RAPPORT DE PROJET

PYLOT-Online Portfolio Allocation

### par

### Tarik Bouchabchoub – Quentin Baldo – Zouhir Driouache

### Francesco Madrisotti

## 01/06/2023

**Introduction :**

Le présent rapport vise à fournir une analyse détaillée du projet de sélection de portefeuille "online" mené par notre équipe. L'objectif de ce projet était de concevoir et de développer une solution permettant aux utilisateurs de créer et de gérer des portefeuilles d'investissement personnalisés. Ce rapport mettra en évidence les principales étapes du projet, les choix technologiques effectués, les difficultés rencontrées et les résultats obtenus.

**Objet du projet :**

La Data dans le secteur de la finance de marché, est très importante et prend une place primordiale dans le processus d’investissement. Le domaine d’allocation de portefeuille est particulièrement impacté par la Data puisqu’elle constitue à la fois le point de départ de chaque investissement dans les marchés.

L’objectif de ce projet est de créer un modèle d’allocation de portefeuille qui adapte sa stratégie “online”, mais aussi d’adapter l’approche machine learning à un secteur d’activité présentant des points communs.

Dans cette phase initiale, notre équipe a effectué une analyse approfondie des besoins des utilisateurs potentiels. Nous avons identifié les fonctionnalités essentielles, telles que la gestion des actifs, la visualisation des performances et l'accès à des informations en temps réel sur les marchés financiers. Cette analyse a été cruciale pour définir la portée et les exigences du projet.

**Exploration et Visualisation des données :**

La première partie porte sur l'exploration de trois secteurs du CAC40 : l'industrie, le luxe et la finance. Nous utiliserons l'API YahooFinance comme source principale de données financières pour analyser l'évolution des actions associées au CAC40.

Pour commencer, nous avons utilisé les packages pandas\_datareader et yfinance en utilisant les commandes suivantes :

* **pip install pandas\_datareader**
* **pip install yfinance**

Ces packages nous permettent d'extraire les données financières associées au CAC40 à partir de l'API YahooFinance.

from datetime import datetime

import seaborn as sns

import matplotlib.pyplot as plt

import yfinance as yf

#CAC40 histo

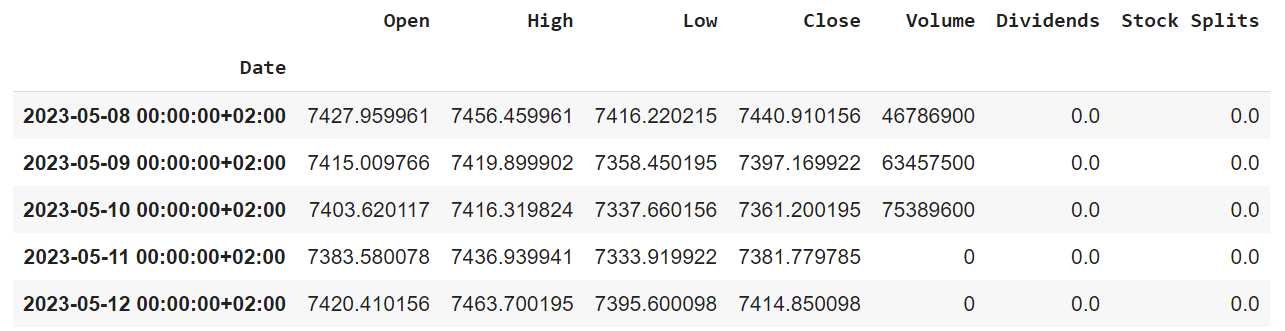
data = yf.Ticker("^FCHI")

end = datetime.now()

start = datetime(end.year - 4, end.month, end.day)

data\_CAC40 = data.history(period="4y")

data\_CAC40.tail()



Le package yfinance permet de récupérer moyennant les paramètres de data l'historique des KPI financiers suivants:

* Date : date de la cotation
* High : le prix le plus haut atteint de la période
* Low : le prix le plus bas de la période
* Open : le prix de cotation au début de la période
* Close : le prix de cotation a la fermeture de la période
* Volume : Le nombre de titres échangés de la période

#init

sns.set\_theme()

#plot du niveau d'indice

data\_CAC40.plot (y = ['Close'], figsize =  (16,10))

plt.title ('CAC40')

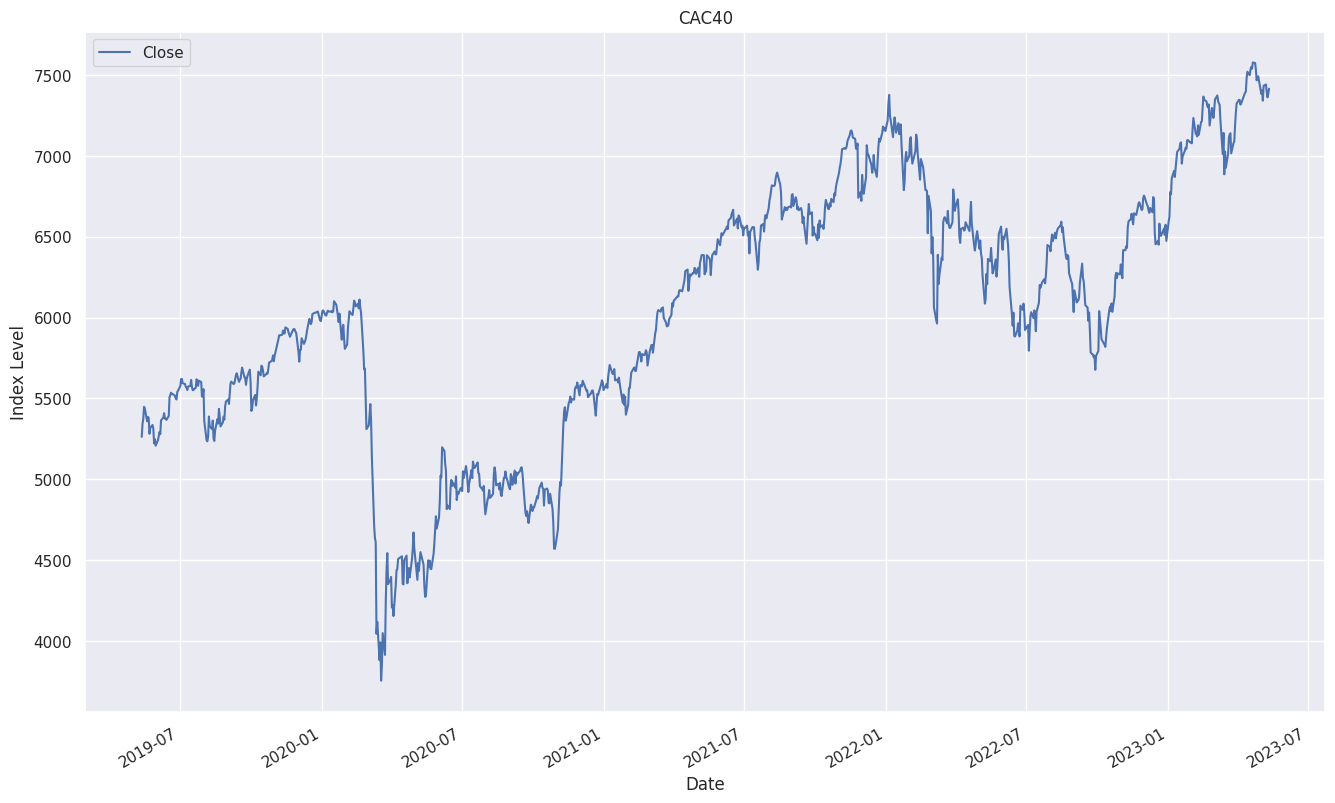
plt.ylabel ('Index Level')

plt.legend()

plt.show();

Évolution de l'indice CAC40:

Nous avons extrait les données de l'indice CAC40 sur une période de 4 ans à l'aide du package yfinance. En traçant les données de clôture, nous avons observé une chute significative au premier trimestre 2020, correspondant à l'impact de la pandémie du COVID-19. Par la suite, à partir du deuxième trimestre, nous avons remarqué une tendance haussière, notamment grâce à la production de vaccins et à la levée des restrictions liées à la pandémie.



Actions du secteur financier:

Nous avons ensuite examiné les actions du secteur financier. Nous avons sélectionné quatre entreprises : AXA, Societe Generale, BNP Paribas et Crédit Agricole. En utilisant le package yfinance, nous avons récupéré les données historiques de ces actions sur une période de 4 ans. Les Equities du secteur financier ont une tendance stable mais subissent fortement la crise du covid 19 en 2020, aussi on peut remarquer visuelement que la courbe des equities du secteur financier est fortement similaire à celle de l'indice CAC40, ce qui signifie que le secteur financier impacte fortement l'indice et donc contribue directement à la performance de l'indice

sns.set\_theme()

import pandas as pd

from datetime import datetime

import seaborn as sns

import matplotlib.pyplot as plt

import yfinance as yf

ticker = ['CS.PA',

'GLE.PA',

'BNP.PA',

'ACA.PA']

Consumer = ['AXA','SOCIETE GENERALE',

'BNP PARIBAS','CREDIT AGRICOLE'

