Eléments De réponses

Sghuri Chaimaa

FinaMaze™

Introduction

- Ces slides accompagnent le code réponse pour le test Technique qui concerne :Traitement d'une base de donnée 'Market_data', il présente un document explicatif du code et des réponses

- Cette base de donnée expose les données financières de 6 entreprises :MSFT, IBM, AAPL, FB, GOOG et NDX

- Pour chacune des entreprises citées ci-haut, on peut tirer les informations suivantes :
  - Prix de l'action (stock price)
  - Id (de 0 à 1646)
  - Date

TABLE OF CONTENTS

Step 01

01
- Importer les données et
- Organiser ces données sous la forme souhaite afin de faciliter l'extraction

02
- Definir une fonction qui retourne les prix des stocks spécifies sinon par defaut tous les stocks , pendant une periode determinée

03
- Analyse des données manquantes,
- Gérer les Datamissing

04
- Visualisation des données

05
Calculer les daily Returns

06
Calculer la moyenne et l'ecart type de ces daily returns

01

Importer les données et Organiser ces données sous la forme souhaitée afin de faciliter l'extraction

Importer les données:

- On a la base de donnée est sous forme un fichier json , Pour Faciliter l'import des données , On va utiliser la librairie Pandas .
- On a une dataframe dont les données ont la forme d'un dictionnaire d'ou la nécessité de reorganiser cette dataframe afin de faciliter l'extraction
- Data Initiale :

```
Entrée [28]: market_data=pd.read_json('market_data.json')
             market_data
```

Out[28]:

	MSFT	IBM	AAPL	FB	GOOG	NDX
0	{'_id': 0, 'data': {'Date': '2012-05-18T00:00:...	{'_id': 0, 'data': {'Date': '2012-05-18T00:00:...	{'_id': 0, 'data': {'Date': '2012-05-18T00:00:...	{'_id': 0, 'data': {'Date': '2012-05-18T00:00:...	{'_id': 0, 'data': {'Date': '2012-05-18T00:00:...	{'_id': 0, 'data': {'Date': '2012-05-18T00:00:...
1	{'_id': 1, 'data': {'Date': '2012-05-21T00:00:...	{'_id': 1, 'data': {'Date': '2012-05-21T00:00:...	{'_id': 1, 'data': {'Date': '2012-05-21T00:00:...	{'_id': 1, 'data': {'Date': '2012-05-21T00:00:...	{'_id': 1, 'data': {'Date': '2012-05-21T00:00:...	{'_id': 1, 'data': {'Date': '2012-05-21T00:00:...
2	{'_id': 2, 'data': {'Date': '2012-05-22T00:00:...	{'_id': 2, 'data': {'Date': '2012-05-22T00:00:...	{'_id': 2, 'data': {'Date': '2012-05-22T00:00:...	{'_id': 2, 'data': {'Date': '2012-05-22T00:00:...	{'_id': 2, 'data': {'Date': '2012-05-22T00:00:...	{'_id': 2, 'data': {'Date': '2012-05-22T00:00:...
3	{'_id': 3, 'data': {'Date': '2012-05-23T00:00:...	{'_id': 3, 'data': {'Date': '2012-05-23T00:00:...	{'_id': 3, 'data': {'Date': '2012-05-23T00:00:...	{'_id': 3, 'data': {'Date': '2012-05-23T00:00:...	{'_id': 3, 'data': {'Date': '2012-05-23T00:00:...	{'_id': 3, 'data': {'Date': '2012-05-23T00:00:...
4	{'_id': 4, 'data': {'Date': '2012-05-24T00:00:...	{'_id': 4, 'data': {'Date': '2012-05-24T00:00:...	{'_id': 4, 'data': {'Date': '2012-05-24T00:00:...	{'_id': 4, 'data': {'Date': '2012-05-24T00:00:...	{'_id': 4, 'data': {'Date': '2012-05-24T00:00:...	{'_id': 4, 'data': {'Date': '2012-05-24T00:00:...
...	...	...	...	...	...	...
1642	{'_id': 1642, 'data': {'Date': '2018-12-19T00:...	{'_id': 1642, 'data': {'Date': '2018-12-19T00:...	{'_id': 1642, 'data': {'Date': '2018-12-19T00:...	{'_id': 1642, 'data': {'Date': '2018-12-19T00:...	{'_id': 1642, 'data': {'Date': '2018-12-19T00:...	{'_id': 1642, 'data': {'Date': '2018-12-19T00:...
1643	{'_id': 1643, 'data': {'Date': '2018-12-24T00:...	{'_id': 1643, 'data': {'Date': '2018-12-24T00:...	{'_id': 1643, 'data': {'Date': '2018-12-24T00:...	{'_id': 1643, 'data': {'Date': '2018-12-24T00:...	{'_id': 1643, 'data': {'Date': '2018-12-24T00:...	{'_id': 1643, 'data': {'Date': '2018-12-24T00:...

