Projet PI²

Réplication d'indice par la cointégration

Rapport d'avancement

Méthodes Python et MySQL

Table des matières

I. Base de données MySQL

Afin de fluidifier les extractions de données depuis une structure de données, une liste à plusieurs dimensions à d'abord été envisagée mais aux vues du temps nécessaire à la compilation et à l'exécution du code, une autre structure de données a été choisie.

Finalement, nous avons opté pour une **base de données sous MySQL en local**. Le code Python associé a été développé au sein du fichier « *Datas-to-SQL.py* ». Une fois les datas importées, ce fichier ne sera plus utilisé et l'exportation/importation de la database sera privilégiée.

1. Connexion Python & MySQL

Afin de permettre l'ouverture et l'utilisation d'un flux entre Python et MySQL, la librairie « *mysql.connector* » a été utilisée.

Au sein de ce projet, cette **base de données** a été nommée « ***allDatas*** » et contient **2 tables** « ***Composition*** » et « ***Datas*** » correspondant aux fichiers CSV fournis.

La **database** étant **en local** sur l'ordinateur de chaque utilisateur, elle a été créée sur 1 ordinateur puis exportée aux autres membres pour qu'ils puissent l'importer directement.

2. Extraction et ajout des données CSV dans MySQL

Les méthodes permettant l'ajout de données dans les tables sont « *Insert_Datas_In_Composition()* » pour la table Composition et « *Insert_Datas_In_Datas()* » pour la table Datas.

Les données sont d'abords extraites des fichiers CSV et stockées dans une liste sous Python pour être ensuite utilisée dans les méthodes d'ajout aux tables. Afin de s'assurer et d'homogénéiser le format des dates, l'utilisation de la librairie *datetime* a été nécessaire (transformation en objet date puis en string).

Une remarque intéressante pourrait être l'utilisation de **tuple** comme type de variable pour l'utilisation de **paramètres via les requêtes MySQL** sous Python (une liste n'est pas acceptée par la méthode de la librairie mais un tuple). Il est également nécessaire d'utiliser la **méthode « *commit()* » pour enregistrer** ces ajouts(ou changements) dans la base de donnée.

II. Régression linéaire

Les méthodes liées à la **régression linéaire** sont développées au sein du fichier « *MySQL-to-Python.py* ». Il sera peut-être nécessaire d'importer plusieurs librairies (ces dernières sont précisées en début de code dans la section *Librairies*).

L'objectif de cette régression est de définir un modèle pour prédire la valeur de l'indice en utilisant le log(Price) des stocks le composant. Les prix utilisés sont les **prix de clôture** des stocks.

Lors des requêtes MySQL, les résultats obtenus sont stockés sous forme de **dataframes Pandas**.

1. Création de la matrice des données

Le modèle de régression linéaire nécessite **4 sets de données** :

- 2 d'entrainement : X_train et Y_train
- 2 de test : X_test et Y_test
  Les **X** sont les **inputs** du modèle et les **Y** les **output** attendus (**True values**).

La matrice de données est sous la forme suivante :

Dates	*Valeur de l'indice (Y)*	*Stock_1 (X1) - log(Price)*	*…*	*Stock_500 (X500) - log(Price)*
Date_1	Y_1	X1_1	…	X500_1
…	…	…	…	…
Date_T	Y_T	X1_T	…	X500_T

Le modèle sera développé à partir des valeurs des sets d'entrainement et sera ensuite testée pour vérification sur les sets de test.

Afin de réaliser ce modèle plusieurs étapes sont nécessaires :

- **Définir la composition de l'indice pour une date donnée (i.e. *date_T*) :**
  Cela est fait au sein de la méthode « *Composition_Indice_Date_T()* » qui nécessite la connexion avec MySQL et la date à laquelle nous faisons la recherche.

- **Recréer l'indice :**
  Cela est fait dans la méthode « *Recreation_Indice ()* ». Cette dernière nous renvoie le benchmark complet à l'aide de la requête MySQL suivante :

  « *SELECT trade_Date, AVG(log(close_Value)) FROM datas GROUP BY datas.trade_Date;* »

  Pour chaque jour, nous prenons la moyenne des log(Price) des stocks.

- **L'extraction et la construction de la matrice des log(Price)** se fait via les méthodes « *Extract_LogClosePrice_Stocks_Btw_2_Dates* » et « *Create_Df_ClosePrice ()* ».

Comme paramètres des méthodes il peut notamment y avoir la date de fin qui est date_T et la fenêtre de jours qui nous intéresse.
Rmq : Cette fenêtre comporte les jours fériés et weekend mais durant ces derniers les marchés sont fermés. Par exemple, pour une fenêtre de 14 jours commençant un lundi, seules les données de 10 jours sont renvoyées car il y a 2 weekends.

Dans cette matrice, **certains stocks ne sont pas présents dans la composition de l'indice sur toute la fenêtre**. Il a été décidé de **supprimer les stocks** se trouvant dans cette situation. Ainsi sur une moyenne de 500 stocks et suivant la taille de la fenêtre la quantité d'indice présent au final dans la matrice peut fortement varier (de 500 à 200 pour les très grandes fenêtres).
Nous conseillons de ne pas aller au-delà d'une fenêtre de 100 jours.

- **La séparation de cette matrice en 2 types de sets : Train et Test :**
  Cette séparation se fait via la méthode « *Split_Df_Train_Test ()*» qui va séparer en 4 sets (2 pour X et 2 pour Y) suivant le pourcentage de répartition désiré**. Ce pourcentage est celui de la composition du set de Test.**
  Nous conseillons de ne pas dépasser les 40% pour le set de Test.

Une fois ces 4 sets créés, nous pourrons les utiliser pour définir et entrainer notre model puis pour le tester.
Pour rappel, les 4 sets sont composés de :

- 2 sets d'entrainement : X_train et Y_train
- 2 sets de test : X_test et Y_test

2. Fit model

L'entrainement du modèle se fait via la méthode « *Fit_Model ()* » qui définit un **modèle de régression linaire**. Les datas X_train et Y_train sont utilisées pour cela.

Une fois le modèle défini, ce dernier est testé en faisant des prédictions et en comparant ces prédictions aux valeurs connues de l'indice.

Les **prédictions** sont donc définies à l'aide de la méthode « *model.predict ()* » qui prend en paramètre les inputs (ici X_test). Cette méthode retourne les prédictions sous forme d'une liste de dimension 1. Enfin, les **prédictions sont comparées aux True outputs** (Y_test) au sein de la méthode « *Print_Results ()* ». Cette dernière affiche les résultats, les erreurs (via l'utilisation de la MSE, R^2-error et MAE), les p-values et les coefficients de chaque input.

III. Test ADF

La méthode effectuant le **test ADF** est également développée dans le fichier « *MySQL-to-Python.py* ». Nous aurons certainement également la nécessité d'importer des librairies.

L'objectif de cette partie est de savoir si le **portefeuille** est **co-intégrable** ou non en testant la **stationnarité** d'une série temporelle.

1. Création de la liste des résidus

Tout d'abord nous récupérons en paramètre de la méthode « *ADF()* » le set de test Y_test mais également les prédictions effectuées lors de la régression linéaire.

En premier lieu, nous allons créer la **liste des résidus** en soustrayant à chaque valeur « *Y_test['AVG(log(close_Value))']* » les prédictions associées, autrement dit nous **retirons la valeur des prédictions à chaque moyenne du logarithme de la valeur du stock à la fermeture** du set Y_test.

2. Test ADF sur les résidus

Nous avons alors importé la librairie « *statsmodel.api* », particulièrement la méthode « *adfuller()* » qu'elle contient.

Nous avons appliqué cette méthode à la **liste des résidus** car cette liste est la série temporelle pour laquelle nous nécessitons de savoir si elle est **stationnaire ou non**. Afin de savoir si celle-ci est stationnaire ou non, nous avons alors comparé l'**ADF Statistic avec les Critical Values**, tous deux obtenues avec la méthode « *adfuller()* ». Nous avons 3 seuils pour les Critical Values, correspondants aux seuils de confiance de la co-intégrabilité d'une série temporelle. Nous avons alors deux possibilités quant aux résultats obtenus :

- ADF Statistic > Critical Values : le **portefeuille n'est pas co-intégrable**
- ADF Statistic < Critical Values : le **portefeuille est co-intégrable** avec une certitude correspondant au seuil choisi.

IV. Sélections des stocks dans le rebalancement du portefeuille

L'étude de la corrélation entre les stocks et entre les stocks et l'indice est faite au sein de la méthode « ***Correlation_Matrix(df_stocks, df2)*** » . La méthode prend en input la dataframe contenant les informations sur les stocks (*df_stocks*) et la dataframe contenant les informations sur l'indice (*df2*).

Elle retourne la correlation Stocks Vs Indice (*corr_stocks_indice*) et Stocks Vs Stocks (*corr_stocks_stocks*).

1. Sélection corrélation Stock Vs Stock

Dans l'optimisation du portefeuille afin de se rapprocher au mieux de l'indice à répliquer, il est parfois nécessaire d'affiner la sélection des stocks composant l'indice. C'est pourquoi nous avons décidé de sélectionner ces stocks en fonction de la corrélation entre eux. En effet en calculant la matrice de corrélation de ces stocks, il est possible de savoir quels stocks sont corrélés entre eux. Et afin d'en tirer le plus d'information possible, il est important d'éliminer les stocks qui auraient une corrélation trop élevée (degré alpha) avec les autres. Ces derniers n'apportent pas plus d'information utiles à la réplication de l'indice contrairement à un stock qui auraient une corrélation moindre avec tous les autres stocks. Ainsi dans cette optique de réduire cette information inutile, cette sélection de stock permet à un certain degré alpha de minimiser le nombre de ces stocks et donc d'améliorer la précision de la réplication de l'indice.

2. Sélection corrélation Stock Vs Indice

Comme nous voulons répliquer l'indice, nous allons sélectionner les stocks ayant les plus fortes corrélations positives avec l'indice. Une fois les valeurs des corrélations tirées nous sélectionnons les '*n*' premières pour un panier de '*n*' stocks.

En théorie et sur la période analysée, il s'agit des meilleurs stocks à sélectionner puisque ces derniers suivent au mieux l'indice.

V. Ratios

Nous avons décidé d'implémenter un ratio afin de pour pouvoir mesurer les performances du portefeuille.

1. Information Ratio

Le ratio d'information, également appelé ratio d'évaluation, mesure et compare le rendement actif d'un investissement par rapport à un indice de référence par rapport à la volatilité du rendement actif. Il est défini comme le retour actif divisé par l'erreur de suivi.

$$\text{IR} = \frac{\text{Portfolio Return} - \text{Benchmark Return}}{\text{Tracking Error}}$$

where:

$\text{IR} = \text{Information ratio}$

$\text{Portfolio Return} = \text{Portfolio return for period}$

$\text{Benchmark Return} = \text{Return on fund used as benchmark}$

$\text{Tracking Error} = \text{Standard deviation of difference between portfolio and benchmark returns}$

Il est utilisé comme une mesure du niveau de compétence d'un gestionnaire de portefeuille et de sa capacité à générer des bénéfices par rapport à un indice de référence, mais il tente également d'identifier la cohérence de la performance en incorporant une erreur de suivi, ou un écart type, dans le calcul.

2. Sharpe ratio

Le ratio de Sharpe mesure l'écart de rentabilité d'un portefeuille d'actifs financiers par rapport au taux de rendement d'un placement sans risque, divisé par un indicateur de risque, l'écart type de la rentabilité de ce portefeuille, autrement dit sa volatilité.

Formula and Calculation of Sharpe Ratio

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

where:

R_p = return of portfolio

R_f = risk-free rate

σ_p = standard deviation of the portfolio's excess return

Il est utilisé ici pour comparer la performance du portefeuille vis-à-vis du marché.

VI. Rebalancement

1. Méthodes utilisées pour le rebalancement

Au total, 2 méthodes sont utilisées pour faire le rebalancement :

- **Rebalancement_UnCycle() :**
  Il s'agit de la méthode permettant de faire le rebalancement du portefeuille sur 1 cycle de rebalancement. Elle permet de relever les rendements du portefeuille et de l'indice sur la période qui vient de se terminer. Elle refait les calculs pour déterminer le nouveau portefeuille sur la nouvelle période.

- **Rebalancement() :**
  Il s'agit de la méthode générale appelant ***Rebalancement_UnCycle*()** qui permet d'éffectuer tous les rebalancement sur la plage de temps saisie.
  Elle retourne la composition du portefeuille (comparable à un historique) sur toute la période contenant tous les rebalancements. Elle retourne également les rendements du portefeuille et de l'indice sur toute la période des rebalancements.

2. Plots des rendements et de la réplication

Une fois les rendements obtenus, la méthode « ***Rendement_Percent(df)*** » est appelée pour calculer l'évolution des rendements et pour les comparer avec d'autres mais surtout pour pouvoir les afficher via un plot.

Le calcul effectuer pour calculer l'évolution du rendement est (avec $r_{\%}$ le rendement en pourcentage):

$$\begin{cases} V_0 = 100 \\ V_n = V_{n-1} * (1 + \frac{r_{\%}}{100}) \end{cases}$$

Pour ploter les rendements la méthode « ***Plot_Indice_Stocks()*** » est utilisée.
Elle prend en input une liste de dataframe à ploter ainsi qu'une liste pour les labels de chaque dataframe et le titre de la figure.

Il s'agit d'une méthode très flexible pouvant afficher plusieurs courbes sur la même figure.

VII. Long Short Strategy

Après avoir construit la stratégie de suivi simple, une extension naturelle pour exploiter le potentiel de suivi des portefeuilles cointégrés serait de répliquer des indices de référence "plus" et "moins".

Les benchmarks « plus » et « minus » peuvent être construits en ajoutant ou en soustrayant aux rendements du benchmark un rendement excédentaire annuel.

La stratégie est tout simplement être short sur le portefeuille qui suit l'indice « minus » et long sur le portefeuille qui suit l'indice de référence plus.

- Les caractéristiques d'une stratégie long et short sont généralement reconnues comme étant une évolution régulière des rendements, une faible volatilité et la neutralité du marché.
- Le fait que les stratégies long-short actions assurent une utilisation plus efficace de l'information que les stratégies long only résulte du fait que les pondérations des actifs sous-évalués ne sont pas limitées à zéro.

Les résultats des stratégies long-short dépendent fortement de la méthode de sélection des titres utilisée et de l'écart entre les indices de référence "plus" et "moins" suivis.

1. Les étapes à suivre :

1. ***Test the cointegration :***

The new cointegration regressions can be written as:

$$\log(\text{index\_plus}_t) = a_1 + \sum_{k=1}^{n} a_{k-1} * \log(P_{k,t}) + \varepsilon_t \quad (7)$$

$$\log(\text{index\_minus}_t) = b_1 + \sum_{k=1}^{n} b_{k-1} * \log(P_{k,t}) + \varepsilon_t \quad (8)$$

We note that the stock weights are not restricted to be positive in the tracking portfolios above; in fact it is likely that we shall take some short positions in the portfolios tracking both 'plus' and 'minus' benchmarks.

Les portefeuilles restent assez cointégrés avec les indices de référence suivis, même si ces derniers divergent sensiblement de

l'indice réel du marché.

2. ***Corrélation des rendements des portefeuilles de suivi avec les rendements du marché :***

In order to examine the correlation of the long-short strategy returns with the market returns, we can write the returns on the 'plus' and 'minus' portfolios separately as follows:

$$R_{+,t} = \alpha_+ + \rho_+ \frac{\sigma_+}{\sigma_{B+}} R_{B+,t} + \varepsilon_{+,t} \quad (10)$$

$$R_{-,t} = \alpha_- + \rho_- \frac{\sigma_-}{\sigma_{B-}} R_{B-,t} + \varepsilon_{-,t} \quad (11)$$

where $\rho_{+/-}$ are the correlation coefficients between the 'plus' and respectively 'minus' portfolio returns with the 'plus'/'minus' benchmark returns, σ_+ / σ_{B+} and σ_- / σ_{B-} are the relative volatilities of the 'plus' respectively 'minus' portfolio returns to the 'plus'/'minus' benchmark returns, and ε_+ and ε_- are the tracking errors of the 'plus'/'minus' portfolios.

La méthode de sélection des titres et l'écart entre les indices de référence suivis influent sur le niveau de corrélation, mais la relation n'est pas directe. En outre, on observe une légère diminution des corrélations à mesure que le nombre d'actions dans les portefeuilles de suivi augmente.

3. ***Calcul du ratio de Sharpe et ratio d'information***

Les ratios de Sharpe produits par le reclassement annuel et le reclassement basé sur la fréquence sont considérablement plus élevés que ceux affichés par les stratégies de reclassement quotidien.

En résumé :

Les résultats obtenus par le back-testing prouvent que, lors de la mise en place d'une stratégie de réplication et de long-short market neutral basée sur la cointégration, les paramètres suivants ont un impact significatif sur le succès de la stratégie :

- Méthode de sélection des titres - en termes de rendement, la méthode de sélection des titres est déterminante pour le succès de la stratégie long-short.

- Les indices de référence à suivre par les portefeuilles "plus"/"moins" - comme le montrent les résultats du back-test, l'écart entre les indices de référence suivis dans la stratégie long-short ne peut être augmenté sans une augmentation correspondante de la volatilité

- Nombre d'actions dans chaque portefeuille - comme le montrent les tests de stationnarité des résidus, afin d'identifier une relation de cointégration, un nombre minimum d'actions est requis dans le portefeuille de suivi.

- Période de calibrage - comme le montrent les tests de cointégration et comme l'implique la théorie, un nombre minimum d'années est nécessaire pour construire une relation de cointégration.