Rapport d'analyse de la base de données

I- importation des bibliothèques utiles

Entrée [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
```

II- Importation et traitement de la base de données

Dans cette portie, après avoir importé notre base de données,nous allons lui appliquer quelques traitements afin de la rendre utilisable pour notre étude. Les opérations que nous allons effectuer sont: 1-changer l'indexation de la base 2-changer le type de l'index en datetime 3-classer les date par période 4-selectionner la période sur laquelle on veut travailler 5-éliminer les titres qui ne contiennent pas de valeurs sur cette période

Entrée [2]:

```
def get_return(data,start,end):
    d=pd.read_excel(data,header=0,index_col=0,parse_dates=True,na_values=" ")
    d.index=pd.to_datetime(d.index,format="%Y%m%d")
    d.index=d.index.to_period("D")
    d=d[start:end]
    d=d.dropna(axis=1)
    return d
```

importation de notre base de données

Entrée [3]:

```
dataset=get_return("data.xlsx","2015","2019")
```

Entrée [4]:

```
#statistique descriptive de la base de données
dataset.describe()
```

Out[4]:

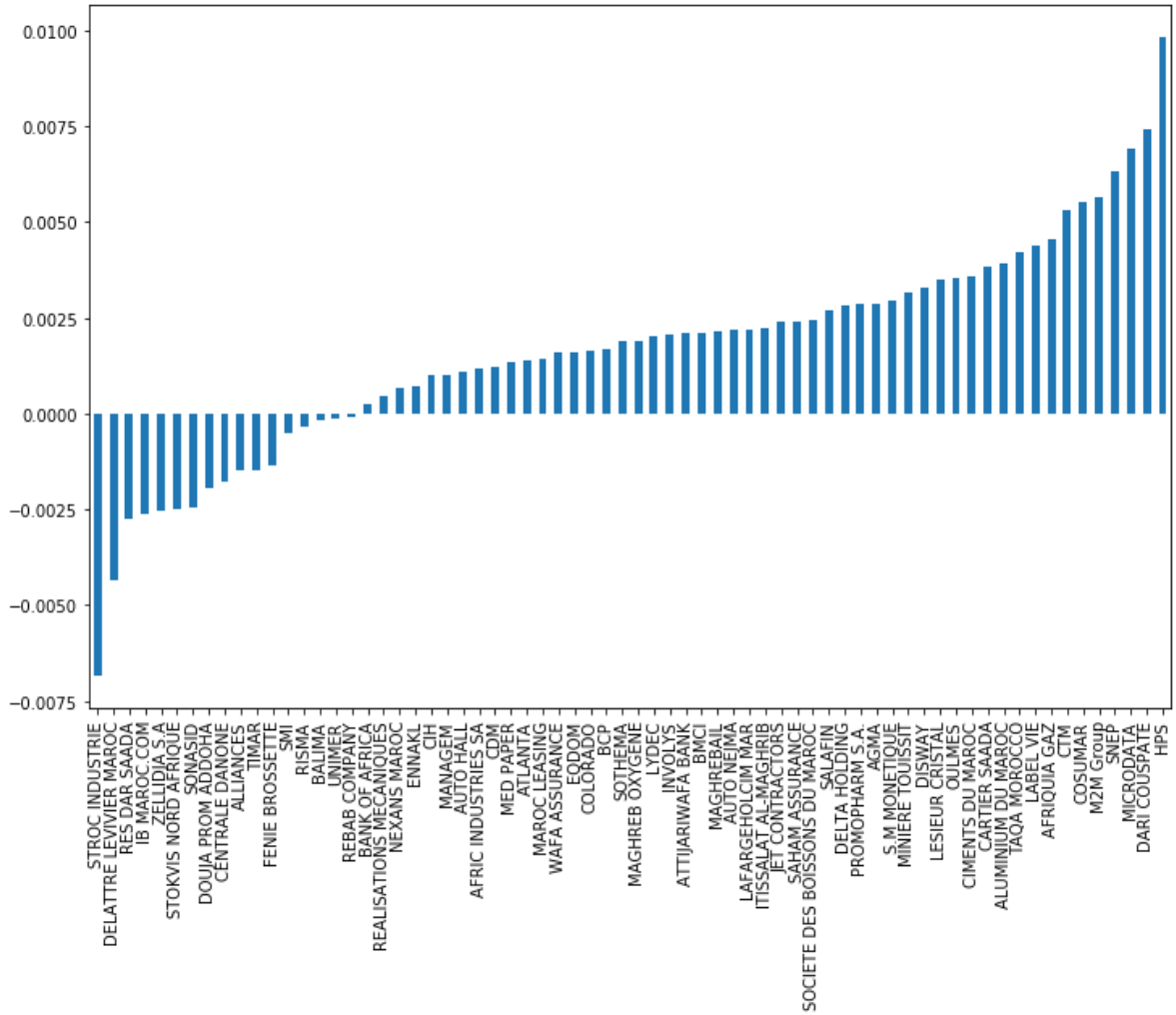
	DISWAY	ALLIANCES	IB MAROC.COM	ATLANTA	ZELLIDJA S.A	TAQA MOROCCO	SM
count	261.000000	261.000000	2.610000e+02	2.610000e+02	261.000000	261.000000	261.00000
mean	0.003302	-0.001489	-2.622401e-03	1.398602e-03	-0.002533	0.004200	-0.00051
std	0.037341	0.094299	7.738225e-02	3.048948e-02	0.061190	0.024808	0.04821
min	-0.152588	-0.355809	-2.305263e-01	-8.311445e-02	-0.213503	-0.075975	-0.22441
25%	-0.013357	-0.035417	-3.461538e-02	-1.598332e-02	0.000000	-0.009009	-0.01639
50%	0.000000	-0.005699	-1.110223e-16	-1.110223e-16	0.000000	0.000000	0.00000
75%	0.019553	0.026728	1.754386e-02	1.960784e-02	0.000000	0.016774	0.02214
max	0.161538	0.467156	4.023211e-01	1.256831e-01	0.319752	0.163403	0.27000

8 rows × 68 columns

Entrée [5]:

```
print(dataset.mean().sort_values().plot.bar(figsize=(12,8)))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```

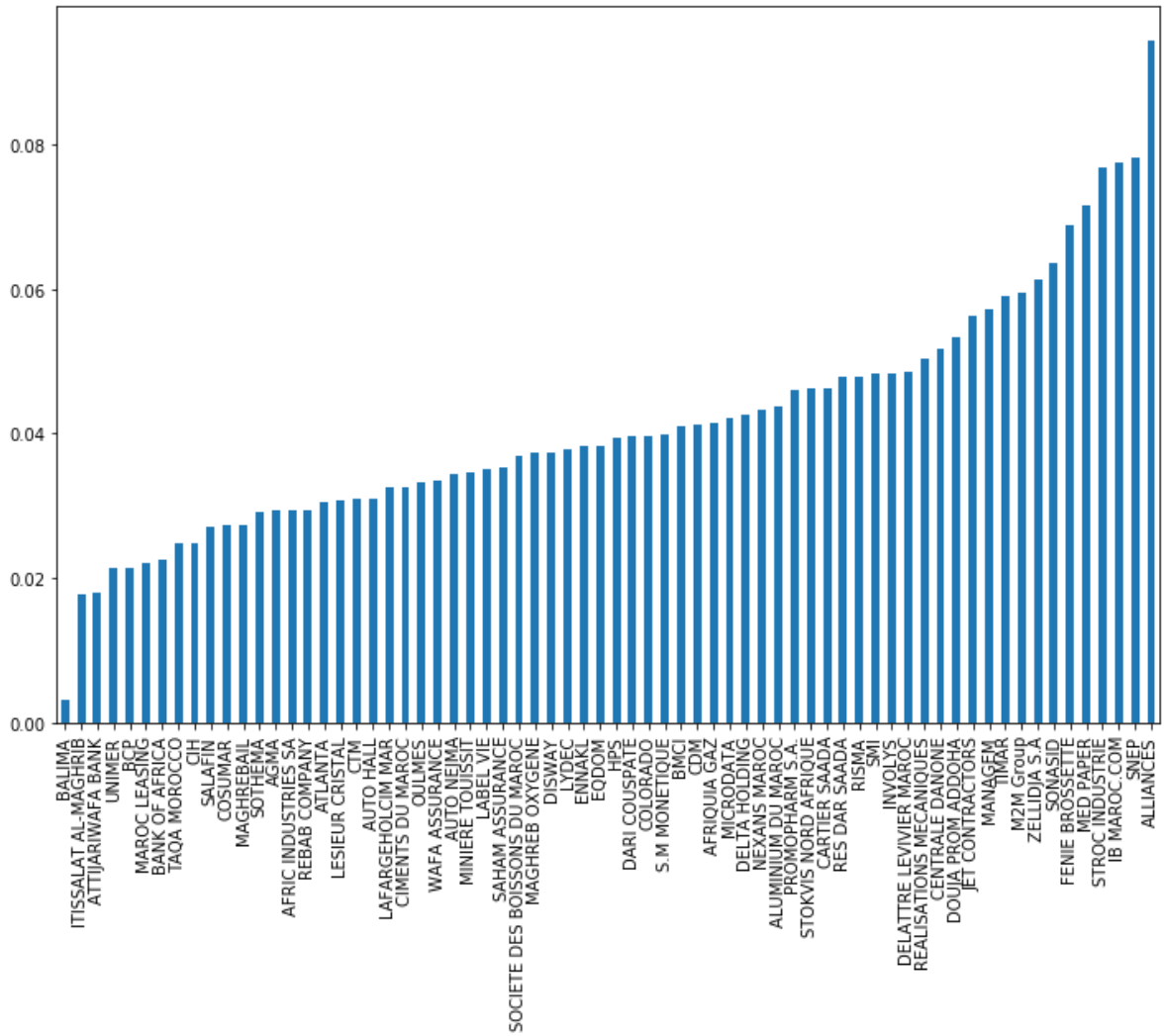


on remarque qu'il existe des titres qui ont un rendement élévé pendant que d'autres ont un rendement très négatif d'ou la nécessité d'une analyse minicieuse du portefeuille.

Entrée [6]:

```
print((dataset.std().sort_values().plot.bar(figsize=(12,8))))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



on remarque que comme les rendements, certains titres sont plus volatiles que d'autres. On peut voir le titre ALLIANCE qui a une très grande volatilité et un rendement négatif.

Vérification de quelques hypothèses

#vérification de la normalité

Avant d'effectuer des tests statistiques de normalité, nous regardons un peu les coefficients d'asymétrie et d'aplatissement de la distribution des rendements de chacun des actifs de notre dataset. Ces coefficients, s'ils sont respectivement proches de 0 et 3, suggèrent des distributions symétriques et aux queues peu épaisses, et donc proches de celle d'une loi normale.

calcul des coéfficients d'aplatissement

Entrée [7]:

```
from scipy.stats import norm, kurtosis
```

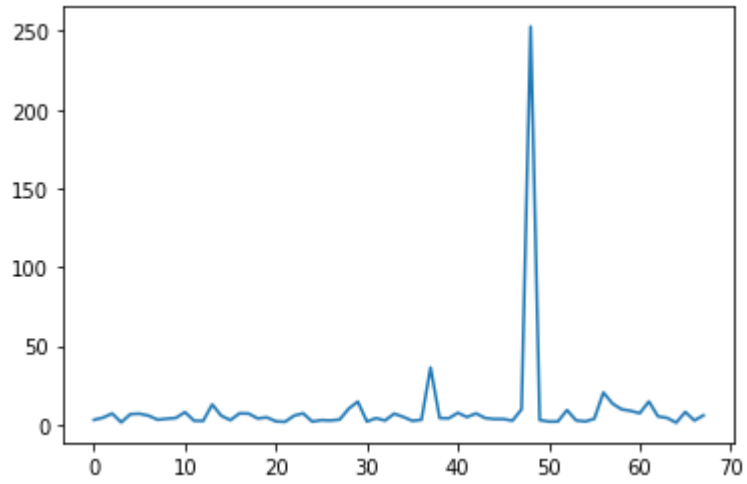
Entrée [8]:

```
x = np.linspace(-5, 5, 100)
ax = plt.subplot()
kur = kurtosis(dataset, fisher=True)
print(kur)
plt.plot(kur)
```

```
[  3.09399594   4.48344472   7.12276575   1.50161316   6.70154568
   6.99586568   5.74113529   3.20711408   3.79376891   4.28446786
   7.98802695   2.56948663   2.38826177  12.86751243   5.68548996
   2.91902272   7.21562645   7.07256031   3.97655435   4.66654161
   2.16759872   1.83453788   5.76180942   7.21757164   1.98747798
   2.93120532   2.69737144   3.23984885  10.1511329   14.68484326
   2.02942839   4.16793524   2.6680181    7.03159288   4.96091466
   2.48386958   3.14545014  36.3264566    4.2147361    4.05525004
   7.5592368    4.85419114   7.15576949   4.2322102    3.60807971
   3.63406625   2.56556089   9.74685614 252.79326856   2.96491129
   2.00772583   2.07463456   9.28301327   2.74218024   2.08015053
   3.76169373  20.4862172   13.39111487   9.78296646   8.79548027
   7.30088144  14.74887498   5.28169498   4.34447639   1.29866419
   8.07802192   2.67291571   5.96027361]
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x2284e542d48>]
```



à travers les valeurs et le graphique nous remarquons que la majeur partie des titres ont un coéffiient d'aplatissement loin de 3. On peut remarquer que le titre à la position 48 a un coéfficient égal à 252.79326856 ce qui est très enorme.Ces observations viennent intensifier notre intuition sur la non normalité de la distribution.

coéfficient d'asymétrie

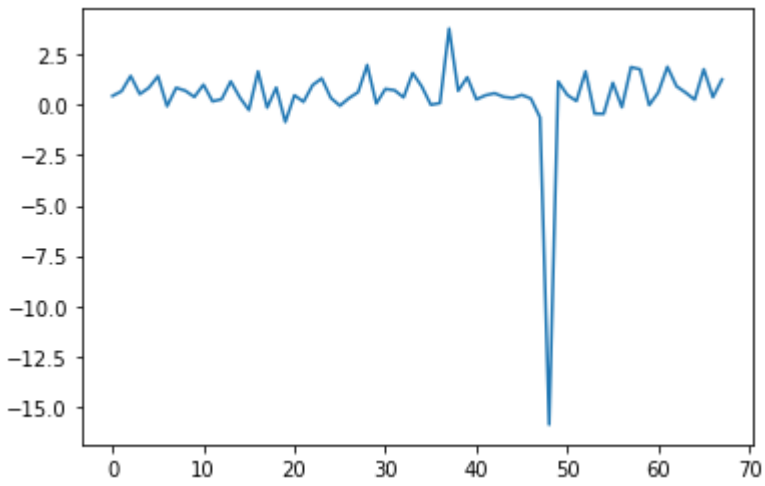
Entrée [9]:

```
x = np.linspace(-5, 5, 100)
ax = plt.subplot()
skw=scipy.stats.skew(dataset, axis=0)
print(skw)
plt.plot(skw)
```