Manipulation de la data pour obtenir une DataFrame Claire ,et bien organise :
Ps:On remarque que tous les enregistrements des prix des stocks sont effectuées dans la meme date d'ou le résultat suivant :

Entrée [31]:

```
market_data=pd.read_json('market_data.json')
L=list(market_data.columns)
#On remarque que pour tous les enregistrements de tous les stocks :ont le meme id et le meme stock , donc on procede comme suit
market_data['id']=market_data[L[0]].apply(lambda x:x['_id'])
market_data['Date']=market_data[L[0]].apply(lambda x:x['data']['Date'])
for t in L:
    market_data[t]=market_data[t].apply(lambda x: x['data']['price'])
market_data
```

Out[31]:

	MSFT	IBM	AAPL	FB	GOOG	NDX	id	Date
0	24.327055	146.358612	65.592659	38.230000	299.078979	2478.530029	0	2012-05-18T00:00:00
1	NaN	147.763336	69.414101	34.029999	305.908386	2545.429932	1	2012-05-21T00:00:00
2	24.734301	147.060989	68.881065	31.000000	299.278229	2539.199951	2	2012-05-22T00:00:00
3	24.194075	146.537994	70.561768	32.000000	303.592072	2547.080078	3	2012-05-23T00:00:00
4	24.160828	146.515564	69.913734	33.029999	300.702881	2531.350098	4	2012-05-24T00:00:00
...	...	...	...	...	...	...	...	...
1642	101.617836	108.498039	157.706192	133.240005	1023.010010	6342.970215	1642	2018-12-19T00:00:00
1643	92.248878	100.241623	143.924454	124.059998	976.219971	5899.350098	1643	2018-12-24T00:00:00
1644	99.158005	106.028572	153.059998	134.520004	1043.880005	6288.299805	1644	2018-12-27T00:00:00
1645	98.383781	105.329666	153.138428	133.199997	1037.079956	6285.270020	1645	2018-12-28T00:00:00

02

Définir une fonction qui retourne les prix des stocks spécifies sinon par défaut tous les stocks , pendant une période déterminée

on a défini une fonction sur Python qui permet d'afficher le prix des stocks sur la période comprise entre la date de début et la date de fin, ces dernières étant prises comme arguments au niveau de la fonction suivante:

Entrée [2]:

```
#Step1#Reponse
#on va creer une fonction qui prend comme arguments positionnels datedebut et datefin ,et un argment par defaut , une liste
#qui contient tous les stock , C'est une fonction qui retourne les prix des stock specifiee sinon par defaut tous les stock
#pendant la periode specifiee :
T=['MSFT','IBM','AAPL','FB','GOOG','NDX']
def ExtractDonnee(datedebut,datefin,Stock=T):
    #Organisation de La dataframe
    market_data=pd.read_json('market_data.json')
    L=list(market_data.columns)
    #On remarque que pour tous les enregistrements de tous les stocks :ont le meme id et le meme stock , donc on procede comme su
    market_data['id']=market_data[L[0]].apply(lambda x:x['_id'])
    market_data['Date']=market_data[L[0]].apply(lambda x:x['data']['Date'])
    for t in L:
        market_data[t]=market_data[t].apply(lambda x: x['data']['price'])
    #eT Voila LE RESultat , on obtient une dataframe bien organisee

    market_data.set_index('Date',inplace=True)
    datedebut=datedebut+'T00:00:00'
    datefin=datefin+'T00:00:00'
    N=market_data.copy()
    return N.loc[datedebut:datefin,Stock]
```

Testons cette fonction sur la période comprise du 01-01-2017 au 31-12-2018, notre fonction retourne le résultat suivant :

```
S=ExtractDonnee('2017-01-01','2018-12-31').reset_index()
S
```

