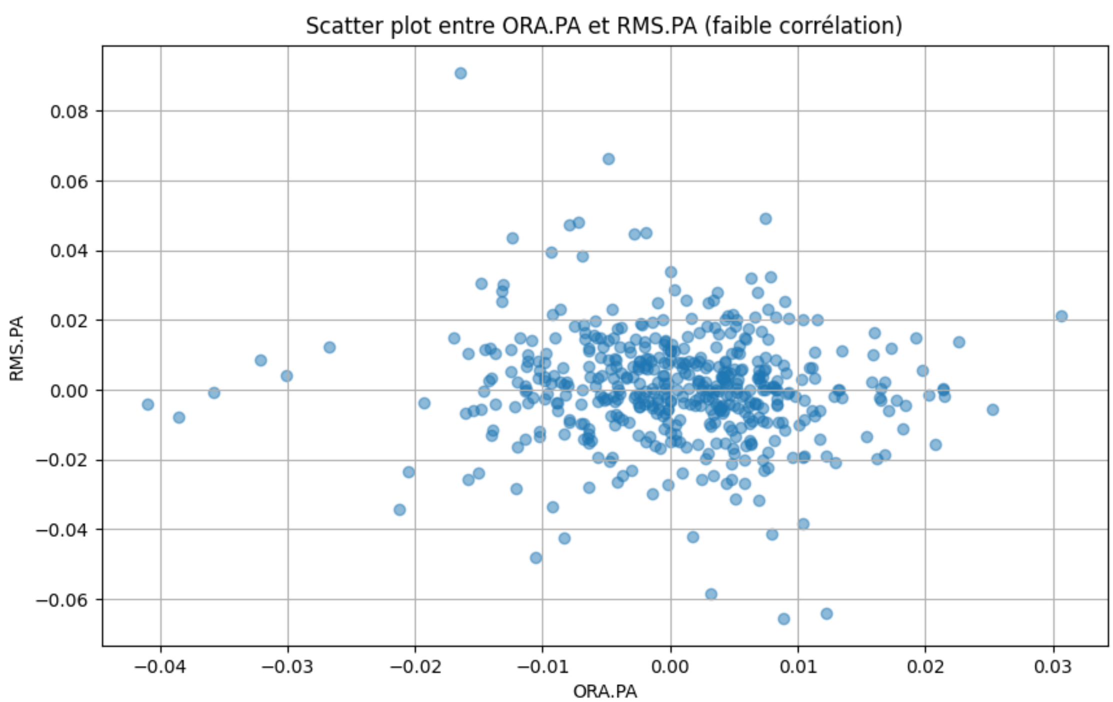

Scatter plot entre ORA.PA et SU.PA (faible corrélation)
SU.PA
ORA.PA
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03

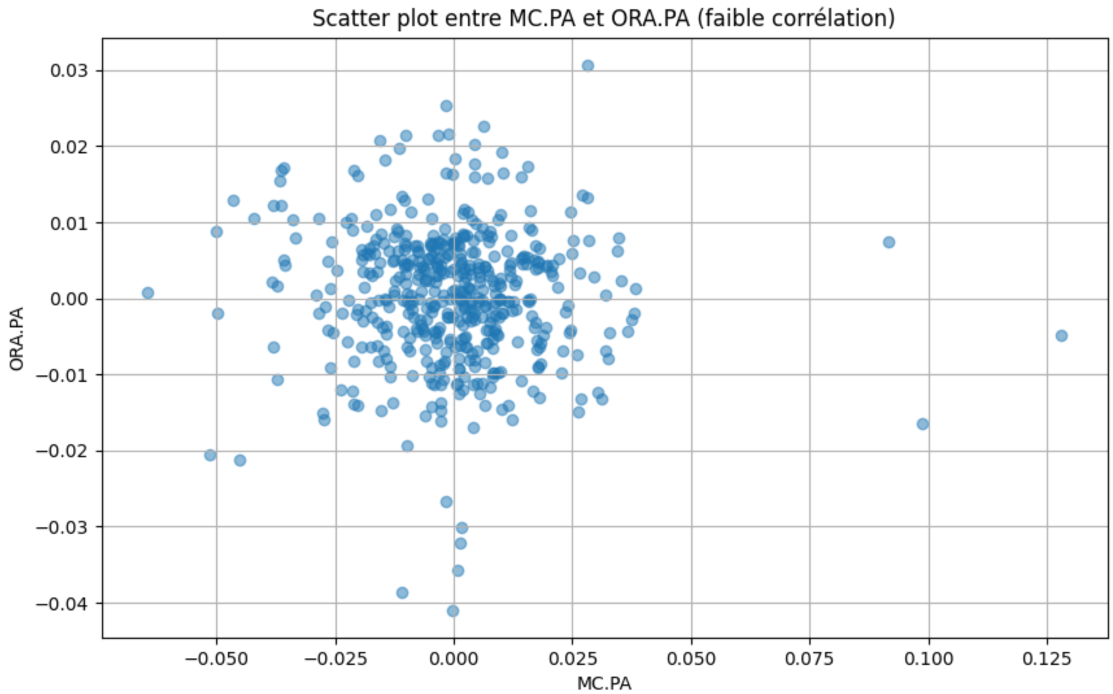

Scatter plot entre SU.PA et ORA.PA (faible corrélation)
ORA.PA
SU.PA
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075

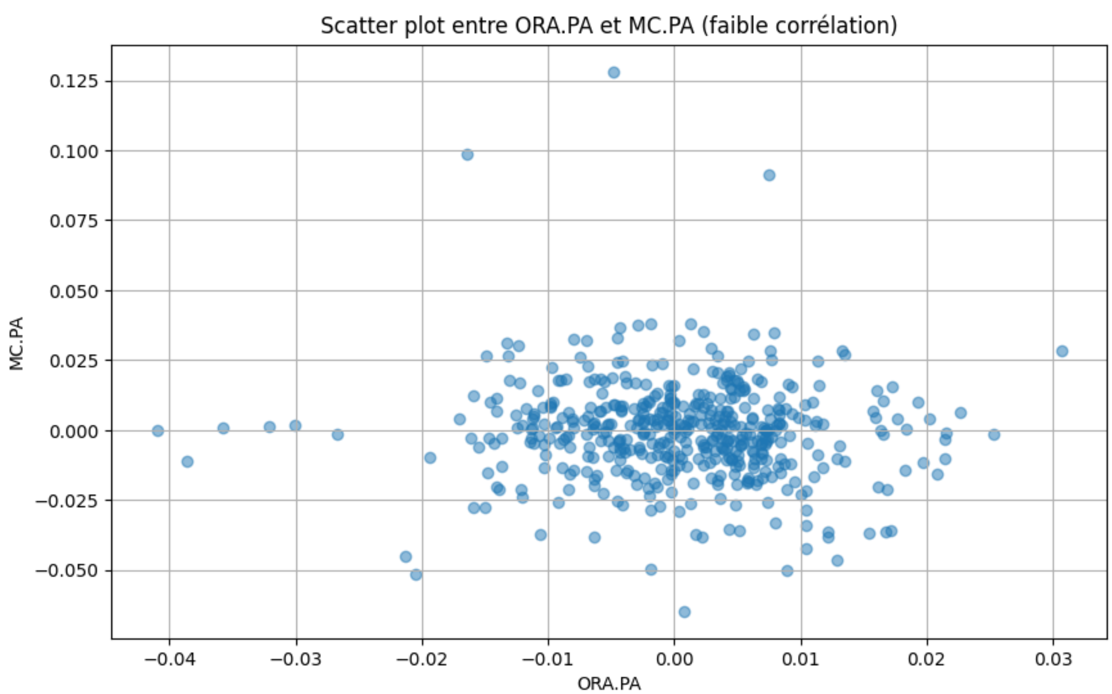

Scatter plot entre ORA.PA et STMPA.PA (faible corrélation)
0.10
0.05
0.00
−0.05
−0.10
STMPA.PA
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03
ORA.PA