```
[ 4.46402726e-01  6.83289608e-01  1.44271757e+00  5.39174031e-01
  8.55163430e-01  1.42957990e+00 -6.41824556e-02  8.51665969e-01
  7.07277243e-01  3.96753273e-01  1.00778760e+00  1.78404603e-01
  2.83197556e-01  1.17199442e+00  3.69656417e-01 -2.53508691e-01
  1.66628980e+00 -1.32308308e-01  8.60919337e-01 -8.49176152e-01
  4.78039268e-01  1.55436148e-01  9.84557537e-01  1.31949900e+00
  3.48818063e-01 -3.89955983e-02  3.39046206e-01  6.34556156e-01
  1.98561380e+00  7.05868416e-02  7.94805638e-01  7.30053914e-01
  3.75908553e-01  1.58427482e+00  9.14778461e-01  6.03380232e-03
  9.14569912e-02  3.79798107e+00  7.01716730e-01  1.37857064e+00
  2.74847836e-01  4.78904910e-01  5.80183561e-01  4.00022105e-01
  3.43531392e-01  4.98145000e-01  3.15632817e-01 -6.23985201e-01
 -1.59032887e+01  1.16434117e+00  4.92105961e-01  1.94925975e-01
  1.66544329e+00 -4.33783892e-01 -4.41548049e-01  1.10042118e+00
 -1.12989265e-01  1.88039928e+00  1.77196248e+00 -1.07017236e-02
  6.24210737e-01  1.89594945e+00  9.29140452e-01  6.15427539e-01
  2.60992086e-01  1.77512428e+00  3.85297976e-01  1.26445161e+00]
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x2284e2717c8>]
```



le titre 48 est visiblement une valeur aberrante. Nous allons donc chercher le titre auquel il correspond afin de l'éliminer de la base de données.

Entrée [10]:

```
print(skw.max())
print(skw.argmin())
dataset.columns[skw.argmin()]
```

```
3.7979810652467094
48
```

Out[10]:

```
'BALIMA'
```

Entrée [11]:

```
print(kur.max())
print(kur.argmax())
dataset.columns[kur.argmax()]
```

```
252.79326856030963
48
```

Out[11]:

```
'BALIMA'
```

le titre 48 qui est une valeur aberrante, correspond au titre BALIMA que nous allons éliminer de la base de données. Car un titre avec un coéfficient d'applatissement très élévé est un titre qui enregistre régulièrement des grosses pertes ou de gros gains. Ces valeurs peuvent donc biaiser notre analyse.

Entrée [12]:

```
dataset=dataset.drop(["BALIMA"],axis=1)
```

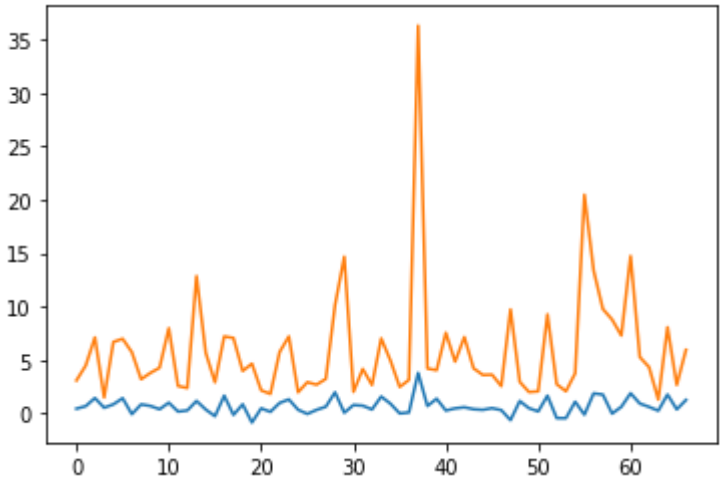
asymétrie et aplatissement

Entrée [13]:

```
x = np.linspace(-5, 5, 100)
ax = plt.subplot()
skw=scipy.stats.skew(dataset, axis=0)
kur = kurtosis(dataset, fisher=True)
plt.plot(skw)
plt.plot(kur)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x2284e401c88>]
```

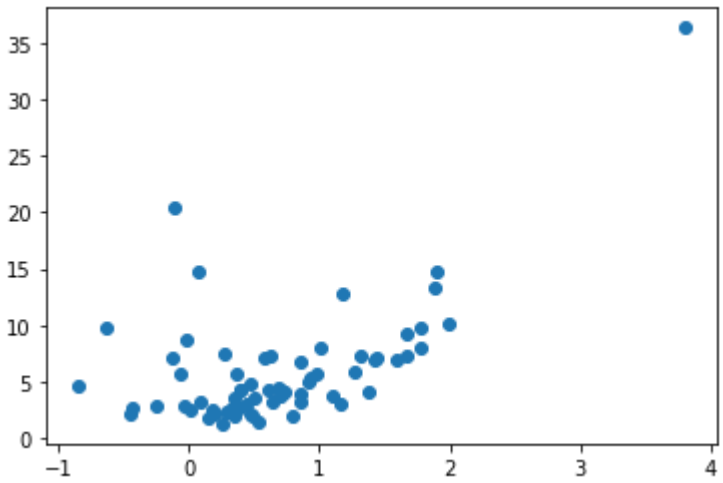


Entrée [14]:

```
plt.scatter(skw,kur)
```

Out[14]:

```
<matplotlib.collections.PathCollection at 0x2284e57d7c8>
```



à travers ce nuage de points nous remarquons qu'il existe des titres qui ont une distribution très loin de la normale. Tandis que d'autres même s'ils ne sont pas normaux, se raprohent un peu de la normalité.

Test de normalité

ici nous écrivons une fonction qui prend en argument notre base de données et nous donne en sortir les resultats des tests de Jarque Bera de Shapiro Wilk et de Kolmogorov.

Entrée [15]:

```
def is_jarque_shapiro_ks_normal(data,level=0.1):
    statistic_jarque,p_value_jarque=scipy.stats.jarque_bera(data)
    statistic_shapiro,p_value_shapiro=scipy.stats.shapiro(data)
    statistic_ks,p_value_ks=scipy.stats.kstest(data,"norm")
    return p_value_jarque>level, p_value_shapiro>level, p_value_ks>level
```

application du test sur tous nos titres

Entrée [16]:

```
res=dataset.aggregate(is_jarque_shapiro_ks_normal)
```

Entrée [17]:

```
res.value_counts()
```

Out[17]:

```
(False, False, False)    67
dtype: int64
```

d'après les résultats des trois tests, aucun de nos titres n'a une distribution signifiativement normale des rendements

Verifiation de l'hypothèse sur les corrélations entre les titres

Visualisation

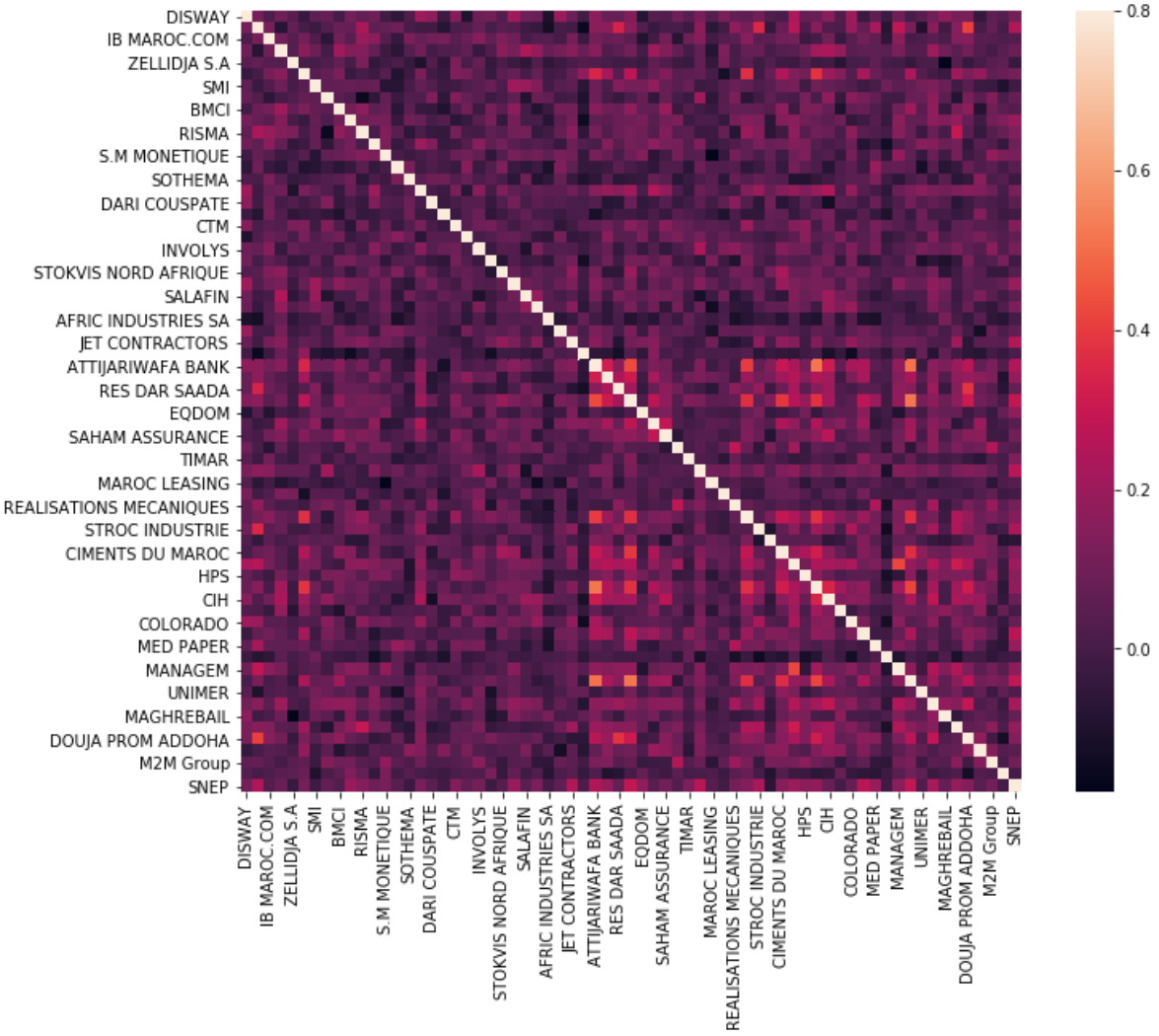
Entrée [18]:

```
corrmat = dataset.corr()
fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2284e5c4408>
```



On remarque qu'en générale que les titres sont corrélés entre eux. On a des corrélations positives fortes et aussi des corrélations négatives. cependant il n'y a pas de corrélation parfaite entre les titres.ceux-ci sont juste plus ou moins corrélés et cela est favorable à une diversification du portefeuille.D'autre part nous observons des corrélations négatives moins fortes cela n'est pas très favorable à une bonne diversifiation de portefeuille. cependant l'hypothèse sur la corrélation de Markowitz peut être considérée comme vérifiée.

Evolution du portefeuille

1- Calcul du drowdowns 2-observation de l'écart entre le rendement expéré et le rendement actuel 3-Portefeuille à poids égaux 4-simulation de plusieurs portefeuilles

Entrée [19]:

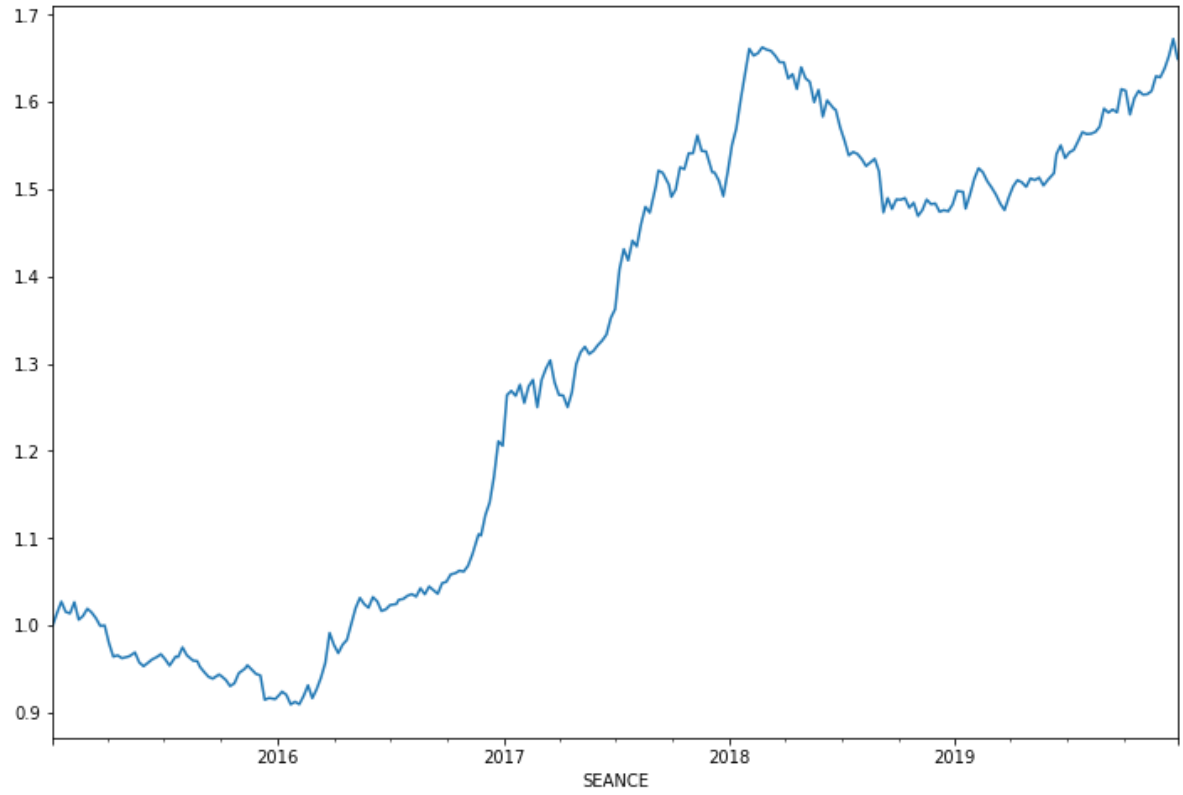
```
pds=np.ones((dataset.shape[1]))
pds=pds.reshape((1,pds.shape[0]))
pds = pds/np.sum(pds,axis=1)
def portefeuille(rendement:pd.Series,pds):
    les_poids=dataset.copy
    les_poids=pd.DataFrame(les_poids())
    les_poids.iloc[0,:]=pds
    les_poids.iloc[1:,:]=dataset.iloc[1:,:]+1
    wealth_index=(les_poids).cumprod()
    total=np.sum(wealth_index,axis=1)
    previous_peak=total.cummax()
    drowdowns= (total-previous_peak)/previous_peak
    return pd.DataFrame({"wealth":total,
                         "peak": previous_peak,
                         "drowdowns":drowdowns}),wealth_index
```

Entrée [20]:

```
drowdown,wealth_index=portefeuille(dataset,pds)
drowdown["wealth"].plot.line(figsize=(12,8))
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2284f2e5b08>
```



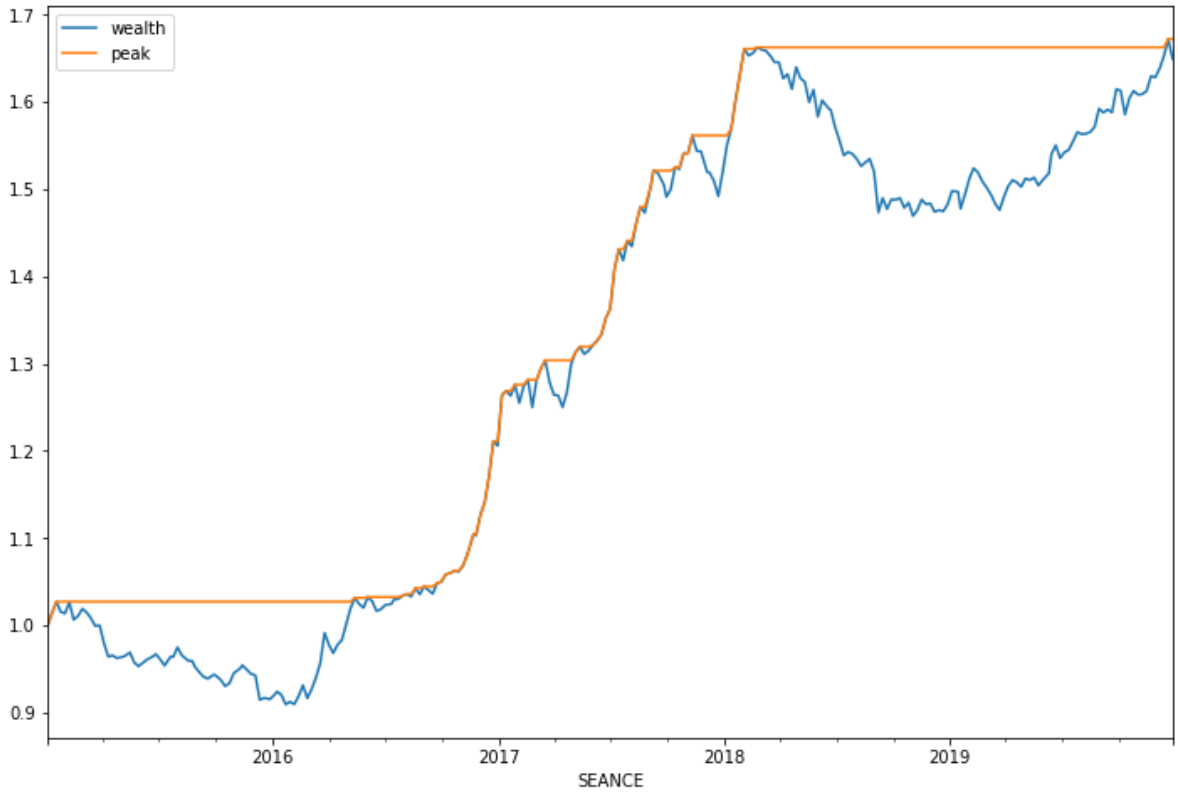
ici nous pouvons voir en temps réel l'évolution de notre portefeuille qui part de 1 à environ 1.6 ce qui correspond à une performence d'environ 70% sur les 5 ans pour un portefeuille à poids égal.

Entrée [21]:

```
drowdown[["wealth","peak"]].plot(figsize=(12,8))
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2284f55c8c8>
```



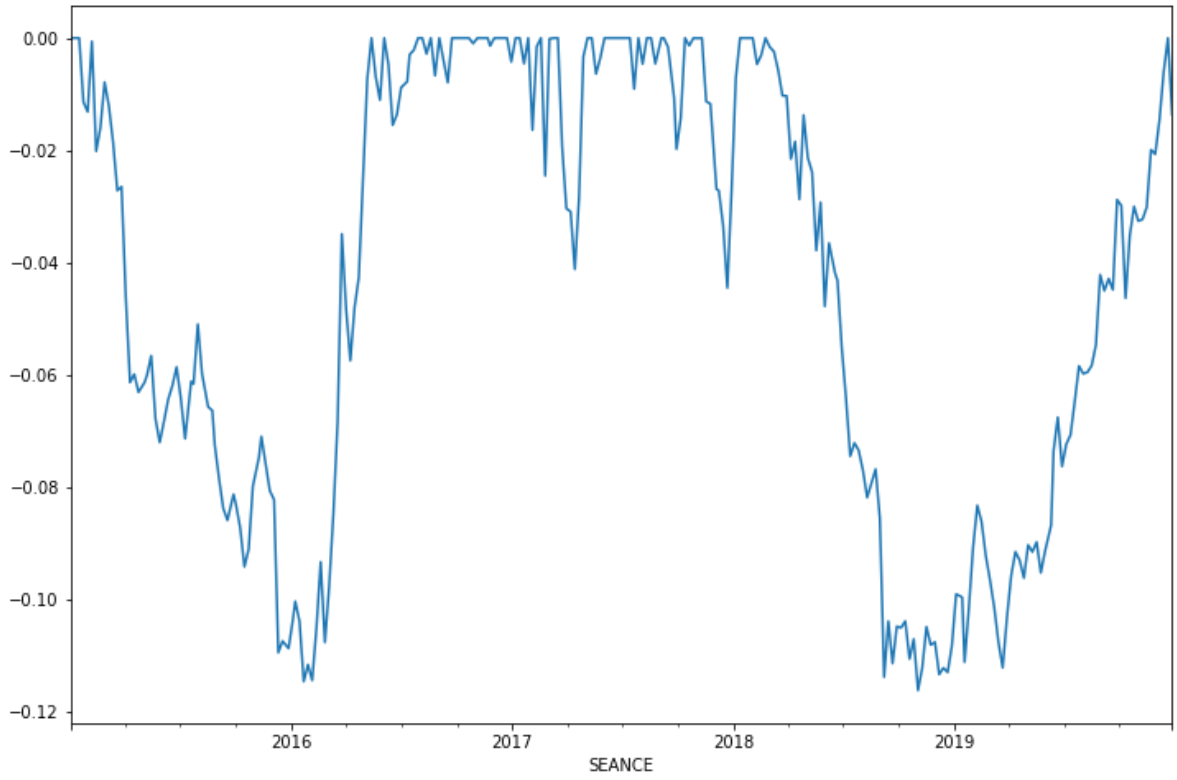
Sur ce graphique nous pouvons constater les éarts entre le rendement maximal de chaque période et le rendement actuel. On remarque que l'écart est grand sur toute la période de 2019. alors qu'il est Quasi-inexistant sur la période de fin 2016

Entrée [22]:

```
drowdown["drowdowns"].plot(figsize=(12,8))
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2284e542708>
```



Entrée [23]:

```
print(drowdown["drowdowns"].min())
print(drowdown["drowdowns"].idxmin())
```

```
-0.11625139692433314
2018-11-02
```

Entrée [24]:

```
print(drowdown["drowdowns"]["2019"].idxmax())
```

```
2019-12-20
```

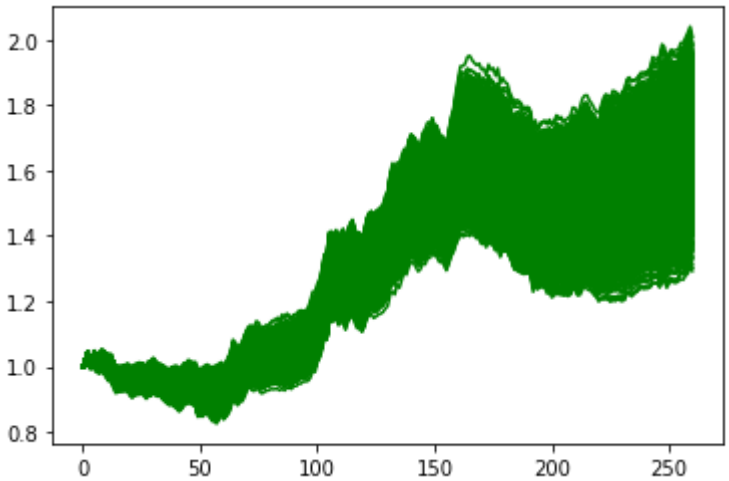
Ici, un investisseur malchanceux qui rentre sur le marché le 11-02-2018 se voit perdre 11,63 % de son capital en une journée, pour récupérer son capital il doit rester sur le marché en conservent ses titres jusqu'au 20-12-2019

Simulation de plusieurs portefeuilles

ici nous allons simuler un ensemble de portefeuille afin de voir comment la variation de portefeuille peut influencer le rendement et le risque.

Entrée [25]:

```
np.random.seed(41)
num_ports = 5000
all_weights = np.zeros(((num_ports),dataset.shape[0]))
weal_ind=np.zeros((num_ports,dataset.shape[1]))
for i in range(5000):
    pds = np.array(np.random.random(dataset.shape[1]))
    pds=pds.reshape((1,pds.shape[0]))
    pds = pds/np.sum(pds,axis=1)
    drowdown,wealth_index=portefeuille(dataset,pds)
    all_weights[i]=drowdown["wealth"]
    weal_ind[i,:]=wealth_index.iloc[0,:]
    #plt.scatter(rendvol[1],rendvol[0],c="g", cmap='viridis')
    plt.plot(all_weights[i],c='g')
```



Entrée [26]:

```
portmax=all_weights[:,-1]
print(portmax.argmax())
print(portmax.max())
print(portmax.argmin())
print(portmax.min())
```

```
1172
2.0089045026854286
3286
1.2925358076286428
```

le graphique nous donnee l'évolution de 5000 portefeuilles différents. On constate qu'en changeant de portefeuille on obtient des rendements différents. Dans le cas des 5000 portefeuilles, les rendements vont de 29,25% à environ 100% sur les 5 ans. Ce qui represente une différence significative. On peut donc conclure qu'en agissant juste sur les poids, on peut améliorer notre portefeuille.

nous pouvons voir que le portefeuille qui donne le plus grand rendement sur les 5000 portefeuilles est le portefeuille numéro 1172 tandis que celui qui donne le plus petit rendement est le numéro 3286.

Analyse détaillée des rendements

Entrée [27]:

```
wealth_index.describe()
```

Out[27]:

	DISWAY	ALLIANCES	IB MAROC.COM	ATLANTA	ZELLIDJA S.A	TAQA MOROCCO	SMI
count	261.000000	261.000000	261.000000	261.000000	261.000000	261.000000	261.000000
mean	0.053935	0.004789	0.000410	0.034742	0.008004	0.014521	0.018795
std	0.017249	0.002799	0.000147	0.006055	0.002599	0.003717	0.005294
min	0.027538	0.001248	0.000118	0.025376	0.004294	0.007052	0.009073
25%	0.039910	0.002710	0.000293	0.029251	0.006695	0.010730	0.014240
50%	0.052693	0.003698	0.000435	0.034340	0.007436	0.016082	0.018889
75%	0.064270	0.006469	0.000511	0.038705	0.007875	0.017660	0.021816
max	0.090965	0.012682	0.000899	0.047274	0.018135	0.019818	0.030086

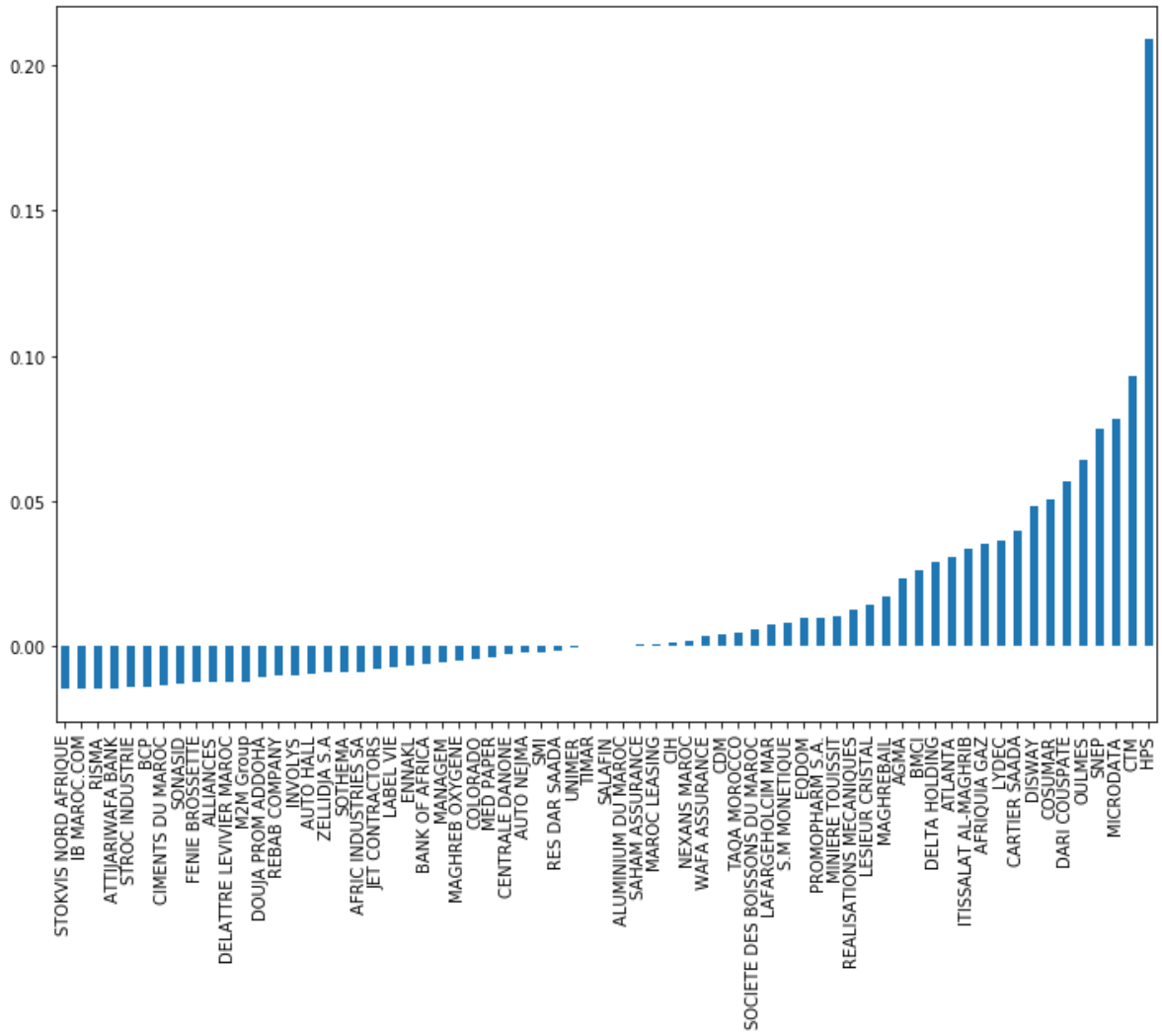
8 rows × 67 columns

Entrée [28]:

```
(wealth_index.iloc[-1,:]-1/dataset.shape[1]).sort_values().plot.bar(figsize=(12,8))
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22858e916c8>
```



on peut voir sur le graphique que même si le portefeuille a fait une performence générale importante,la réalité est que cette performence a été tirée vers le haut par certains titres en particulier. Et elle a aussi été tirée vers le bas par d'autres titres qui ont un rendement négatif.

Entrée [29]:

```
print((wealth_index.iloc[-1,:]>(pds[0,0])).value_counts())
print((wealth_index.iloc[-1,:]>((pds[0,0])*drowdown["wealth"][-1])).value_counts())
print((wealth_index.iloc[-1,:]<(pds[0,0])-0.5*(pds[0,0])).value_counts())
```

```
False    50
True     17
Name: 2019-12-27, dtype: int64
False    59
True      8
Name: 2019-12-27, dtype: int64
True     36
False    31
Name: 2019-12-27, dtype: int64
```

ici les détails sont clairs, on a 47 titres qui ont un rendement positif et 20 titres qui ont un rendement négatif. Parmi les titres ayant un rendement positif, 22 ont un rendement supérieur à celui du portefeuille. Et parmi les titres ayant un rendement négatif, 12 ont perdu plus de 50% du capital initial investi.

Entrée [30]:

```
drowdown["wealth"].describe()
```

Out[30]:

```
count    261.000000
mean       1.356551
std        0.309870
min        0.909735
25%        1.017784
50%        1.451892
75%        1.624891
max        1.794011
Name: wealth, dtype: float64
```

Optimisation du portefeuille selon Markowitz

Pour commencer nous allons définir des fonctions qui nous aideront par la suite à optimiser notre portefeuille et à tracer notre frontière d'éfficience.

les trois prémières fonctions, nous donnent respectivement le rendement annualisé, la volatilité annualisée et le ratio de sharpe de la base de données de rendements.

Entrée [31]:

```
def annual_returns(data):
    return ((dataset+1).prod())**(52/data.shape[0])-1

def annual_vol(data,nperiod_per_year):
    return data.std()*(nperiod_per_year**0.5)

def sharpe(returns,poids):
    ret=annual_returns(returns)
    vol=annual_vol(returns,52)
    return ret/vol
```

les deux fonctions suivantes nous aideront à calculer le rendement et la volatilité de chaque portefeuille donné en argument.

Entrée [32]:

```
def port_ret(returns,poids):
    return poids.T@returns

def port_vol(poids):
    return (np.sqrt(np.dot(poids.T, np.dot(dataset.cov()*52, poids))))
```

Les fonctions d'optimisation

la fonction optim prend en argument un rendement espéré par l'investisseur ainsi que le rendement annualisé et retourne le portefeuille(les poids) qui procure la volatilité minimale pour ce rendement.

Entrée [33]:

```
def optim(target_return,er):
    n=er.shape[0]
    init_guess=np.repeat(1/n,n)
    bounds=((0,1),)*n
    return_is_target=({ "type":"eq",
                        "args": (er,),
                        "fun":lambda er,weigth: target_return-port_ret(er,weigth)},
                         {"type":"eq",
                          "fun":lambda weigth: np.sum(weigth)-1
                         }
                     )

    results=scipy.optimize.minimize(port_vol,init_guess,method="SLSQP",options={"disp":Fals
    return results.x
```

La fonction opt_weigth le nombre de rendement espéré pour lesquels on veut minimiser la volatilité ainsi que le rendement annualisé et retourne un tableau contenant les portefeuilles qui réalisent ces minima.

Entrée [34]:

```
def opt_weigth(npoint,er):
    target_rt= np.linspace(er.min(),er.max(),npoint)
    weigth=[optim(target_return,er) for target_return in target_rt]
    return weigth

def plot_ef(npoint,er):
    weigths=opt_weigth(npoint,er)
    ret=[port_ret(er,w) for w in weigths]
    vol=[port_vol(w) for w in weigths]
    ef= pd.DataFrame({
        "return":ret,
        "volatility": vol
    })
    return ef.plot.line(x="volatility", y="return",style='.-')
```

Traçage de la frontière d'éfficience

Entrée [35]:

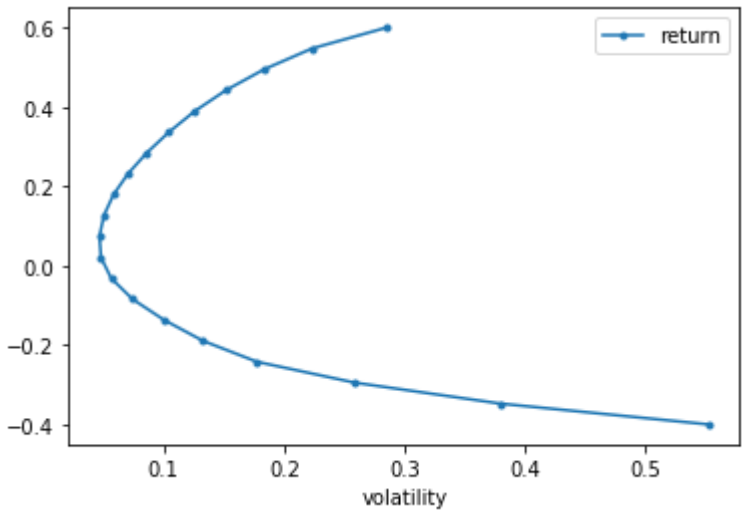
```
er=annual_returns(dataset)
```

Entrée [36]:

```
plot_ef(20,er)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22858ca74c8>
```



Nous pouvons voir ici la frontière d'éfficience tracée. On remarque qu'elle a la forme d'une parabole renversée. Cela parce que la representation de la volatilité en fonction du rendement est une parabole. Par consequent une volatilité donnée correspond à deux rendements possibles. On élimine le plus petit pour deux raisons. 1- il est plus faible que le rendement de l'actif sans risque. 2- l'investisseur est rationnel donc préfère toujours plus à moins.

Autres mesure du risque

1-VaR 2-CVaR

La VaR et la CVaR sont des mésures du risque et de l'aversion au risque de l'investisseur. Ils viennent combler les limites de la théorie de Markowitz en matière d'estimation de l'aversion au risque de l'investisseur.

Entrée [51]:

```
def Var(data,level=5):
    if isinstance(data,pd.Series):
        return np.percentile(data,level)
    elif isinstance(data,pd.DataFrame) :
        return data.aggregate(Var,level=level)
    else:
        raise typeError('data maust be DataFrame or Series')
```

Entrée [52]:

```
Var=Var(dataset)
```

Entrée [53]:

```
Var.mean()
```

Out[53]:

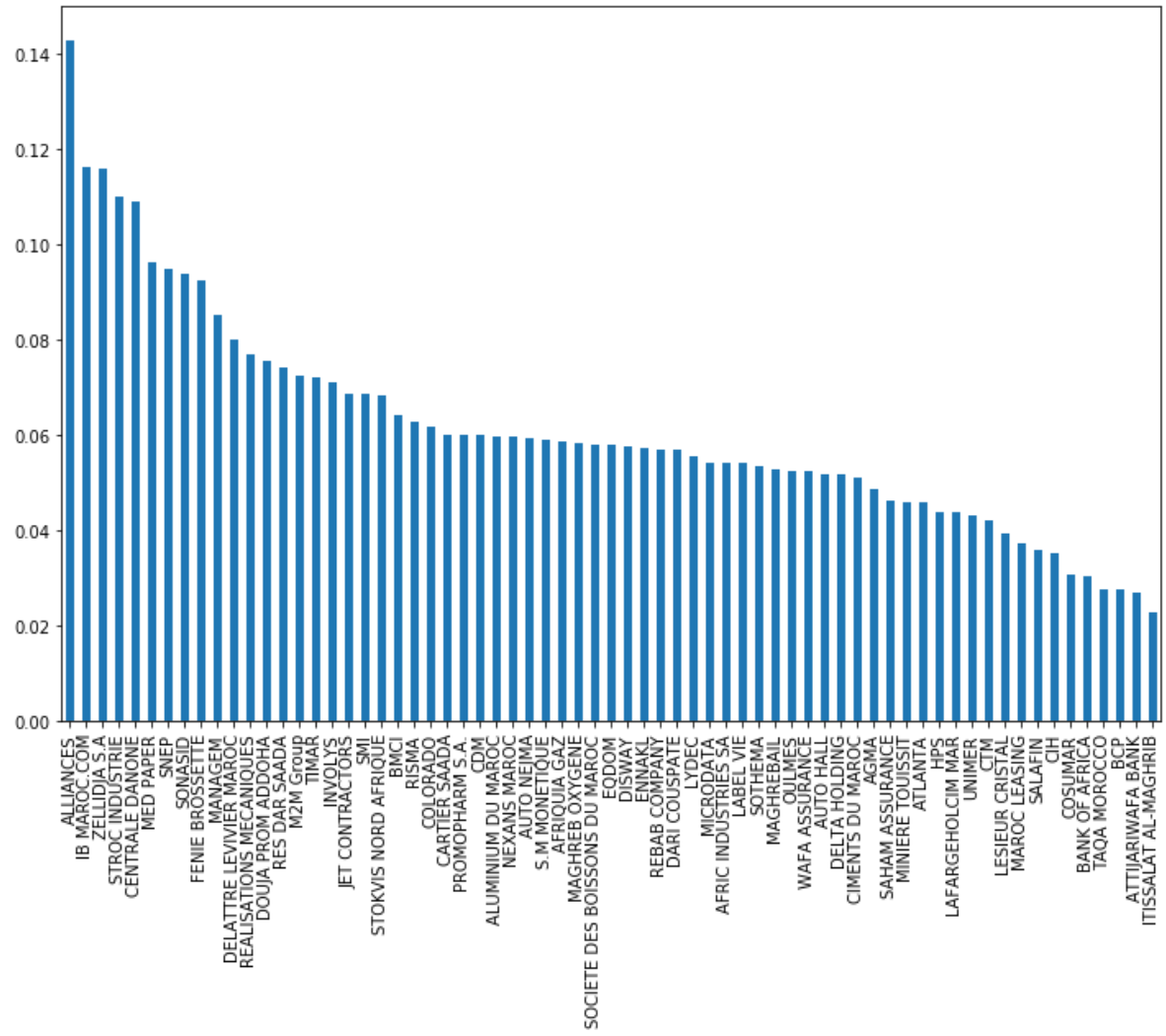
```
-0.06138492526792504
```

Entrée [55]:

```
(-Var.sort_values()).plot.bar(figsize=(12,8))
```

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22858dc86c8>
```



on remarque en générale les titres ont une value at risk supérieure à 0.02 ce qui signifie qu'il y a 5% de chance que tous les titres perdent au moins 2% du capital investi sur une année. En plus de cela on a des titres tels que ALLIANCE et IB MAROC COM qui ont une value at risk élévée. Avec une autre manière de voir, on peut dire qu'à 95% ALLIANCE ne perdra pas plus de 14% du capital investi.

Entrée [74]:

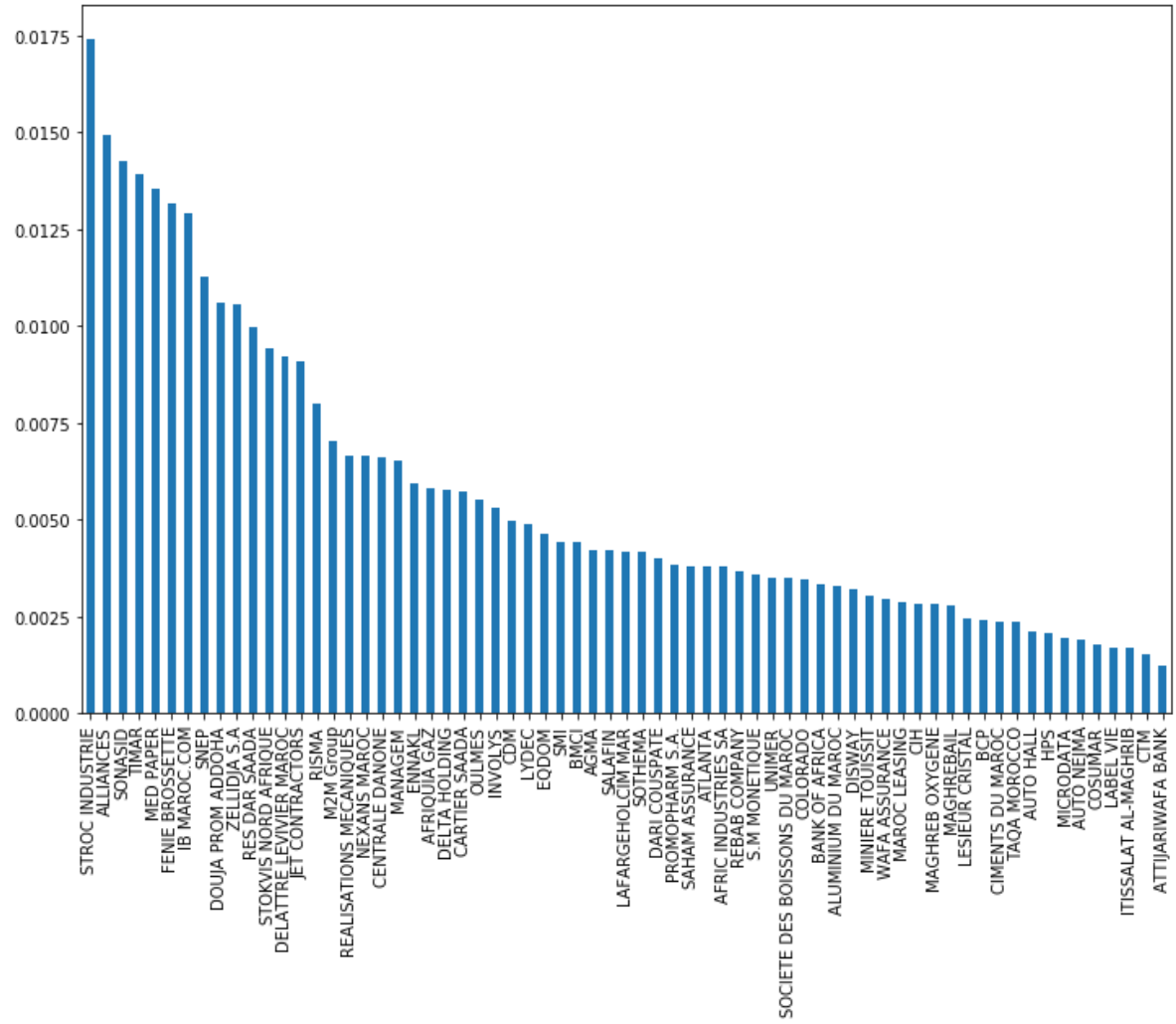
```
def CVar(data,level=5):
    if isinstance(data,pd.DataFrame):
        return data.aggregate(CVar,level=level)
    elif isinstance(data,pd.Series):
        inferieur=data<=-Var(data,level=level)
        return data[inferieur].mean()
    else:
        raise typeError('data maust be DataFrame or Series')
```

Entrée [173]:

```
(-CVar(dataset).sort_values()).plot.bar(figsize=(12,8))
```

Out[173]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f297297d88>
```



la CVaR vient apporter un soutient à la VaR car la VaR nous donne la perte maximale à ne pas depasser à 95% au mieux. Mais dans les 5% au pire il nous donne pas la perte maximale qu'on peut atteindre. La CVaR nous donne la moyenne des pertes dans les 5% au pire des cas. On peut deduire ici qu'au cas où l'action STROC INDUSTRIE n'a pas la chance de son côté, il perdra en moyenne 17.