Out[4]:

	Date	MSFT	IBM	AAPL	FB	GOOG	NDX
0	2017-01-03T00:00:00	58.969059	143.492233	110.392334	116.860001	786.140015	4911.330078
1	2017-01-04T00:00:00	58.705212	145.268768	110.268791	118.690002	786.900024	4937.200195
2	2017-01-05T00:00:00	58.705212	144.788177	110.829552	120.669998	794.020020	4964.950195
3	2017-01-06T00:00:00	59.214054	145.500534	112.065109	123.410004	806.150024	5007.080078
4	2017-01-09T00:00:00	59.025597	143.886993	113.091560	124.900002	806.650024	5024.899902
...	...	...	...	...	...	...	...
479	2018-12-19T00:00:00	101.617836	108.498039	157.706192	133.240005	1023.010010	6342.970215
480	2018-12-24T00:00:00	92.248878	100.241623	143.924454	124.059998	976.219971	5899.350098
481	2018-12-27T00:00:00	99.158005	106.028572	153.059998	134.520004	1043.880005	6288.299805
482	2018-12-28T00:00:00	98.383781	105.329666	153.138428	133.199997	1037.079956	6285.270020
483	2018-12-31T00:00:00	99.540192	105.926056	154.618546	131.089996	1035.609985	6329.959961

484 rows × 7 columns

Resultat step 1

Entrée [6]:
```
#Resultat step1:
#On appelle la fonction ExctractDonnee en specifiant les stock :
S=ExtractDonnee('2017-01-01','2018-12-31',['GOOG','AAPL']).reset_index()
S
```

Out[6]:

	Date	GOOG	AAPL
0	2017-01-03T00:00:00	786.140015	110.392334
1	2017-01-04T00:00:00	786.900024	110.268791
2	2017-01-05T00:00:00	794.020020	110.829552
3	2017-01-06T00:00:00	806.150024	112.065109
4	2017-01-09T00:00:00	806.650024	113.091560
...	...	...	...
479	2018-12-19T00:00:00	1023.010010	157.706192
480	2018-12-24T00:00:00	976.219971	143.924454
481	2018-12-27T00:00:00	1043.880005	153.059998
482	2018-12-28T00:00:00	1037.079956	153.138428
483	2018-12-31T00:00:00	1035.609985	154.618546

484 rows × 3 columns

03

- Analyse des données manquantes,
- Gérer les Datamissing

A ce stade, nous nous intéressés particulièrement à deux actions uniquement, à savoir AAPL et GOOG.Nous avons utilisé la fonction définie : on spécifie cette information au niveau de l'argument de notre fonction, de la manière suivante :

Entrée [3]:
```
#Resultat step1:
#On appelle la fonction ExctractDonnee en specifiant les stock :
J=ExtractDonnee('2017-01-01','2018-12-31',['GOOG','AAPL']).reset_index()
#step2:
#Premiere Partie
#Pour gerer les Datamissing , je propose deux methode , celui de l'interpolation lineaire ou supprimer les donnees ;
#Methode 1 :Interpolation lineaire
#Mais avant , Checkons s'il y a Missing data  dans notre base de donnee
J.isnull().describe()
```

Out[3]:

	Date	GOOG	AAPL
count	484	484	484
unique	1	1	2
top	False	False	False
freq	484	484	483

En vue de gérer les datamissing dans notre base de données, 2 cas de figure se présentent : - la suppression des enregistrements qui contiennent des datamissings - l'utilisation d'une interpolation linéaire sur la période déterminée J'ai choisi d'utiliser la seconde méthode qui peut être implémentée de la manière suivante

Entrée [4]:
```
#On remarque que stock appl  a seulement 483 valeurs de false , donc y a surement une valeur True , ce qui signifie une valeur
#manquante pour le stock app .Pour gerer cela on va utiliser methode d'interpolation lineaire
J=J.interpolate(method ='linear', limit_direction ='forward')
#Verification
J.isnull().describe()
```

Out[4]:

	Date	GOOG	AAPL
count	484	484	484
unique	1	1	1
top	False	False	False
freq	484	484	484