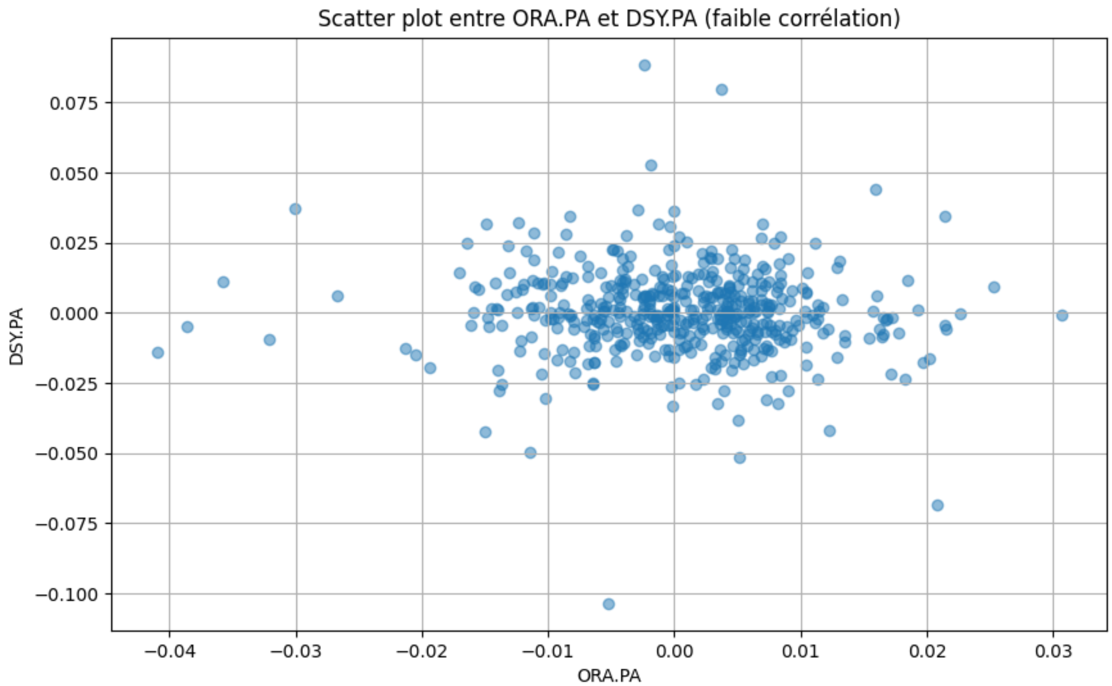

Scatter plot entre STMPA.PA et ORA.PA (faible corrélation)
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
ORA.PA
−0.10
−0.05
0.00
0.05
0.10
STMPA.PA

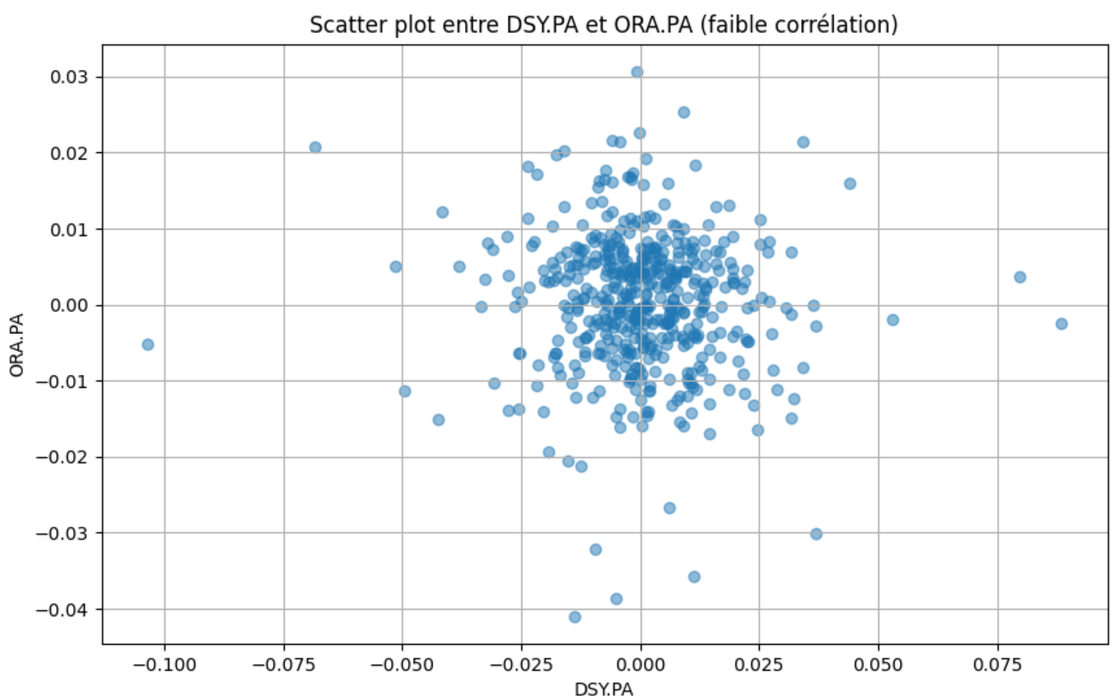

Scatter plot entre RMS.PA et ORA.PA (faible corrélation)
ORA.PA
RMS.PA
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08


Scatter plot entre ORA.PA et RMS.PA (faible corrélation)
RMS.PA
ORA.PA
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03


Scatter plot entre MC.PA et ORA.PA (faible corrélation)
ORA.PA
MC.PA
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100
0.125


Scatter plot entre ORA.PA et MC.PA (faible corrélation)
MC.PA
ORA.PA
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03


Scatter plot entre ORA.PA et DSY.PA (faible corrélation)
DSY.PA
ORA.PA


Scatter plot entre DSY.PA et ORA.PA (faible corrélation)
ORA.PA
DSY.PA

0.2 OPTIMISATION DE PORTEFEUILLE - MODELE DE MAKOWITZ

[108]:

```python
# Définition des éléments du modèle

## Rendements moyens des actifs
mean_returns = returns.mean()

## Matrice de covariance des actifs
cov_matrix = returns.cov()

## Les statistiques du portefeuille où le Taux sans risque = 2%

def portfolio_stats(weights):
    port_return = np.sum(weights * mean_returns) * 252
    port_volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix * 252,
 ↪weights)))
    sharpe_ratio = (port_return - 0.02) / port_volatility
    return np.array([port_return, port_volatility, sharpe_ratio])


## La minimisation de la volatilité
def minimize_volatility(weights):
    return portfolio_stats(weights)[1]


## Détermination des contraintes du modèle
constraints = {'type': 'eq', 'fun': lambda x: np.sum(x) - 1}
bounds = tuple((0, 1) for asset in range(len(mean_returns)))
initial_weights = np.array(len(mean_returns) * [1. / len(mean_returns)])


# Optimisation du portefeuille
optimal = sco.minimize(
    minimize_volatility,
    initial_weights,
    method='SLSQP',
    bounds=bounds,
    constraints=constraints
)

optimal_weights = optimal.x
print("Poids optimaux du portefeuille : \n \n ", optimal_weights)
```

```
Poids optimaux du portefeuille :

  [2.91659198e-02 2.44626806e-02 0.00000000e+00 4.35781511e-17
 2.15658182e-01 2.24971951e-18 2.84171452e-02 0.00000000e+00
 1.20075391e-17 0.00000000e+00 3.04472415e-02 0.00000000e+00
```

```
1.72024880e-02 1.09741589e-17 1.32998356e-01 4.32601684e-03
0.00000000e+00 2.28129708e-02 1.75640752e-17 3.37843474e-02
1.05879118e-17 5.66967511e-02 0.00000000e+00 0.00000000e+00
2.15981494e-01 4.14524509e-02 1.20168734e-02 2.23589259e-02
1.68728963e-18 7.66358317e-03 1.87302220e-02 3.28648784e-19
1.53075794e-17 8.01114258e-03 1.27259423e-02 0.00000000e+00
6.12282106e-02 3.11030498e-18 2.71491000e-17 3.85905576e-03]
```

[109]:
```python
# Générer les poids optimaux en indexant l'action correspondante
optimal_weights_df = pd.DataFrame({
    'Ticker': mean_returns.index,
    'Optimal Weight': np.round(optimal_weights,3)
})

# Le portefeuille optimal
portefeuille_optimal = pd.merge(info_tickers, optimal_weights_df, on="Ticker")
portefeuille_optimal = portefeuille_optimal[portefeuille_optimal["Optimal␣
 ↪Weight"]!=0]
print(f"\n Le portefeuille optimal est : \n \n {portefeuille_optimal.
 ↪sort_values(by= 'Optimal Weight', ascending=False)}")

# Les statistiques du portefeuille optimal
print(f"\n Avec un rendement de {portfolio_stats(optimal_weights)[0]:.2%}\n ")
print(f"Une volatilité de {portfolio_stats(optimal_weights)[1]:.2%} \n ")
print(f"Un ratio de Sharpe de {portfolio_stats(optimal_weights)[2]:.2f}")
```

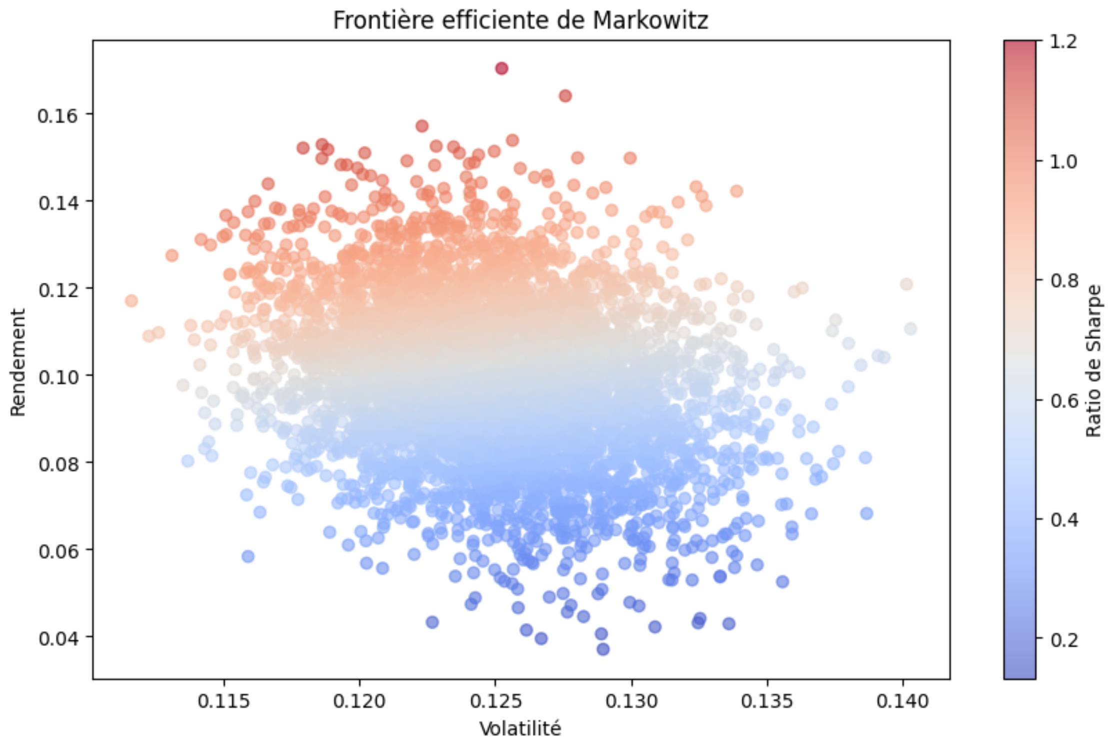
```
 Le portefeuille optimal est :

      Ticker              Company                  Sector  Optimal Weight
10     BN.PA               Danone      Consumer Defensive           0.216
22    ORA.PA               Orange  Communication Services           0.216
13   ENGI.PA                Engie               Utilities           0.133
35    TTE.PA         TotalEnergies                  Energy           0.061
21     ML.PA             Michelin             Industrials           0.057
24    PUB.PA             Publicis  Communication Services           0.041
19     LR.PA              Legrand             Industrials           0.034
11    DSY.PA     Dassault Systèmes              Technology           0.030
0      AC.PA                Accor       Consumer Services           0.029
8      CA.PA            Carrefour      Consumer Defensive           0.028
9     ACA.PA      Crédit Agricole      Financial Services           0.024
34     HO.PA               Thales             Industrials           0.023
16    RMS.PA               Hermès       Consumer Cyclical           0.022
28    SAN.PA               Sanofi              Healthcare           0.019
14     EL.PA      EssilorLuxottica              Healthcare           0.017
29     SU.PA   Schneider Electric             Industrials           0.013
23     RI.PA         Pernod Ricard      Consumer Defensive           0.012
26    SAF.PA               Safran             Industrials           0.008
```

```
32  STMPA.PA   STMicroelectronics            Technology           0.008
15    ERF.PA  Eurofins Scientific            Healthcare           0.004
39    VIV.PA              Vivendi  Communication Services           0.004
```

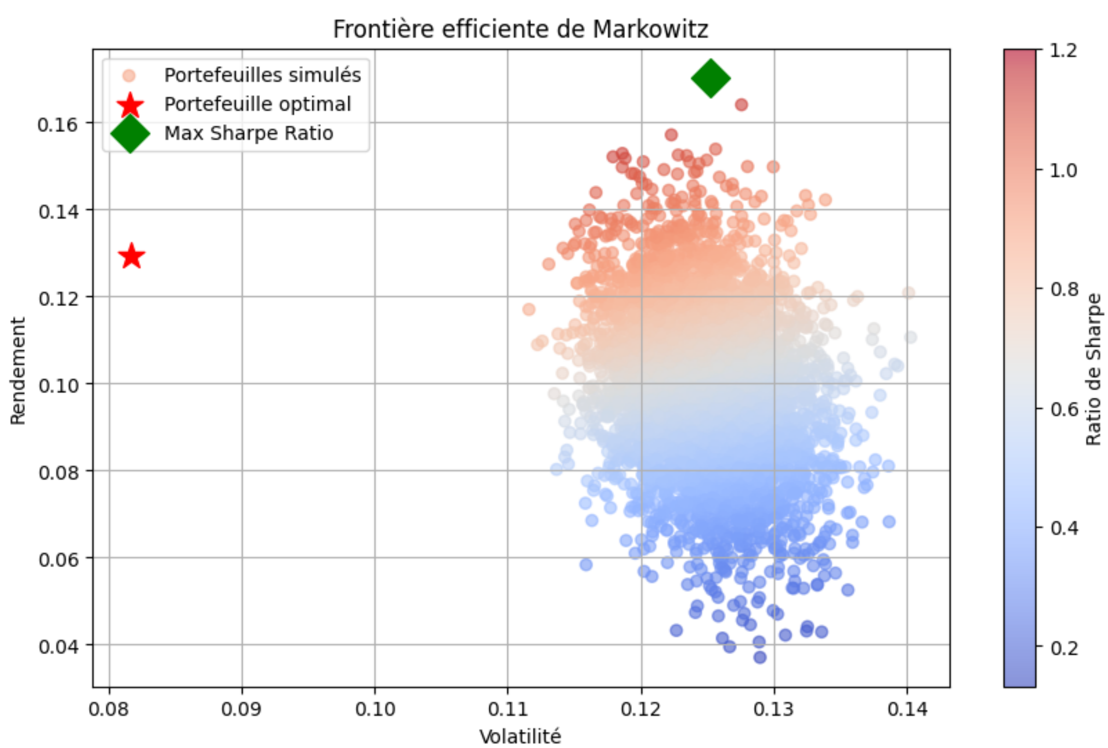
```
 Avec un rendement de 12.92%

Une volatilité de 8.17%

Un ratio de Sharpe de 1.34
```

[110]:

```python
# Simuler plusieurs portefeuilles aléatoires
num_portfolios = 5000
results = np.zeros((3, num_portfolios))
weights_record = []

for i in range(num_portfolios):
    weights = np.random.random(len(mean_returns))
    weights /= np.sum(weights)
    weights_record.append(weights)

    port_return, port_volatility, sharpe = portfolio_stats(weights)
    results[0, i] = port_return
    results[1, i] = port_volatility
    results[2, i] = sharpe

# Tracer la frontière efficiente
plt.figure(figsize=(10, 6))
plt.scatter(results[1, :], results[0, :], c=results[2, :], cmap='coolwarm',␣
 ↪alpha=0.6)
plt.xlabel('Volatilité')
plt.ylabel('Rendement')
plt.colorbar(label="Ratio de Sharpe")
plt.title('Frontière efficiente de Markowitz')
plt.show()
```


Frontière efficiente de Markowitz
Rendement
Volatilité
Ratio de Sharpe
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.115
0.120
0.125
0.130
0.135
0.140
1.2
1.0
0.8
0.6
0.4
0.2

[111]:
```
# Calculer les statistiques du portefeuille optimal
optimal_return, optimal_volatility, optimal_sharpe =␣
↪portfolio_stats(optimal_weights)

# Trouver le portefeuille avec le ratio de Sharpe le plus élevé
max_sharpe_idx = np.argmax(results[2])
max_sharpe_return = results[0, max_sharpe_idx]
max_sharpe_volatility = results[1, max_sharpe_idx]

# Tracer la frontière efficiente avec des améliorations
plt.figure(figsize=(10, 6))
plt.scatter(results[1, :], results[0, :], c=results[2, :], cmap='coolwarm',␣
↪alpha=0.6, label='Portefeuilles simulés')
plt.colorbar(label="Ratio de Sharpe")
plt.scatter(optimal_volatility, optimal_return, marker='*', color='r', s=200,␣
↪label='Portefeuille optimal')
plt.scatter(max_sharpe_volatility, max_sharpe_return, marker='D', color='g',␣
↪s=200, label='Max Sharpe Ratio')
plt.xlabel('Volatilité')
plt.ylabel('Rendement')
plt.title('Frontière efficiente de Markowitz')
plt.legend(loc='upper left')
plt.grid(True)
plt.show()
```



0.3 SUIVI DU PORTEFEUILLE D'INVESTISSEMENT

L'objectif est de suivre en temps réel le portefeuille optimal en tenant compte des fluctuations de l'activité économique et du marché, tout en automatisant le processus d'optimisation.

Une mise à jour automatique des poids du portefeuille optimal est effectuée tous les 30 jours en utilisant les deux dernières fonctions d'optimisation. Une fois la mise à jour terminée, je reçois automatiquement un message contenant les performances du nouveau portefeuille, incluant le rendement, la volatilité et le ratio de Sharpe.

- ***Fonction de performance***

[ ]:
```
# Fonction de performance
def check_performance():
    #  Recalcul des rendements et de la covariance sur une fenêtre de 30 jours
    rolling_weights = []
    rolling_portfolio_returns = pd.Series(index=returns.index)
    window_size = 30  # Rééquilibrage mensuel

    for i in range(window_size, len(returns), window_size):
        rolling_period = returns.iloc[i - window_size:i]  # Fenêtre des 30␣
↪derniers jours

        # Recalcul des rendements moyens et de la covariance
        mean_returns_rolling = rolling_period.mean()
        cov_matrix_rolling = rolling_period.cov()

        # Nouvelle optimisation de Markowitz avec les données récentes
        new_optimal = sco.minimize(
            minimize_volatility,
            initial_weights,
            method='SLSQP',
            bounds=bounds,
            constraints=constraints
        )

        new_weights = new_optimal.x
        rolling_weights.append(new_weights)

    def send_mail():
        import smtplib
        server = smtplib.SMTP_SSL('smtp.gmail.com',465)
        server.ehlo()
            #server.starttls()
        server.ehlo()
        server.login('abdoulayetangara722@gmail.com','motdepassedumail')

        subject = "Nouveau portefeuille optimal ! "
        body = "Hello ! Ablo, voici le nouveau portefeuille optimal : \n"
```

```
        body += f"Le rendement est de {portfolio_stats(new_weights)[0]:.2%}\n"
        body += f"La volatilité est de {portfolio_stats(new_weights)[1]:.2%}\n"
        body += f"Le ratio de Sharpe est de {portfolio_stats(new_weights)[2]:.
↪2f}"

        msg = f"Subject: {subject}\n\n{body}"

        server.sendmail(
            'abdoulayetangara722@gmail.com',
            msg)
        server.quit()
        print("Email envoyé avec succès !")

        send_mail()
    return new_weights, mean_returns_rolling
```

- ***Fonction de nouvelle optimisation***

[ ]:
```
# Fonction de nouvelle optimisation
def optimal_portfolio_func(weights, mean_returns):
    new_weights = weights
    def portfolio_stats(new_weights):
        port_return = np.sum(new_weights * mean_returns) * 252
        port_volatility = np.sqrt(np.dot(new_weights.T, np.dot(cov_matrix * 252,␣
 ↪new_weights)))
        sharpe_ratio = (port_return - 0.02) / port_volatility
        return np.array([port_return, port_volatility, sharpe_ratio])


    ## La minimisation de la volatilité
    def minimize_volatility(new_weights):
        return portfolio_stats(new_weights)[1]


    ## Détermination des contraintes du modèle
    constraints = {'type': 'eq', 'fun': lambda x: np.sum(x) - 1}
    bounds = tuple((0, 1) for asset in range(len(mean_returns)))
    initial_weights = np.array(len(mean_returns) * [1. / len(mean_returns)])


    # Optimisation du portefeuille
    optimal = sco.minimize(
        minimize_volatility,
        initial_weights,
        method='SLSQP',
        bounds=bounds,
        constraints=constraints
    )

    new_optimal_weights = optimal.x
    optimal_weights_df = pd.DataFrame({
    'Ticker': mean_returns.index,
    'Optimal Weight': np.round(optimal_weights,3)
    })
    rendement = f"Avec un rendement de {portfolio_stats(new_optimal_weights)[0]:.
 ↪2%}"
    volatilite = f"Une volatilité de {portfolio_stats(new_optimal_weights)[1]:.
 ↪2%}"
    ratio_sharpe = f"Un ratio de Sharpe de␣
 ↪{portfolio_stats(new_optimal_weights)[2]:.2f}"

    print(" Mise à jour mensuelle des poids du portefeuille !")
    return (optimal_weights_df, rendement, volatilite, ratio_sharpe)
```

- ***Automatisation du processus***

Cette automatisation vise à surveiller quotidiennement si une tâche permet de mettre à jour le portefeuille optimal et, tous les 30 jours, à recalculer les poids optimaux des actifs. Ce processus est conçu pour fonctionner en continu, même lorsque les appareils utilisés pour cette tâche sont éteints, garantissant ainsi une exécution ininterrompue.

[ ]:
```python
# Exécuter tous les 30 jours
schedule.every(30).days.do(check_performance)
schedule.every(30).days.do(optimal_portfolio_func)

# Boucle infinie pour exécuter la tâche planifiée
while True:
    schedule.run_pending()
    time.sleep(86400)  # Vérifier une fois par jour
```