]

d = {}

plt.figure ( figsize = (20,10)  )   #taille de la figure

#get data from yahoo avec les tickers YahooFinance

for company , ticket in zip(Consumer ,ticker) :

    try:

        y\_data = yf.Ticker(ticket)

        d[company]=y\_data.history(period="4y")

    except:

        continue

    plt.plot( d [company].index, d [company]['Close'], label = company)   #plot

#plot

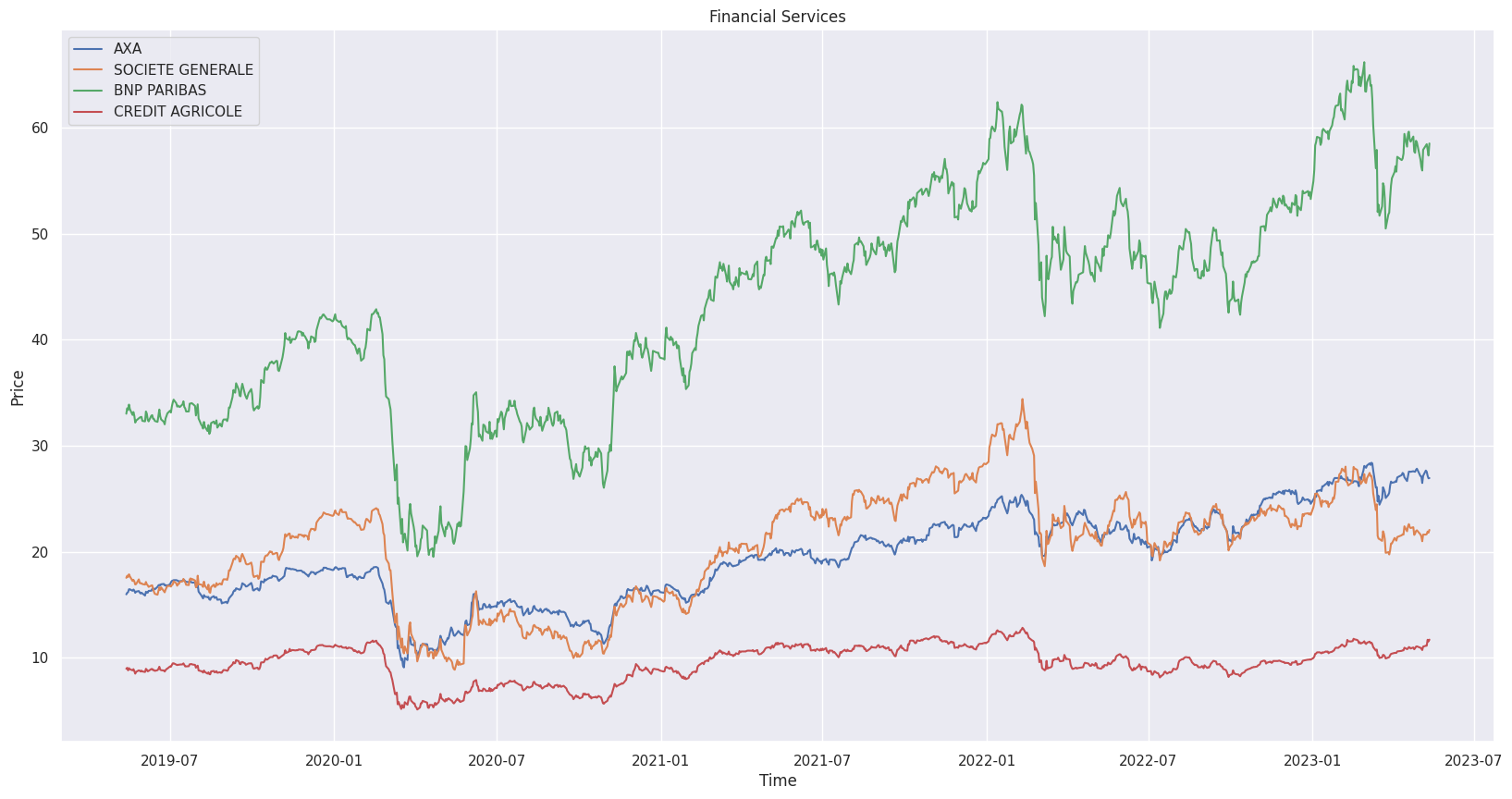
plt.title ('Financial Services')

plt.xlabel ('Time')

plt.ylabel ('Price')

plt.legend(loc='upper left')

plt.show();



Analyse des actions individuelles:

Nous avons ensuite analysé les actions individuelles des entreprises sélectionnées. Nous avons tracé l'historique des cours de clôture et le volume échangé pour chaque action. Cela nous a permis de visualiser l'évolution des prix des actions et de comprendre l'impact du volume sur les variations des prix.

Afin de différencier les tendances par action, on va afficher l'historique des cours de clôture par action

plt.figure(figsize=(15, 6))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(Consumer, 1):

    plt.subplot(2, 2, i)

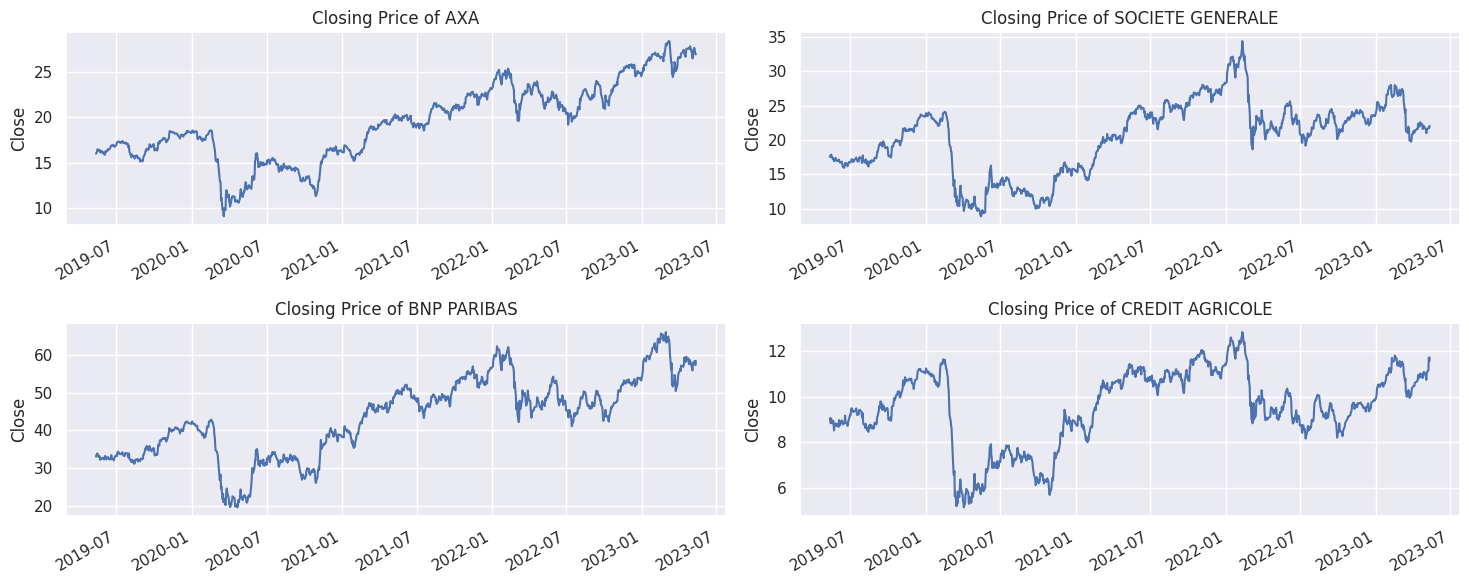
    d[company]['Close'].plot()

    plt.ylabel('Close')

    plt.xlabel(None)

    plt.title(f"Closing Price of {Consumer[i - 1]}")

plt.tight\_layout()



Le cours de l'action nous donne une idée de l'évolution de l'action unitaire au niveau du marché, le prix d'action dépend aussi du volume tradé courant la période.

plt.figure(figsize=(15, 6))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(Consumer, 1):

    plt.subplot(2, 2, i)

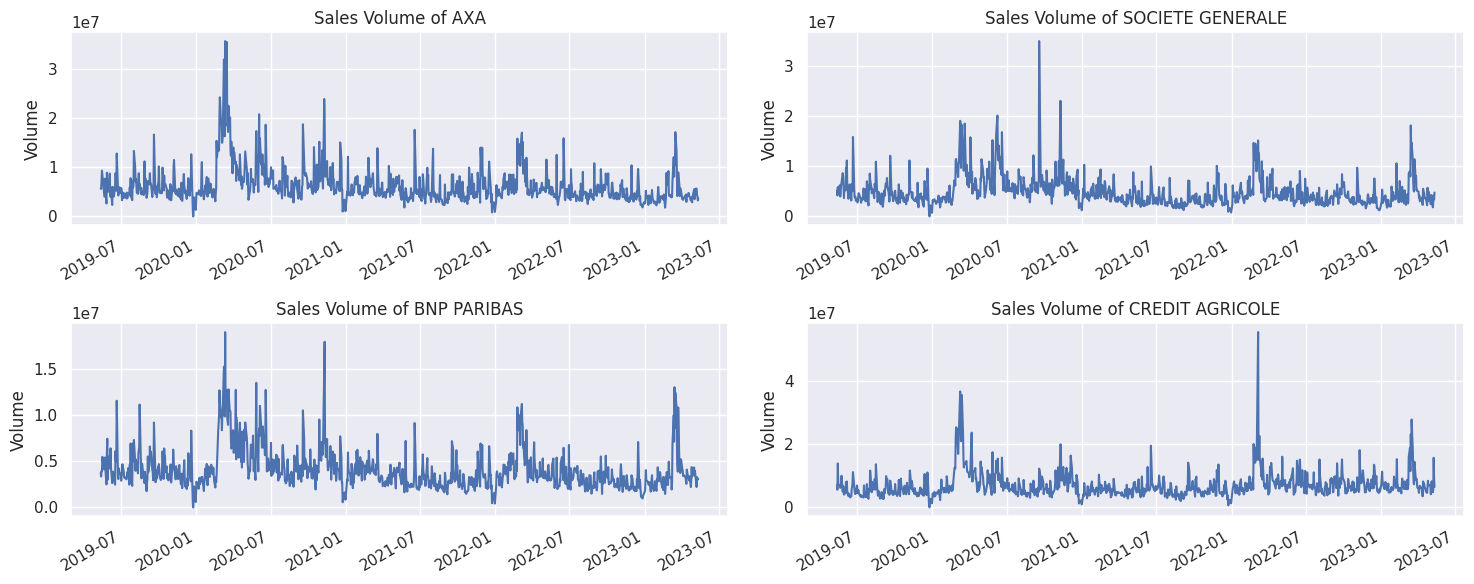
    d[company]['Volume'].plot()

    plt.ylabel('Volume')

    plt.xlabel(None)

    plt.title(f"Sales Volume of {Consumer[i - 1]}")

plt.tight\_layout()



Calcul de la moyenne mobile:

Nous avons calculé la moyenne mobile du cours de l'action pour différentes fenêtres temporelles (10, 20 et 50 jours). Cela nous a permis de lisser les variations à court terme et d'identifier les tendances à plus long terme

ma\_day = [10, 20, 50]

for ma in ma\_day:

    for company in Consumer:

        column\_name = f"MA for {ma} days"

        d[company][column\_name] = d[company]['Close'].rolling(ma).mean()

fig, axes = plt.subplots(nrows=2, ncols=2)

fig.set\_figheight(8)

fig.set\_figwidth(15)

d['AXA'][['Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])

axes[0,0].set\_title('AXA')

d['SOCIETE GENERALE'][['Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])

axes[0,1].set\_title('SOCIETE GENERALE')

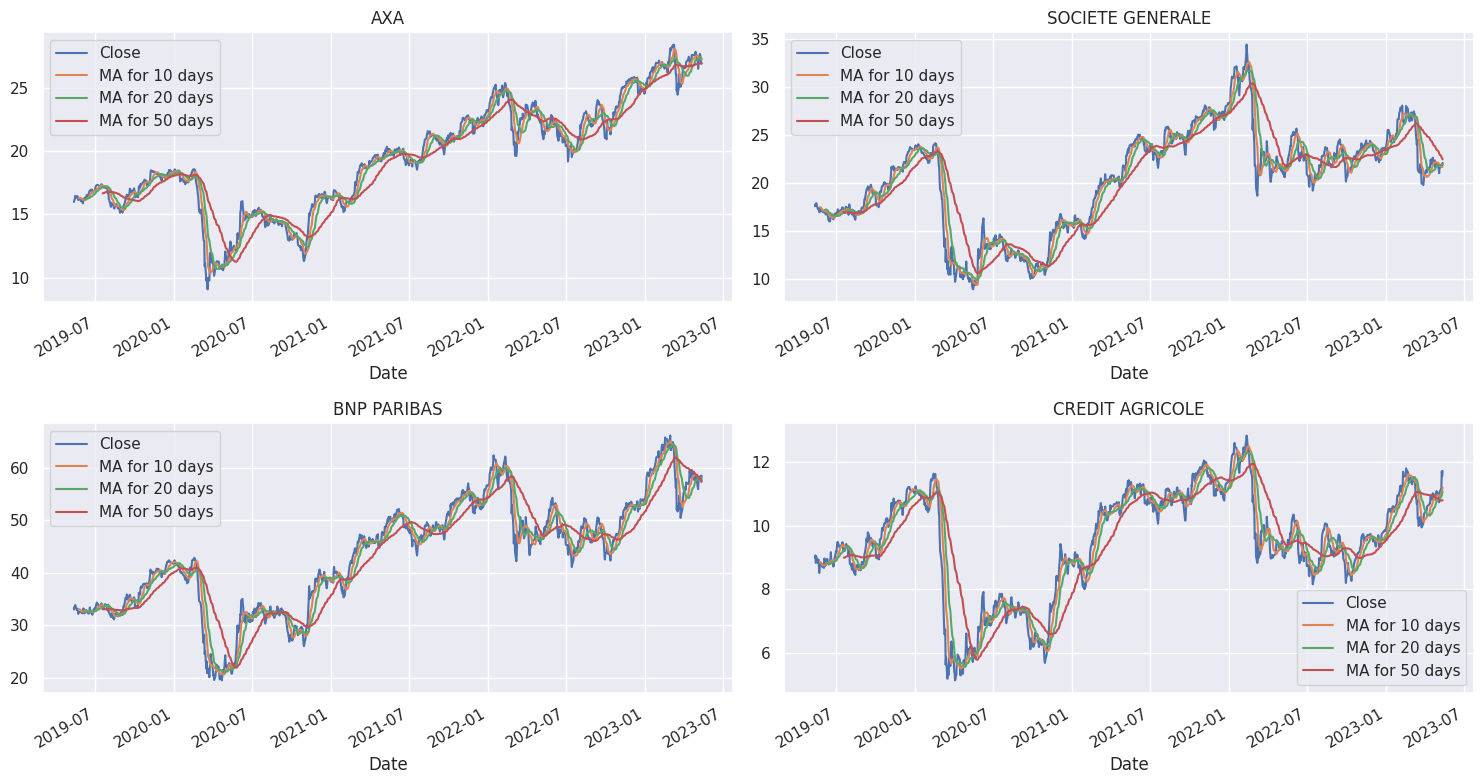
d['BNP PARIBAS'][['Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])

axes[1,0].set\_title('BNP PARIBAS')

d['CREDIT AGRICOLE'][['Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])

axes[1,1].set\_title('CREDIT AGRICOLE')

fig.tight\_layout()



Calcul du rendement quotidien:

Nous avons calculé le rendement quotidien de chaque action en utilisant la fonction pct\_change() du package pandas. En traçant l'historique des rendements quotidiens, nous avons pu observer les variations et les performances des actions.

#on va utiliser pct\_change pour calculer la variation quotidienne

for company in Consumer:

    d[company]['Daily Return'] = d[company]['Close'].pct\_change()

fig, axes = plt.subplots(nrows=2, ncols=2)

fig.set\_figheight(20)

fig.set\_figwidth(30)

d['AXA']['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')

axes[0,0].set\_title('AXA')

d['SOCIETE GENERALE']['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')

axes[0,1].set\_title('SOCIETE GENERALE')

d['BNP PARIBAS']['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')

axes[1,0].set\_title('BNP PARIBAS')

d['CREDIT AGRICOLE']['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')

axes[1,1].set\_title('CREDIT AGRICOLE')

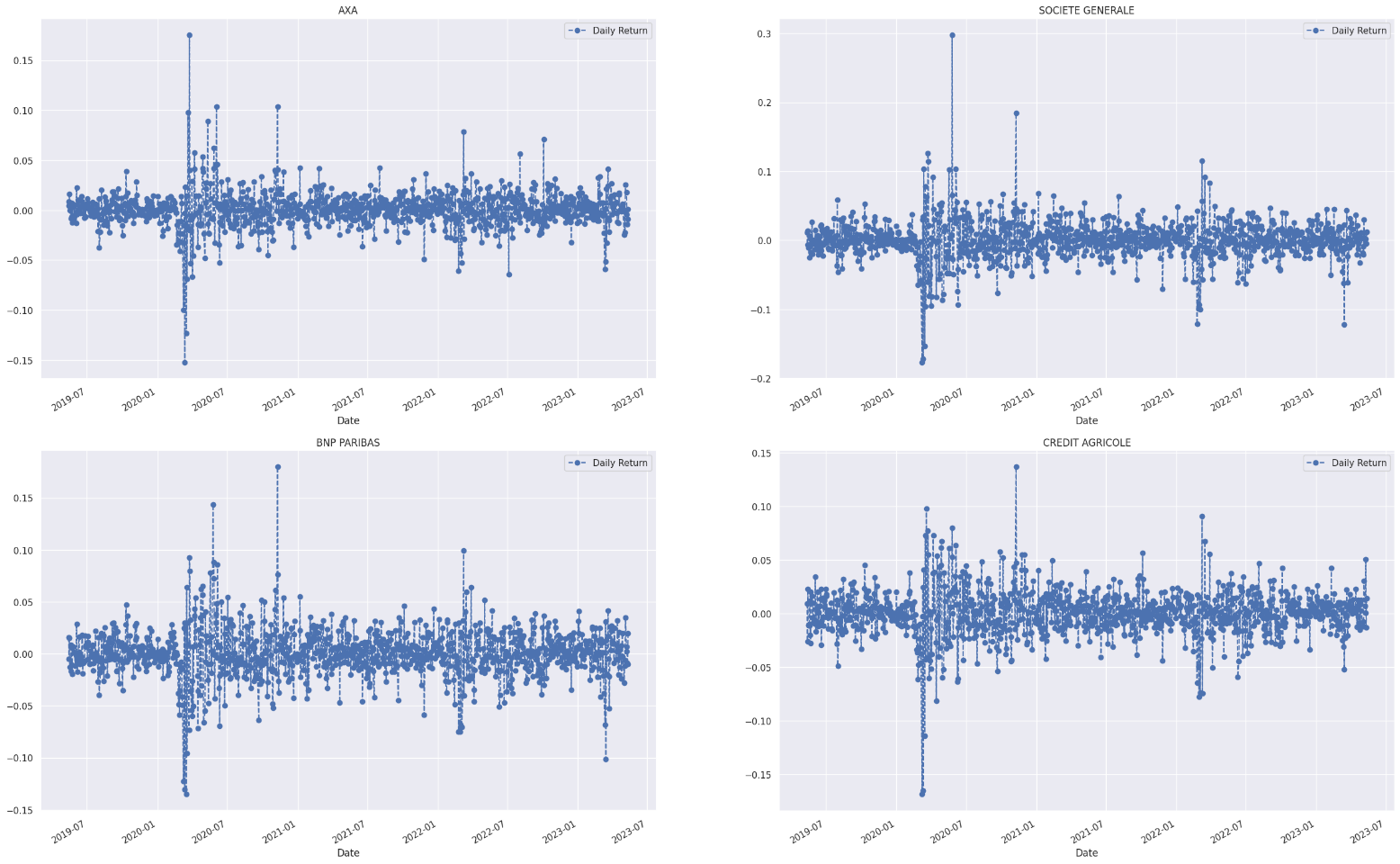


fig.tight\_layout()

plt.figure(figsize=(12, 7))

for i, company in enumerate(Consumer, 1):

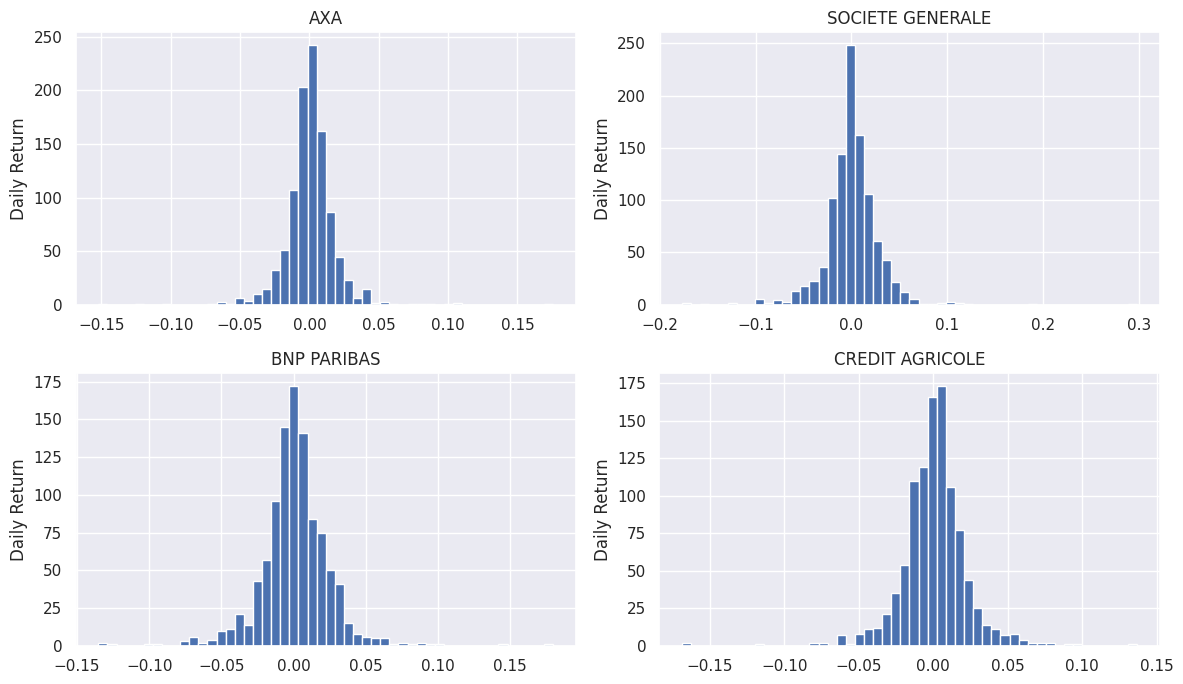
    plt.subplot(2, 2, i)

    d[company]['Daily Return'].hist(bins=50)

    plt.ylabel('Daily Return')

    plt.title(f'{Consumer[i - 1]}')

plt.tight\_layout()



Analyse de la corrélation entre la sélection d'actions financières du CAC40:

Nous pouvons donc maintenant voir que si deux actions sont parfaitement (et positivement) corrélées entre elles, une relation linéaire entre leurs valeurs de rendement quotidiennes devrait se produire.

Seaborn et pandas permettent de répéter très facilement cette analyse comparative pour chaque combinaison possible d'actions dans notre liste d'equities. Nous utiliserons sns.pairplot()

Ytickers\_df=yf.download(tickers = ticker,  # list of tickers

            period = "4y",         # time period

            interval = "1d",       # trading interval

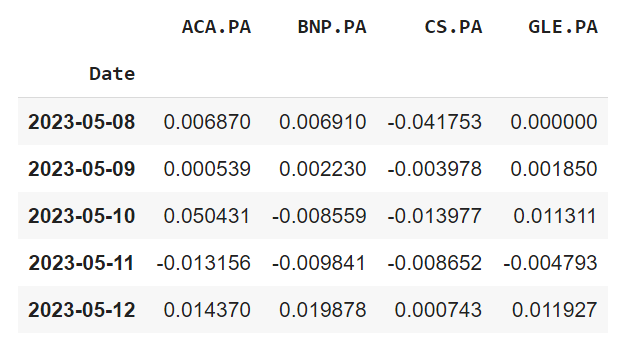
            ignore\_tz = True,      # ignore timezone when aligning data from different exchanges?

            prepost = False)

closing\_df=Ytickers\_df['Close']

Fin\_returns=closing\_df.pct\_change()

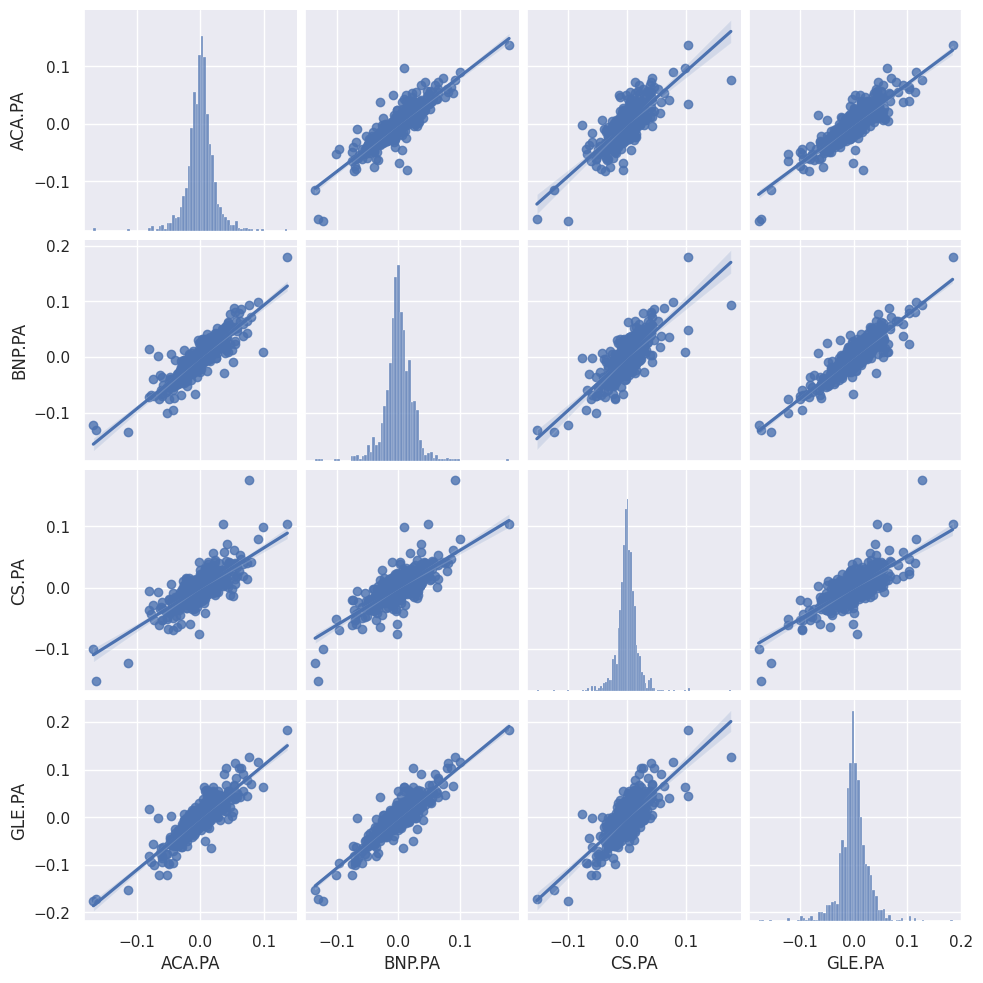
Fin\_returns.tail()

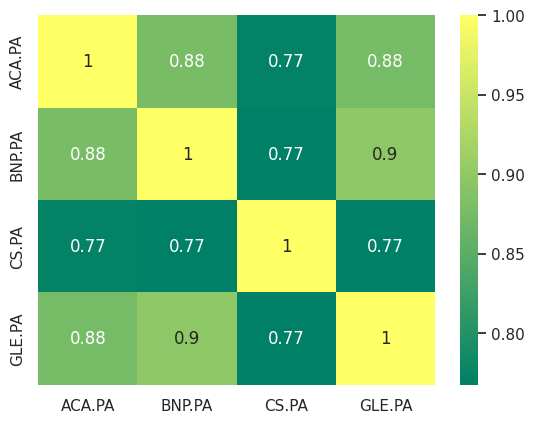


Corrélation entre les actions:

Nous avons analysé la corrélation entre les actions sélectionnées en utilisant la fonction pairplot() du package seaborn. Cette analyse nous a permis de visualiser les relations linéaires entre les rendements quotidiens des différentes actions.

sns.pairplot(Fin\_returns,kind='reg')



sns.heatmap(Fin\_returns.corr(), annot=True, cmap='summer')

# Set up our figure by naming it returns\_fig, call PairPLot on the DataFrame

return\_fig = sns.PairGrid(Fin\_returns.dropna())

# Using map\_upper we can specify what the upper triangle will look like.

return\_fig.map\_upper(plt.scatter, color='purple')

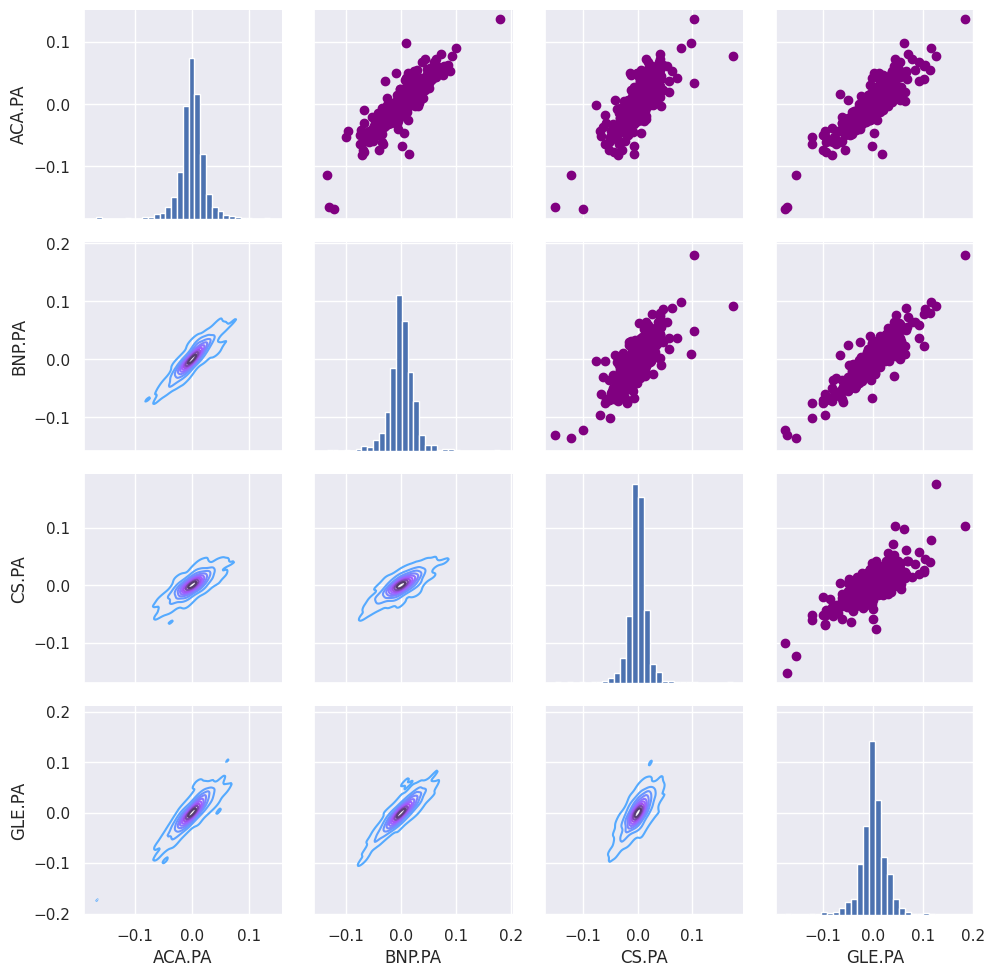
# We can also define the lower triangle in the figure, inclufing the plot type (kde)

# or the color map (BluePurple)

return\_fig.map\_lower(sns.kdeplot, cmap='cool\_d')

# Finally we'll define the diagonal as a series of histogram plots of the daily return

return\_fig.map\_diag(plt.hist, bins=30)



Analyse du risque:

Enfin, nous avons quantifié le risque de notre portefeuille d'actions en comparant le rendement attendu à l'écart type des rendements quotidiens. En traçant le rendement attendu en fonction du risque, nous avons pu identifier les actions présentant un rapport risque/rendement optimal.

import numpy as np

rets = Fin\_returns.dropna()

area = np.pi \* 20

plt.figure(figsize=(10, 8))

plt.scatter(rets.mean(), rets.std(), s=area)

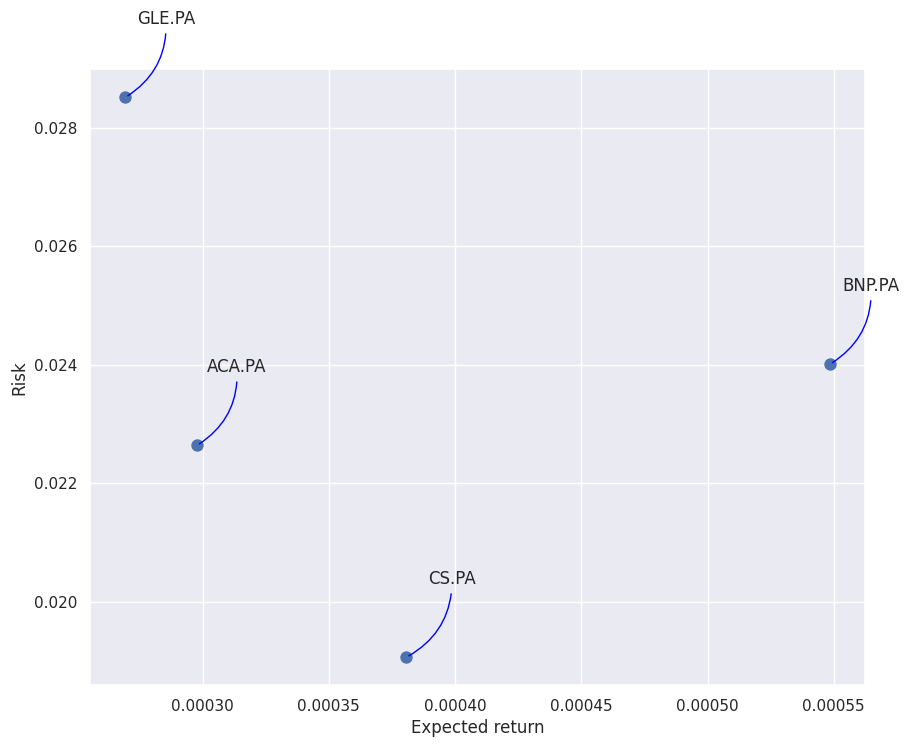
plt.xlabel('Expected return')

plt.ylabel('Risk')

for label, x, y in zip(rets.columns, rets.mean(), rets.std()):

    plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='bottom',

                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))



**Modélisation des données :**

* **CAPM ( MEDAF : Modèle d’évaluation des actifs financiers)**
* **Mouvement Brownien Géométrique**
* **LSTM**

**CAPM :**

Cette partie se concentre sur l'application du modèle d’évaluation des actifs financiers (CAPM) pour analyser les rendements des actifs par rapport à un indice de référence (CAC40). Le CAPM est un modèle qui permet d'estimer la rentabilité attendue d'un actif en fonction de son risque systématique.

Outre la prédiction des cours des actions,un autre moyen permet de se positionner par rapport au marché "la rentabilité estimée", le modèle MEDAF permet de quantifier cette rentabilité. Pour le modèle MEDAF, une analyse YTD est plus pertinente, le dataset se limitera donc sur l'année 2022-2023

Analyse des rendements des actifs et de l'indice de référence (CAC40):

Nous avons collecté les données du CAC40 sur une période d'un an, de 2022 à 2023, en utilisant le package yfinance. En calculant les rendements quotidiens des actions du CAC40 et de l'indice lui-même, nous avons pu analyser les variations et les relations entre les rendements. Aussi,nous avons tracé des graphiques de dispersion pour chaque actif par rapport à l'indice CAC40 afin d'observer la corrélation entre les rendements. Cela nous a permis de visualiser la relation linéaire entre les rendements des actifs individuels et l'indice.

import numpy as np

#Data du CAC40:

end1 = datetime.now()

start1 = datetime(end.year - 1, end.month, end.day)

###############################

ticker\_1= ['CS.PA',

'GLE.PA',

'BNP.PA',

'ACA.PA','^FCHI']

Ytickers\_CPAM=yf.download(tickers = ticker\_1,  # list of tickers

            period = "1y",         # time period

            interval = "1d",       # trading interval

            ignore\_tz = True,      # ignore timezone when aligning data from different exchanges?

            prepost = False)

Fin\_returns\_1Y = Ytickers\_CPAM['Close'].pct\_change()

Fin\_returns\_1Y = Fin\_returns\_1Y.fillna(0)

Fin\_returns\_1Y.head()

Plot\_t = ['CS.PA',

'GLE.PA',

'BNP.PA',

'ACA.PA']

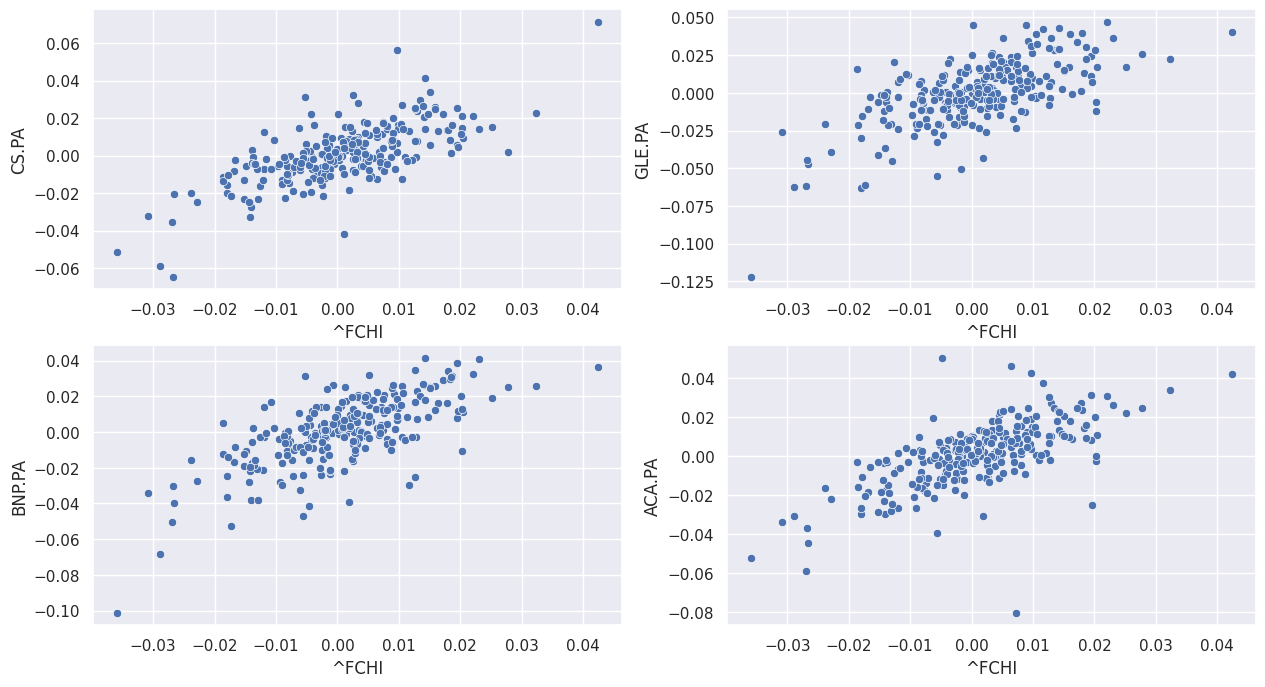
# plot a scatter plot between AXA and the CAC40 (Market)

plt.figure(figsize=(15,8))

for i,t in enumerate(Plot\_t,1):

  plt.subplot(2, 2, i)

  sns.scatterplot(data=Fin\_returns\_1Y,y=t,x='^FCHI')



Calcul du bêta et de l'alpha:

En utilisant la méthode des moindres carrés, nous avons calculé le bêta et l'alpha de chaque actif par rapport à l'indice CAC40. Le bêta mesure la sensibilité d'un actif aux mouvements de l'indice, tandis que l'alpha représente le rendement excédentaire de l'actif par rapport à celui prédit par le modèle CAPM. Nous avons affiché les valeurs de bêta et d'alpha pour chaque actif.

#Calculons à présent le Beta et l'alpha des actions:\*\*

b={}

a={}

for t in Plot\_t:

  beta, alpha = np.polyfit(Fin\_returns\_1Y['^FCHI'],Fin\_returns\_1Y[t], 1)

  b[t]=beta

  a[t]=alpha

  print('Beta for {} stock is = {} and alpha is = {}'.format(t, round(beta,8), round(alpha,8)))

Beta for CS.PA stock is = 0.99401459 and alpha is = 4.683e-05

Beta for GLE.PA stock is = 1.26595333 and alpha is = -0.00087394

Beta for BNP.PA stock is = 1.20062189 and alpha is = -0.00022034

Beta for ACA.PA stock is = 1.01966181 and alpha is = -3.494e-05

Afin de poursuivre notre analyseous avons tracé à nouveau les graphiques de dispersion, cette fois en incluant la droite de régression linéaire correspondant au modèle CAPM. Cela nous a permis de visualiser la relation entre les rendements des actifs et l'indice de référence, ainsi que d'évaluer l'ajustement des données au modèle CAPM.

# Now let's plot the scatter plot and the straight line on one plot

plt.figure(figsize=(15,8))

for i,t in enumerate(Plot\_t,1):

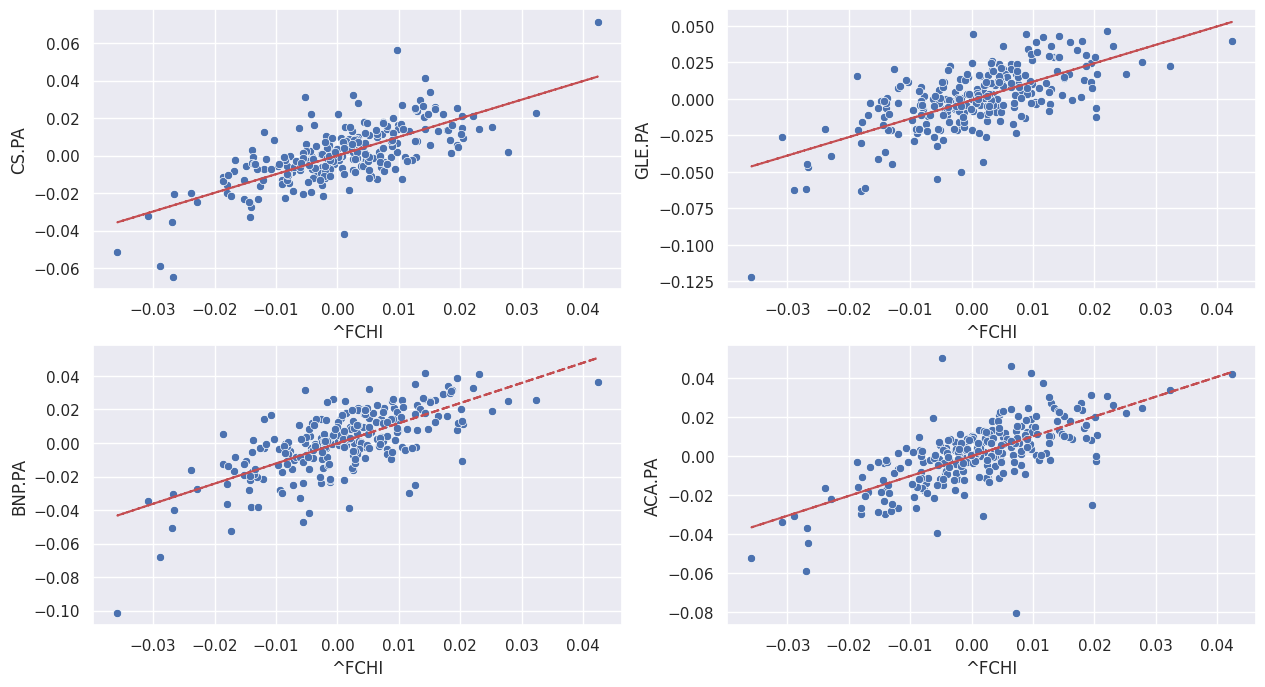
  plt.subplot(2, 2, i)

  sns.scatterplot(data=Fin\_returns\_1Y,y=t,x='^FCHI')

  plt.plot(Fin\_returns\_1Y['^FCHI'], b[t] \* Fin\_returns\_1Y['^FCHI'] + a[t], '--', color = 'r')

# Straight line equation with alpha and beta parameters

# Straight line equation is y = beta \* rm + alpha



Calcul de la rentabilité espérée des actifs:

En utilisant les paramètres du modèle CAPM (bêta, rendement du marché et taux sans risque), nous avons calculé la rentabilité espérée de chaque actif. La rentabilité espérée représente le rendement attendu de l'actif en fonction de son risque systématique.

Fin\_returns\_1Y['^FCHI'].mean()\*100

#Beta:

print(b) #Beta of each Equity

rm = round(Fin\_returns\_1Y['^FCHI'].mean()\*252.3\*100,8)

rm

rf=3.0490 #OAT 10 ans

for t in Plot\_t:

  ER\_Equities = round(rf + ( b[t] \* (rm-rf) ) ,3) # Calculate return for each equity vs Bench

  print(ER\_Equities,"est la rentabilité espérée de l'asset : ",t)

18.859 est la rentabilité espérée de l'asset : CS.PA

23.184 est la rentabilité espérée de l'asset : GLE.PA

22.145 est la rentabilité espérée de l'asset : BNP.PA

19.267 est la rentabilité espérée de l'asset : ACA.PA

Conclusion:

L'application du modèle CAPM nous a permis d'analyser les rendements des actifs par rapport à l'indice de référence CAC40. Nous avons calculé le bêta et l'alpha de chaque actif, tracé des graphiques de dispersion et évalué l'ajustement des données au modèle CAPM. Ces analyses nous ont permis de quantifier le risque systématique des actifs et de calculer leur rentabilité espérée en fonction du marché. Cela peut être utile pour les investisseurs dans la prise de décisions éclairées sur l'allocation de leur portefeuille d'actifs.

**Mouvement Brownien Géométrique :**

Les mouvements browniens géométriques supposent que les rendements (variation en pourcentage du prix d'une action) sont aléatoires et que leur distribution suit une loi normale.

Ces simulations permettent d'obtenir des scénarios possibles pour l'évolution future du prix de l'action, en tenant compte de ses fluctuations historiques et de la tendance générale. Il est important de noter que ces simulations sont basées sur des hypothèses simplificatrices et ne garantissent pas la précision des prévisions. Elles sont plutôt utilisées pour évaluer les risques et les opportunités associés aux mouvements futurs du prix de l'action.

Afin d’appliquer cette modélisation nous avons procédé comme suit :

* Séparation des données en deux ensembles : données d'entraînement (jusqu'au 31 décembre 2018) et données de test (à partir du 1er janvier 2019).
* Calcul des rendements quotidiens (la variation en pourcentage du prix d'un jour à l'autre), ainsi que le rendement moyen et l'écart-type (volatilité) des rendements sur les données d'entraînement.
* Définition des paramètres de la simulation : le nombre de simulations à effectuer (num\_simulations) et l'horizon de prévision (nombre de jours à prédire, défini comme la longueur des données de test).
* Simulation des trajectoires de prix en utilisant un mouvement brownien géométrique. Pour chaque simulation, des rendements aléatoires sont générés en utilisant une distribution normale avec le rendement moyen et l'écart-type calculés précédemment. Ces rendements sont ensuite convertis en prix en prenant le dernier prix des données d'entraînement et en le multipliant par l'accumulation des rendements (c'est-à-dire (1 + rendements).cumprod()).
* Calcul de la moyenne et de la médiane des trajectoires de prix simulées pour chaque jour.

import numpy as np

import pandas as pd

import yfinance as yf

import matplotlib.pyplot as plt

# Téléchargez les données de l'action LVMH

symbol = "MC.PA"

start\_date = "2010-01-01"

end\_date = "2021-12-31"

data = yf.download(symbol, start=start\_date, end=end\_date)

prices = data["Adj Close"]

# Séparez les données d'entraînement et de test

train\_data = prices.loc[:"2018-12-31"]

test\_data = prices.loc["2019-01-01":]

# Calculez les rendements et les statistiques

returns = train\_data.pct\_change().dropna()

mean\_return = returns.mean()

volatility = returns.std()

# Paramètres de la simulation

num\_simulations = 50

forecast\_horizon = len(test\_data)

# Générer les trajectoires de prix en utilisant un mouvement brownien géométrique

last\_price = train\_data.iloc[-1]

simulated\_prices = np.zeros((forecast\_horizon, num\_simulations))

dates = pd.date\_range(train\_data.index[-1], periods=forecast\_horizon + 1)[1:]

for i in range(num\_simulations):

    random\_returns = np.random.normal(mean\_return, volatility, forecast\_horizon)

    simulated\_prices[:, i] = last\_price \* (1 + random\_returns).cumprod()

# Calculez la moyenne des projections

mean\_simulated\_prices = simulated\_prices.mean(axis=1)

# Calculez la mediane des projections

median\_simulated\_prices = np.median(simulated\_prices, axis=1)

# Affichez les trajectoires de prix simulées et les prix réels

plt.figure(figsize=(14, 7))

plt.plot(prices, label="Prix simulé")

plt.plot(prices, label="Prix réel")

for i in range(num\_simulations):

    plt.plot(dates, simulated\_prices[:, i], alpha=0.5, color='blue')

plt.plot(dates, mean\_simulated\_prices, color='red', linewidth=2, label="Moyenne des projections")

plt.plot(dates, median\_simulated\_prices, color='green', linewidth=2, label="Médiane des projections")

plt.title("Projections de prix basées sur un mouvement brownien géométrique ")

plt.xlabel("Date")

plt.ylabel("Prix")

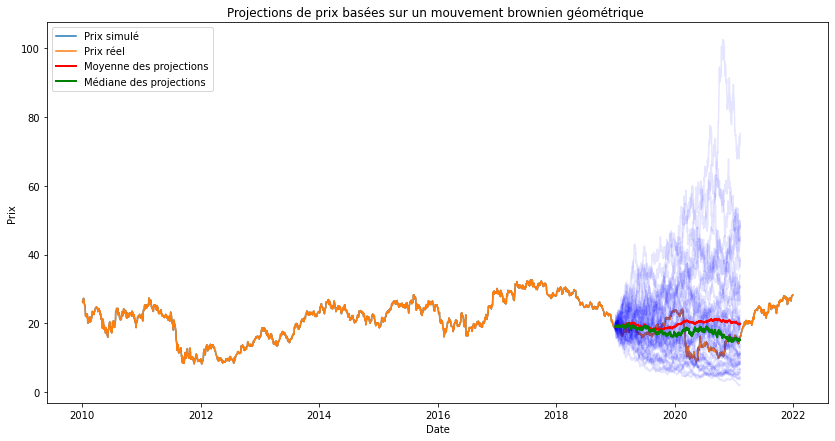
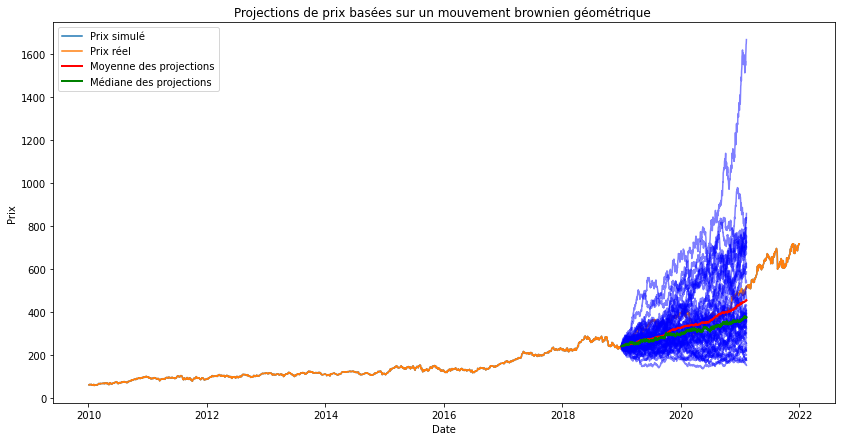
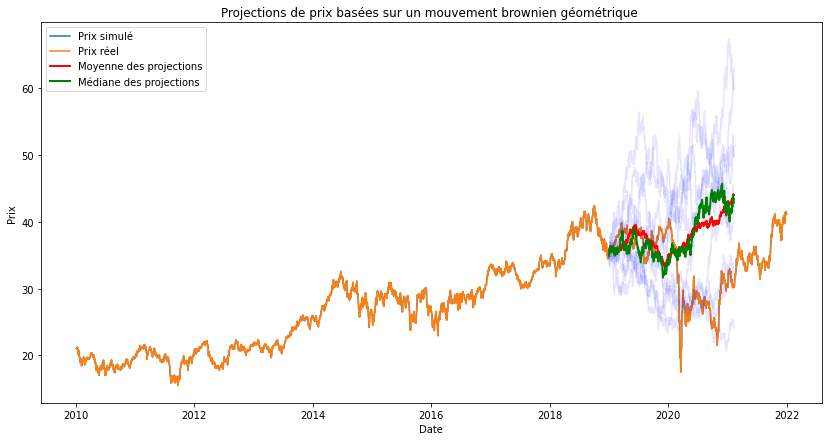
plt.legend()

plt.show()

Dans l'ensemble, ce code est une illustration basique d'une technique de modélisation des prix des actifs en finance qui utilise le concept de mouvement brownien géométrique, très populaire dans le cadre du modèle de Black-Scholes pour les *price options*.

Société Générale

LVMH



TotalEnergies

**LSTM :**

Cette partie se concentre sur l'application du modèle LSTM (Long Short-Term Memory) pour la prédiction des prix de clôture des actions financières. Pour cela, nous avons utilisé l'exemple de l’action AXA, afin de mesurer et prédire le cours à court terme.

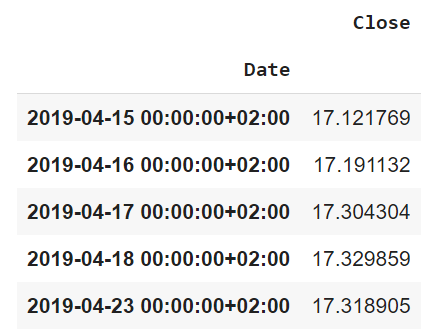
Nous avons commencé par explorer les données en traçant les prix de clôture de l’action au fil du temps. Cela nous a permis d'observer les tendances et les variations des prix.

Nous avons préparé les données en sélectionnant une action spécifique (AXA) et en filtrant uniquement la colonne des prix de clôture. Ensuite, nous avons converti les dates au format datetime et tracé le graphique des prix de clôture.

import numpy as np

d['AXA'].head()

df = d['AXA'].filter(['Close'])



Les bibliothèques importées sont les suivantes :

datetime : Utilisée pour effectuer des opérations sur les dates et les heures.

matplotlib.pyplot : Utilisée pour la visualisation des données.

sklearn.preprocessing.MinMaxScaler : Utilisée pour mettre à l'échelle les données d'entrée.

tensorflow.keras.models.Sequential : Utilisée pour créer un modèle séquentiel de réseau de neurones.

tensorflow.keras.optimizers.Adam : Utilisée pour définir l'optimiseur Adam pour l'entraînement du modèle.

tensorflow.keras.layers : Utilisée pour définir les couches du modèle de réseau de neurones.

Afin de traiter les séries temporelles une fonction str\_to\_datetime a été implémenté comme suit :

La fonction prend une chaîne de caractères au format "AAAA-MM-JJ" en entrée et la convertit en objet datetime.datetime correspondant. Elle divise la chaîne en parties (année, mois, jour) à l'aide de la fonction split, convertit ces parties en entiers et crée un objet datetime.datetime à partir des valeurs obtenues.

  import datetime

  def str\_to\_datetime(s):

    split = s.split('-')

    year, month, day = int(split[0]), int(split[1]), int(split[2])

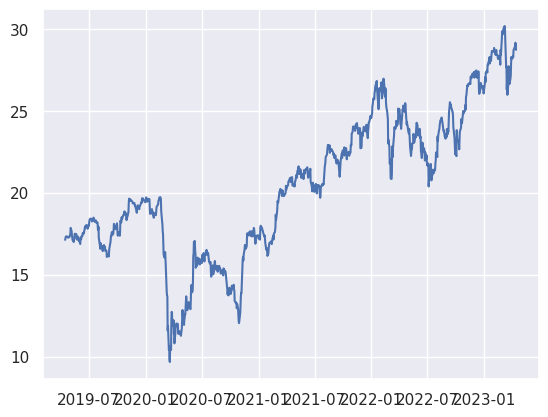
    return datetime.datetime(year=year, month=month, day=day)

  datetime\_object = str\_to\_datetime('2023-04-10')

  datetime\_object

import matplotlib.pyplot as plt

plt.plot(df.index, df['Close'])



Dans la continuité du processus, nous allons nous servir de fenêtre glissante.Pour ce faire nous avons crée la fonction window\_data :

Cette fonction prend en entrée un DataFrame data et un paramètre n (par défaut 3). Elle crée une fenêtre glissante de taille n à partir des données. La fonction boucle sur les valeurs de i de n à 1 et ajoute une colonne au DataFrame windowed\_data pour chaque décalage vers le haut des valeurs de la colonne 'Close' de data. Enfin, la fonction ajoute également une colonne 'Target' correspondant aux valeurs d'origine de la colonne 'Close'. Le DataFrame résultant est renvoyé après avoir supprimé les lignes contenant des valeurs manquantes (dropna).

def window\_data(data, n=3):

    windowed\_data = pd.DataFrame()

    for i in range(n, 0, -1):

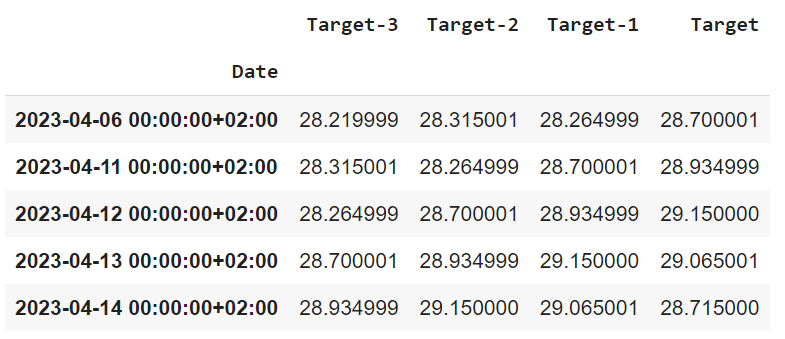
        windowed\_data[f'Target-{i}'] = data['Close'].shift(i)

    windowed\_data['Target'] = data['Close']

    return windowed\_data.dropna()

windowed\_df=window\_data(df)

windowed\_df.tail()



Afin de mettre à l’échelle les données la fonction MinMaxScaler a été implémenté :

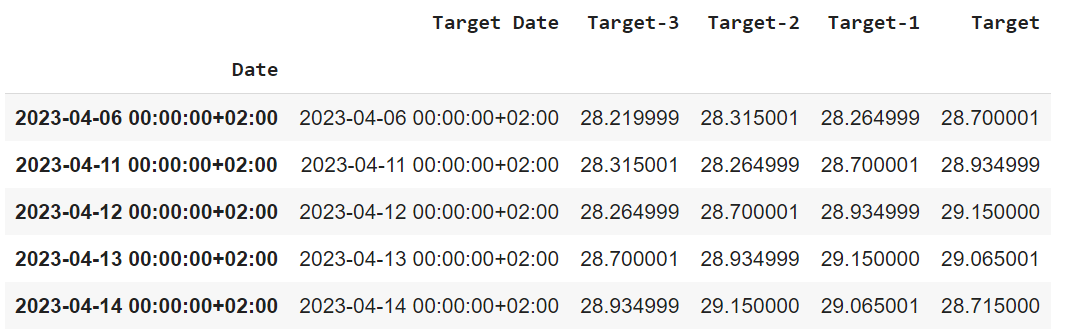
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature\_range=(1,4))

training\_set\_scaled = sc.fit\_transform(windowed\_df)

windowed\_df.insert(0,'Target Date', windowed\_df.index)

windowed\_df.tail()



Une fonction **windowed\_df\_to\_date\_X\_y** a été implémentée afin d’effectuer plusieurs opérations pour préparer les données pour l'entraînement d'un modèle de prédiction.

def windowed\_df\_to\_date\_X\_y(windowed\_dataframe):

  df\_as\_np = windowed\_dataframe.to\_numpy()

  dates = df\_as\_np[:, 0]

  middle\_matrix = df\_as\_np[:, 1:-1]

  X = middle\_matrix.reshape((len(dates), middle\_matrix.shape[1], 1))

Y = df\_as\_np[:, -1]

  return dates, X.astype(np.float32), Y.astype(np.float32)

dates, X, y = windowed\_df\_to\_date\_X\_y(windowed\_df)

dates.shape, X.shape, y.shape

la première partie du code extrait la première colonne du tableau df\_as\_np, qui correspond aux dates, en utilisant l'indexing NumPy [:, 0]. Les valeurs des dates sont assignées à la variable dates.

une sous-matrice du tableau df\_as\_np, en excluant la première colonne (dates) et la dernière colonne. La sous-matrice résultante est assignée à la variable middle\_matrix. Ensuite, la méthode reshape est utilisée pour remodeler middle\_matrix en un tableau tridimensionnel de dimensions (len(dates), middle\_matrix.shape[1], 1). La dimension supplémentaire de taille 1 est ajoutée pour être compatible avec le modèle LSTM qui attend des entrées de forme (batch\_size, time\_steps, features). Le résultat est assigné à la variable X. le code : Y = df\_as\_np[:, -1] nous permet de préparer les valeurs cibles.