Entrée [11]:
```
#Deuxieme Partie
#Dans notre portefuille on detient 100 $ de GooG et 100$ de AAPL à la date precise dans l'enonce , donc on detient X1 de Google
#et X2 de Y, determinons X1 et X2 :
X1=100/(J['GOOG'].iloc[0])
X2=100/(J['AAPL'].iloc[0])
X2,X1
```

Out[11]:
```
(0.9058600030519698, 0.12720380356763814)
```

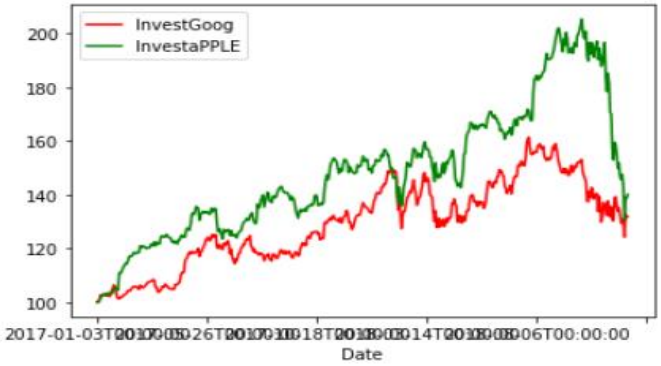
Le graphe qui représente l'évolution des prix des stocks en fonction du temps est donné comme suit :

Entrée [13]:
```
#notre investisment evolue tant quele prix evolue :
J['InvestGoog']=J['GOOG'].apply(lambda x: x*X1)
J['InvestaPPLE']=J['AAPL'].apply(lambda x: x*X2)
J
```

Out[13]:

	Date	GOOG	AAPL	InvestGoog	InvestaPPLE
0	2017-01-03T00:00:00	786.140015	110.392334	100.000000	100.000000
1	2017-01-04T00:00:00	786.900024	110.268791	100.096676	99.888088
2	2017-01-05T00:00:00	794.020020	110.829552	101.002367	100.396058
3	2017-01-06T00:00:00	806.150024	112.065109	102.545349	101.515300
4	2017-01-09T00:00:00	806.650024	113.091560	102.608951	102.445121
...	...	...	...	...	...
479	2018-12-19T00:00:00	1023.010010	157.706192	130.130764	142.859732
480	2018-12-24T00:00:00	976.219971	143.924454	124.178893	130.375406
481	2018-12-27T00:00:00	1043.880005	153.059998	132.785507	138.650930
482	2018-12-28T00:00:00	1037.079956	153.138428	131.920515	138.721977
483	2018-12-31T00:00:00	1035.609985	154.618546	131.733529	140.062756

484 rows × 5 columns



En ce qui concerne le rendement des deux stocks Aapl et Goog, la fonction " pct_change " est utilisée dans ce cas de la manière suivante :

Entrée [24]:

```
#Calcul des rendements quotidiens :
J1=J[['InvestGoog','InvestaPPLE']]
Returns=J1.pct_change()[1:]
Returns['Date']=J['Date']
Returns
```

Out[24]:

	InvestGoog	InvestaPPLE	Date
1	0.000967	-0.001119	2017-01-04T00:00:00
2	0.009048	0.005085	2017-01-05T00:00:00
3	0.015277	0.011148	2017-01-06T00:00:00
4	0.000620	0.009159	2017-01-09T00:00:00
5	-0.002306	0.001009	2017-01-10T00:00:00
...	...	...	...
479	-0.005541	-0.031192	2018-12-19T00:00:00
480	-0.045738	-0.087389	2018-12-24T00:00:00
481	0.069308	0.063475	2018-12-27T00:00:00
482	-0.006514	0.000512	2018-12-28T00:00:00
483	-0.001417	0.009665	2018-12-31T00:00:00

483 rows × 3 columns

Pour la moyenne et l'écart type, ils sont également fournis par les 2 fonctions prédéfinies sur Python :

Entrée [25]:
```
#Calcul moyenne et ecart type des rendements  quotidiens :Returns.mean()
Returns.mean()
```

Out[25]:
```
InvestGoog     0.000677
InvestaPPLE    0.000818
dtype: float64
```

Entrée [26]:
```
Returns.std()
```

Out[26]:
```
InvestGoog     0.014560
InvestaPPLE    0.015458
dtype: float64
```

Entrée [ ]: