Une image contenant logo

Description générée automatiquement

LICENCE D’ÉCONOMIE APPLIQUEE, PARCOURS ÉCONOMIE ET INGENIERIE FINANCIERES

**Commentaires Projet Python**

A L’ATTENTION DE Mr. Faure

PARIS, LE 9 AVRIL 2023

Nous déclarons sur l'honneur que ce mémoire a été écrit de notre main, sans aide extérieure non autorisée, qu'il n'a pas été présenté auparavant pour évaluation et qu'il n'a jamais été publié, dans sa totalité́ ou en partie. Toutes parties, groupes de mots ou idées, aussi limités soient-ils, y compris des tableaux, graphiques, cartes etc. qui sont empruntés ou qui font référence à d'autres sources bibliographiques sont présentes comme tels, sans exception aucune.

Arno CORMERAIS   
Théo BENICOURT

**I- Frontière Efficiente**

1. ***Récupération des données du fichier Excel (0,5 pt)***

***Ne pas changer le nom du fichier Excel et ne pas mettre le chemin d’accès dans votre code.***

Nous avons privilégié l’utilisation d’un input car cela rend le code plus flexible. L’utilisateur n’a qu’à retrouver l’emplacement du fichier sur son ordinateur afin de récupérer le chemin d’accès et le rentrer dans la console *Spyder*. Nous paramétrons ensuite le répertoire de travail par rapport au chemin d’accès de l’utilisateur.

Nous procédons ensuite à l’extraction des données grâce à la fonction *read\_excel* de la bibliothèque *pandas*, importée au préalable. Nous obtenons alors un *DataFrame* que nous traitons par soucis de clarté, en retirant toutes les lignes contenant la mention ‘NaN’. Nous retirons de plus la colonne contenant les dates afin de travailler uniquement sur les cours. Nous obtenons alors le *DataFrame* ‘*df\_2’* suivant :

Une image contenant texte, écran, Appareils électroniques, affichage

Description générée automatiquement

1. ***VL (1 pt)***

***Faire graphique avec la VL de chaque actif.***

La valeur liquidative se calcule généralement pour des fonds d’investissement, pour un titre elle correspond simplement à son cours. Pour le graphique, nous utilisons la fonction *plot* de la bibliothèque *matplotlib*, sur le *dataframe* composé des cours de chaque actif. Il en résulte le graphique suivant :

Une image contenant graphique

Description générée automatiquement

1. ***Métriques (2 pts)***

***Pour chaque actif vous devez calculer les rendements sur la période, les volatilités et les maxdrawdowns.***

Afin d’obtenir les *returns* de chaque actif, nous utilisons la fonction *pct\_change()* provenant de la bibliothèque *pandas*. Cette fonction calcule de manière générale le taux de variation entre les éléments d’un *DataFrame*. Étant donné que nous travaillons sur des cours, cela revient à calculer les rendements pour chaque actif. De la même manière nous obtenons la volatilité avec la fonction *std().*

Pour les *max drawdowns*, on calcule d’abord les rendements cumulés avec la fonction *cumprod()* de la bibliothèque *pandas* afin de déterminer ensuite les maximum des rendements cumulés. On applique ensuite la formule suivante pour obtenir les *drawdowns* :

Une image contenant texte

Description générée automatiquement

Afin d’obtenir les max *drawdowns*, nous appliquons la fonction *max()* sur le *DataFrame* des *drawdowns* puis multiplions par 100 pour obtenir les résultats en pourcentage. Nous obtenons le résultat suivant :



***4) Portefeuilles efficients (4 pts)***

***Il faut trouver le portefeuille avec la variance minimale, avec la performance maximale et avec le meilleur ratio de Sharpe. La somme des poids de chaque portefeuille doit être égale à 100% et chaque actif doit avoir une allocation comprise entre 0 et 100% sauf le S&P 500 qui doit avoir un poids compris entre 0 et 90%.***

Afin de créer le portefeuille efficient, il faut en premier lieu calculer la matrice de covariances ainsi que les retours annualisés.

La matrice de covariances se calcule simplement avec cette formule :

*cov\_matrix = df.pct\_change().apply(lambda x: np.log(1+x)).cov()*

Une image contenant table, calendrier

Description générée automatiquement

Les rendements annualisés sont plus difficiles à obtenir. Tout d’abord, on les calcule de manière journalière puis on les annualise à l’aide de la commande *resample(‘Y’)*.

Afin de les annualiser, on doit d’abord changer le *DataFrame* initial (*df*) pour placer la date en index. On utilise les commandes :

*df['Date'] = pd.to\_datetime(df['Date'])*

*df.set\_index('Date', inplace=True)*

*Une image contenant table

Description générée automatiquement*

Une image contenant texte, tableau

Description générée automatiquement

Le code pour ensuite annualiser les rendements journaliers est le suivant :

ind\_er = df.resample('Y').last().pct\_change().mean()

Une image contenant texte, capture d’écran, plein air

Description générée automatiquement

Puis, il nous faut créer 3 listes pour enregistrer les rendements, la variance ainsi que les poids :

On calcule ensuite la taille de notre *DataFrame* et on définit le nombre de portfolios à créer.

Ensuite, nous créons une boucle qui va tourner pour créer chaque portfolio.

Dans celle-ci, on retrouve :

* La fonction random qui nous donne des poids d’actifs aléatoire ainsi que deux contraintes. La première permet aux poids de se sommer à 1 et une seconde qui empêche l’actif S&P 500 de dépasser 90% du poids dans le portfolio.
* Une variable « return » qui prend la valeur du (poids\*rendement annualisé)
* Une variable « var » qui calcule la variance du portfolio à l’aide de la matrice de covariance et les poids.
* Une variable « sd » qui est la racine carrée de la variance
* Une variable « ann\_sd » qui se définit comme la (std\*(racine carrée de 250)) si on suppose 250 jours ouvrés par an sur les marchés : l’écart type annualisé
* La méthode append pour ajouter les poids, la variance et le rendement dans les listes.

Une image contenant texte

Description générée automatiquement

Nous fabriquons un nouveau *DataFrame* nommé portfolios pour analyser ces valeurs.



On construit ensuite la frontière efficiente avec la fonction plot :

Une image contenant graphique

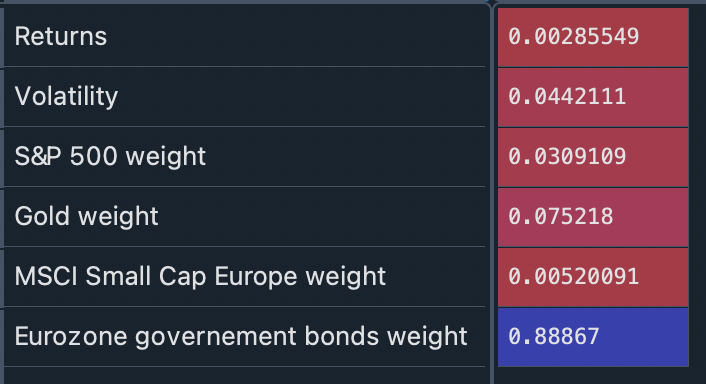
Description générée automatiquement

Maintenant, il est question de créer les *portfolios* le moins risqué, le plus performant et le plus optimale.

Pour cela, on va :

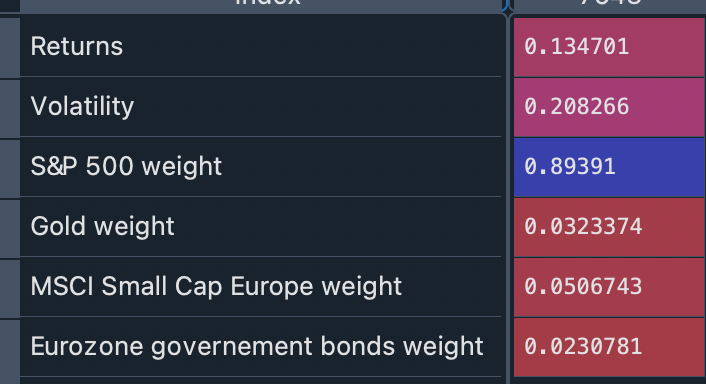
* Minimiser la volatilité sans se soucier du rendement (le moins risqué) :

On obtient une volatilité très faible à moins de 5%.

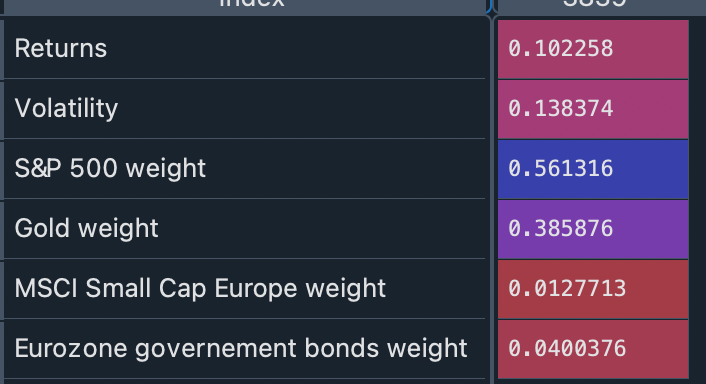


* Maximiser les rendements sans se soucier de la volatilité (le plus performant) :

On obtient un rendement plutôt élevé à plus de 13%.



* Le plus optimal est celui qui possède le meilleur ratio de Sharpe. Il permet de mesurer la rentabilité d’un portefeuille en fonction du risque pris :



On met en image ces résultats à travers les étoiles sur le graphique (verte : optimal, turquoise : performant, rouge : moins risqué) :

***Une image contenant graphique

Description générée automatiquement***

**II-Backtest**

1. ***Bollinger contrariante LONG ONLY avec le premier asset ou autre stratégie (4 pts)***

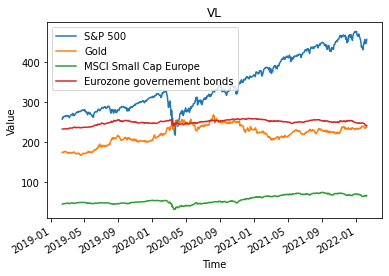
***Il faut calculer la moyenne mobile sur 5 jours ainsi que sa volatilité sur 5 jours. Si le prix est en-dessous de cette moyenne mobile moins cette volatilité alors nous ne sommes pas investis dans cet actif le jour suivant, sinon oui. Faire également pour 10, 15, 20 et 30 jours. Montrer les résultats des VL de vos stratégies sur un graphique puis commenter vos résultats.***

Dans cette question, il est nécessaire de différencier les 5 horizons temporels afin de pouvoir les comparer. Nous créons alors une boucle qui va ajouter plusieurs colonnes dans le *DataFrame* et ce, pour chaque fenêtre mobile :

* La moyenne mobile
* La volatilité
* La condition respectée ou non (Price < MM – Vol)
* Si l’on est investi ou non le jour suivant

Les deux dernières colonnes sont composées de Boolean, nous modifions cela à l’aide de la fonction *astype(int)* qui transforme la valeur « TRUE » en 1 et la valeur « FALSE » en 0. Autrement dit, si le cours de l’actif considéré (ici S&P 500) est supérieur à MM – Vol, alors nous ne serons pas investis le lendemain. La valeur affichée par le *DataFrame* sera 0. La valeur liquidative est ensuite calculée avec la fonction *cumprod().*

Nous représentons nos résultats sur le graphique suivant :



Nous remarquons que plus la fenêtre mobile est courte, plus la valeur liquidative est élevée. En effet, les variations de prix se font plus ressentir lorsque la moyenne mobile et la volatilité sont calculées sur des fenêtres plus courtes, ce qui entraîne une sensibilité plus importante de la stratégie à ces variations.