Il s’ensuit une étape de découpage de la data : la division des données préparées en ensembles d'entraînement, de validation et de test

q\_80 = int(len(dates) \* .8)

q\_90 = int(len(dates) \* .9)

dates\_train, X\_train, y\_train = dates[:q\_80], X[:q\_80], y[:q\_80]

dates\_val, X\_val, y\_val = dates[q\_80:q\_90], X[q\_80:q\_90], y[q\_80:q\_90]

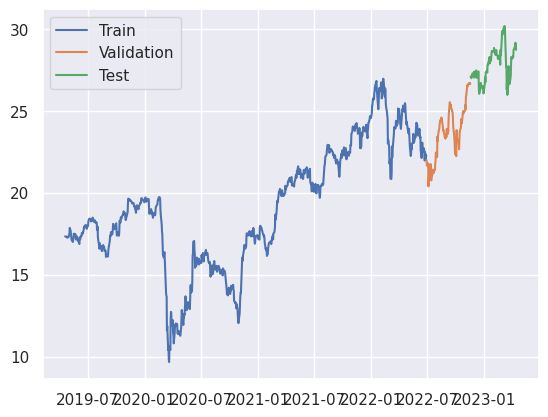
dates\_test, X\_test, y\_test = dates[q\_90:], X[q\_90:], y[q\_90:]

plt.plot(dates\_train, y\_train)

plt.plot(dates\_val, y\_val)

plt.plot(dates\_test, y\_test)

plt.legend(['Train', 'Validation', 'Test'])



La compilation et l’entrainement du modèle sera opérée via le code suivant :

En résumé, cette partie du code définit, compile et entraîne un modèle de réseau de neurones récurrents (LSTM) à l'aide de TensorFlow. Le modèle est configuré avec des couches LSTM et denses, et il est entraîné avec des données d'entrée X\_train et des valeurs cibles y\_train. L'optimiseur Adam est utilisé pour minimiser la fonction de perte (MSE) et les métriques d'erreur absolue moyenne sont calculées pendant l'entraînement.

from tensorflow.keras.models import Sequential

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import layers

model = Sequential([layers.Input((3, 1)),

                    layers.LSTM(64),

                    layers.Dense(32, activation='relu'),

                    layers.Dense(32, activation='relu'),

                    layers.Dense(1)])

model.compile(loss='mse',

              optimizer=Adam(learning\_rate=0.001),

              metrics=['mean\_absolute\_error'])

model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=100)

**Définition du modèle :**

model = Sequential([layers.Input((3, 1)),

                    layers.LSTM(64),

                    layers.Dense(32, activation='relu'),

                    layers.Dense(32, activation='relu'),

                    layers.Dense(1)])

Ces lignes de code définissent le modèle séquentiel en utilisant la classe Sequential de TensorFlow. Le modèle commence par une couche d'entrée (Input) spécifiant la forme des données d'entrée, qui est (3, 1) dans ce cas, indiquant que chaque exemple d'entrée est une séquence de longueur 3 avec une dimension d'attribut de 1. Ensuite, il y a une couche LSTM avec 64 unités pour capturer les motifs temporels. Suivent deux couches denses (Dense) avec 32 unités et une fonction d'activation ReLU. Enfin, il y a une couche de sortie dense avec 1 unité pour prédire une valeur continue.

**Compilation du modèle :**

model.compile(loss='mse',

              optimizer=Adam(learning\_rate=0.001),

              metrics=['mean\_absolute\_error'])

Cette partie compile e modèle en spécifiant la fonction de perte (loss), l'optimiseur (optimizer) et les métriques à évaluer pendant l'entraînement (metrics). Dans ce cas, la perte est définie comme l'erreur quadratique moyenne (mse), l'optimiseur est Adam avec un taux d'apprentissage de 0,001, et la métrique est l'erreur absolue moyenne (mean\_absolute\_error).

**Entraînement du modèle :**

model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=100)

Cette partie entraîne le modèle en utilisant les données d'entraînement X\_train et les valeurs cibles y\_train. Les données de validation sont fournies à l'aide de l'argument validation\_data, qui utilise les ensembles X\_val et y\_val. L'entraînement se fait pendant 100 époques.

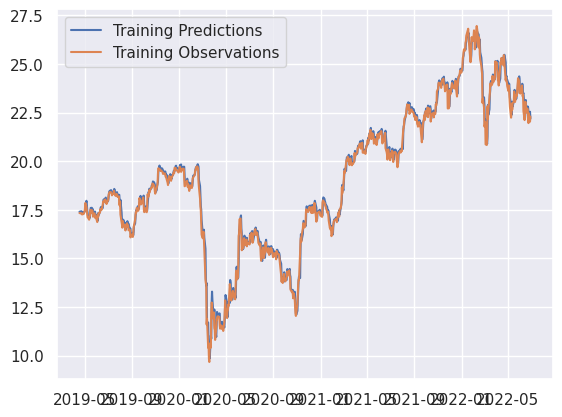
**Prédictions sur l'ensemble d'entraînement** :

val\_predictions = model.predict(X\_val).flatten()

plt.plot(dates\_val, val\_predictions)

plt.plot(dates\_val, y\_val)

plt.legend(['Validation Predictions', 'Validation Observations'])



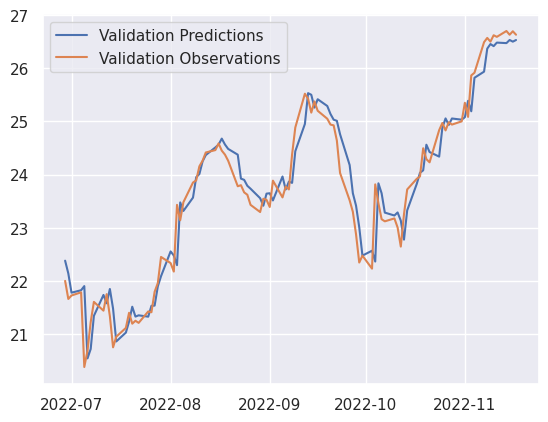
**Prédictions sur l'ensemble de validation** :

val\_predictions = model.predict(X\_val).flatten()

plt.plot(dates\_val, val\_predictions)

plt.plot(dates\_val, y\_val)

plt.legend(['Validation Predictions', 'Validation Observations'])



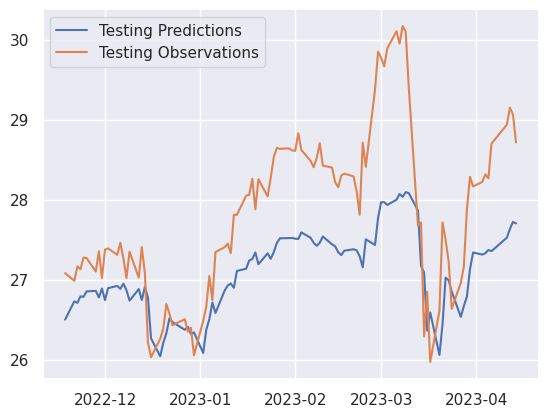
**Prédictions sur l'ensemble de test** :

test\_predictions = model.predict(X\_test).flatten()

plt.plot(dates\_test, test\_predictions)

plt.plot(dates\_test, y\_test)

plt.legend(['Testing Predictions', 'Testing Observations'])



plt.plot(dates\_train, train\_predictions)

plt.plot(dates\_train, y\_train)

plt.plot(dates\_val, val\_predictions)

plt.plot(dates\_val, y\_val)

plt.plot(dates\_test, test\_predictions)

plt.plot(dates\_test, y\_test)

plt.legend(['Training Predictions',

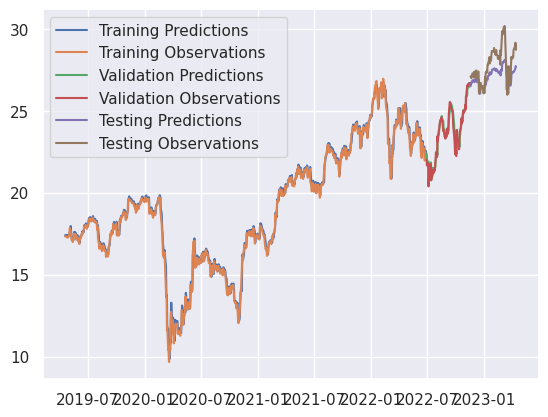
            'Training Observations',

            'Validation Predictions',

            'Validation Observations',

            'Testing Predictions',

            'Testing Observations'])



**Conclusion:**

L’analyse opérée tout au long du projet à travers les différents modèles nous as permis d’explorer deux solutions optimales moyennant la stratégie d’investissement voulue.

Dans l’optique d’un investissement long terme, le modèle le plus adaptée et qui très utilisé en finance de marché, c’est le modèle CAPM.

A contrario, dans une vision plus trading et donc avec un volume de transactions journalier conséquent, le modèle LSTM donne des résultats et surtout un trend très proche de la réalité.