LUISS Guido Carli
LIBERA UNIVERSITA' INTERNAZIONALE DEGLI STUDI SOCIALI

Dipartimento di Impresa e Management
Corso di Laurea Triennale in Economia e Management
Cattedra di Matematica Finanziaria

Applicazioni Pratiche della Moderna Teoria della Selezione del Portafoglio su Python

Relatore:
Emerito Prof. Gennaro Olivieri

Candidato:
Lorenzo Loreti
Matr. 214651

Anno Accademico 2018/2019

Indice

INTRODUZIONE

Questo elaborato ha la finalità di svolgere un'analisi empirica della moderna teoria della selezione del portafoglio tramite il linguaggio di programmazione Python.
Inizialmente, l'analisi consiste nello scaricare i dati storici dei titoli presi in considerazione, per poi tracciare due grafici con le quotazioni di chiusura e i rendimenti giornalieri.
Poi ho eseguito un test di normalità delle distribuzioni dei titoli, per poter dare un giudizio riguardo all'affidabilità delle statistiche utilizzate ai fini dell'analisi e mostrare le criticità delle assunzioni sottostanti al modello.
Ho effettuato anche uno studio sulla forma delle distribuzioni tramite gli indici di skewness e di curtosi, per poter trarre delle conclusioni sull'affidabilità della standard deviation come misura di rischio, e di come la forma della distribuzione dei rendimenti può influenzare tale statistica.
Successivamente, avendo preso consapevolezza dei limiti della standard deviation per l'analisi empirica, ho introdotto un altro indicatore di rischio, il value at risk.
Adottando il metodo della simulazione storica per calcolarlo, ho quantificato il rischio al verificarsi di rendimenti molto negativi, svincolando temporaneamente l'analisi dall'ipotesi di normalità delle distribuzioni.
Nel selezionare i titoli per l'analisi di portafoglio ho preso a riferimento il grafico valore atteso versus varianza.
Ovvero ho selezionato quelle azioni che descrivessero, almeno approssimativamente, una relazione lineare crescente tra i due momenti.
La logica sottostante tale criterio è quella di selezionare dei titoli che siano efficienti e non dominati da altri; infatti non avrebbe senso, ad esempio, scegliere un titolo con stessa varianza di un altro, già selezionato per la costruzione del portafoglio, ma con minore rendimento atteso.
Fase successiva dell'analisi è quella di calcolare la matrice delle covarianze e degli indici di correlazione.
Tali valori sono di fondamentale importanza ai fini del calcolo del rischio di portafoglio, infatti ciò che determina la sua volatilità non è la somma dei rischi dei singoli titoli, ma come i titoli covariano tra loro all'interno del portafoglio.
Tale principio, conosciuto come effetto diversificazione, permette di distinguere il rischio di portafoglio in due parti.
Il rischio specifico del singolo titolo il quale può essere eliminato tramite la diversificazione, e il rischio sistematico il quale coinvolge tutti i titoli presenti nel portafoglio ed è dovuto a fattori di mercato.
Il rischio di portafoglio può essere considerato come somma delle covarianze all'interno della matrice delle covarianze opportunamente ponderate per le percentuali dei titoli investiti nel portafoglio a cui una covarianza si riferisce.
All'aumentare del numero di titoli il numero di covarianze aumenta più che proporzionalmente rispetto al numero delle varianze (covarianza di un titolo con sé stesso) presenti nella diagonale principale della matrice delle covarianze.
E dal momento che le varianze rappresentano il rischio specifico di un titolo, si dimostra intuitivamente come il loro contributo al rischio del portafoglio tenda a 0 al crescere del numero dei titoli, mentre aumenti quello delle covarianze che rappresentano il rischio sistematico.
Poi ho considerato un indice di mercato per calcolare il beta dei singoli titoli, ovvero la sensibilità del rendimento di ogni titolo all'andamento del mercato.
Ciò può essere utile per il calcolo del beta del portafoglio, il quale è la media ponderata dei beta dei singoli titoli.
Successivamente tramite una simulazione Monte Carlo ho tracciato la frontiera dei portafogli, assegnando pesi casuali ai titoli del portafoglio sotto il vincolo che la somma dei pesi sia uguale a uno.

In tal modo, ho trovato statisticamente il portafoglio di minima varianza e il portafoglio con la massima Sharpe ratio.
Quest'ultimo portafoglio è utile nel processo di Asset Allocation, nel quale si traccia la capital allocation line ottimale passante nei punti corrispondenti al titolo privo di rischio e al portafoglio con la massima Sharpe ratio e permette di derivare quindi una seconda frontiera efficiente che domina la prima.
Infine, ho fatto un grafico che mette a confronto la frontiera efficiente dei portafogli con i titoli presi singolarmente di cui tali portafogli si compongono.
Il risultato è che non è razionale investire tutta la propria ricchezza in singoli titoli, in quanto investendo in un portafoglio composto da più titoli, è possibile avere un investimento più efficiente, ovvero con una migliore combinazione rischio-rendimento.

CAPITOLO 1: ANALISI SU PYTHON DI UN PORTAFOGLIO COMPOSTO DA 5 TITOLI

1.1 I dati storici dei titoli

L'analisi da me svolta consiste nello scaricare da Yahoo Finance le quotazioni giornaliere di 5 titoli negoziati nel mercato finanziario italiano comprese in un arco temporale di 3 anni, e attraverso l'uso del linguaggio di programmazione Python, fare un'applicazione pratica della teoria della selezione del portafoglio.
I titoli che ho selezionato ai fini dell'analisi sono azioni di società quotate in Borsa appartenenti a settori produttivi diversi.
La prima società, ENI (Ente Nazionale Idrocarburi), è tra i principali produttori europei di petrolio e gas (codice di Borsa: ENI.MI).
La seconda, Eurotech, è un'azienda specializzata nella progettazione, produzione e commercializzazione di computer integrati per uso professionale (codice di Borsa: ETH.MI).
La terza, Autogrill S.p.A, è il leader mondiale dei servizi di ristorazione per chi viaggia.
(codice di Borsa: AGL.MI).
La quarta, STMicroelectronics N.V, figura tra i leader mondiali del mercato dei semiconduttori (codice di Borsa: STM.MI).
La quinta, FinecoBank Banca Fineco S.p.A, figura tra i principali gruppi di servizi finanziari italiani (codice di Borsa: FBK.MI).
Ai fini di una migliore esposizione nel corso dei capitoli che seguiranno, riporto il codice Python suddiviso in fasi.
Ogni fase è composta da un blocco di codice, seguito dalla corrispondente spiegazione tecnica e analisi finanziaria.

```
import pandas_datareader.data as web

nome_titoli=['ENI.MI','ETH.MI','AGL.MI','STM.MI','FBK.MI']
data_inizio='2016-7-26'
data_fine='2019-7-26'
print('inizio caricamento')
dati_eni=web.get_data_yahoo(nome_titoli[0], data_inizio, data_fine, interval='d')
print(nome_titoli[0] ,dati_eni.shape)
dati_eth=web.get_data_yahoo(nome_titoli[1], data_inizio, data_fine, interval='d')
print(nome_titoli[1] ,dati_eth.shape)
dati_agl=web.get_data_yahoo(nome_titoli[2], data_inizio, data_fine, interval='d')
print(nome_titoli[2] ,dati_agl.shape)
dati_stm=web.get_data_yahoo(nome_titoli[3], data_inizio, data_fine, interval='d')
print(nome_titoli[3],dati_stm.shape)
dati_fbk=web.get_data_yahoo(nome_titoli[4], data_inizio, data_fine, interval='d')
print(nome_titoli[4],dati_fbk.shape)
print ('fine caricamento dati')
```

```
inizio caricamento
ENI.MI (763, 6)
ETH.MI (763, 6)
AGL.MI (763, 6)
STM.MI (763, 6)
FBK.MI (763, 6)
fine caricamento dati
```

La funzione pandas_datareader.data permette di estrarre dati provenienti da varie risorse su Internet e inserirli all'interno di un pandas DataFrame.

Successivamente, creo un vettore (o lista) di stringhe con i codici di Borsa delle azioni di cui voglio scaricare le quotazioni, e poi definisco due variabili “data_inizio” e “data_fine” contenenti gli estremi temporali del campione di dati storici oggetto dell’analisi.
I vettori di stringhe appena definiti servono come parametri per la funzione get_data_yahoo, la quale permette di scaricare le quotazioni dei titoli da Yahoo Finance e inserirli all’interno di un DataFrame (interval=’d’ indica alla funzione di considerare quotazioni giornaliere).
Utilizzando la funzione DataFrame.shape, si ottiene una tupla con all’interno la dimensione del DataFrame.
La schermata di output indica le dimensioni dei 5 DataFrame creati; ovvero 763 righe e 6 colonne, essi risultano pertanto compatibili tra loro ai fini dell’analisi essendo della stessa dimensione.

CAPITOLO 2: LE QUOTAZIONI DI CHIUSURA E I RENDIMENTI

2.1 Calcolo delle quotazioni di chiusura aggiustate e dei rendimenti logaritmici

```
import numpy as np
quotazioni=dati_eni.values
lista_chiusura_eni=quotazioni[:,5]
quotazioni=dati_eth.values
lista_chiusura_eth=quotazioni[:,5]
quotazioni=dati_agl.values
lista_chiusura_agl=quotazioni[:,5]
quotazioni=dati_stm.values
lista_chiusura_stm=quotazioni[:,5]
quotazioni=dati_fbk.values
lista_chiusura_fbk=quotazioni[:,5]

lista_rendimenti_eni=np.array([])
lista_rendimenti_eth=np.array([])
lista_rendimenti_agl=np.array([])
lista_rendimenti_stm=np.array([])
lista_rendimenti_fbk=np.array([])
i=0
lunghezza=lista_chiusura_eni.size
while (i<lunghezza-1):
    rendimento=np.log(lista_chiusura_eni[i+1]/lista_chiusura_eni[i])
    lista_rendimenti_eni=np.append(lista_rendimenti_eni,[rendimento])

    rendimento=np.log(lista_chiusura_eth[i+1]/lista_chiusura_eth[i])
    lista_rendimenti_eth=np.append(lista_rendimenti_eth,[rendimento])

    rendimento=np.log(lista_chiusura_agl[i+1]/lista_chiusura_agl[i])
    lista_rendimenti_agl=np.append(lista_rendimenti_agl,[rendimento])

    rendimento=np.log(lista_chiusura_stm[i+1]/lista_chiusura_stm[i])
    lista_rendimenti_stm=np.append(lista_rendimenti_stm,[rendimento])

    rendimento=np.log(lista_chiusura_fbk[i+1]/lista_chiusura_fbk[i])
    lista_rendimenti_fbk=np.append(lista_rendimenti_fbk,[rendimento])
    i=i+1

    print ('dati presenti=', lunghezza)
```

```
dati presenti= 763
```

Con la funzione DataFrame.values genero una rappresentazione numpy dei pandas DataFrame contenenti i dati storici dei titoli scaricati da Yahoo Finance, tale funzione restituisce solo i valori all'interno del DataFrame, ovvero senza le etichette di riga e di colonna.
Successivamente, creo per ogni titolo dei vettori numpy denominati "lista_chiusura" con all'interno le quotazioni di chiusura aggiustate per dividendi e frazionamenti, corrispondenti alla sesta colonna del DataFrame.
Poi genero degli altri vettori numpy vuoti denominati "lista_rendimenti", i quali saranno riempiti con i rendimenti logaritmici giornalieri dei titoli attraverso un ciclo di while.
Per rendimenti logaritmici intendo i tassi istantanei di rendimenti dei titoli, i quali a differenza dei rendimenti calcolati come semplice variazione percentuale delle quotazioni giornaliere, hanno maggiori proprietà statistiche, e inoltre sono un migliore indicatore di performance dell'investimento, in quanto tengono conto dell'effetto della capitalizzazione.

In particolare, assumendo che le quotazioni istantanee dei titoli abbiano una distribuzione lognormale, allora i rendimenti capitalizzati nel continuo da esse prodotti possono considerarsi normalmente distribuiti a prescindere dalla durata dell'investimento.
Si definisce il tasso istantaneo di rendimento del titolo i in t come

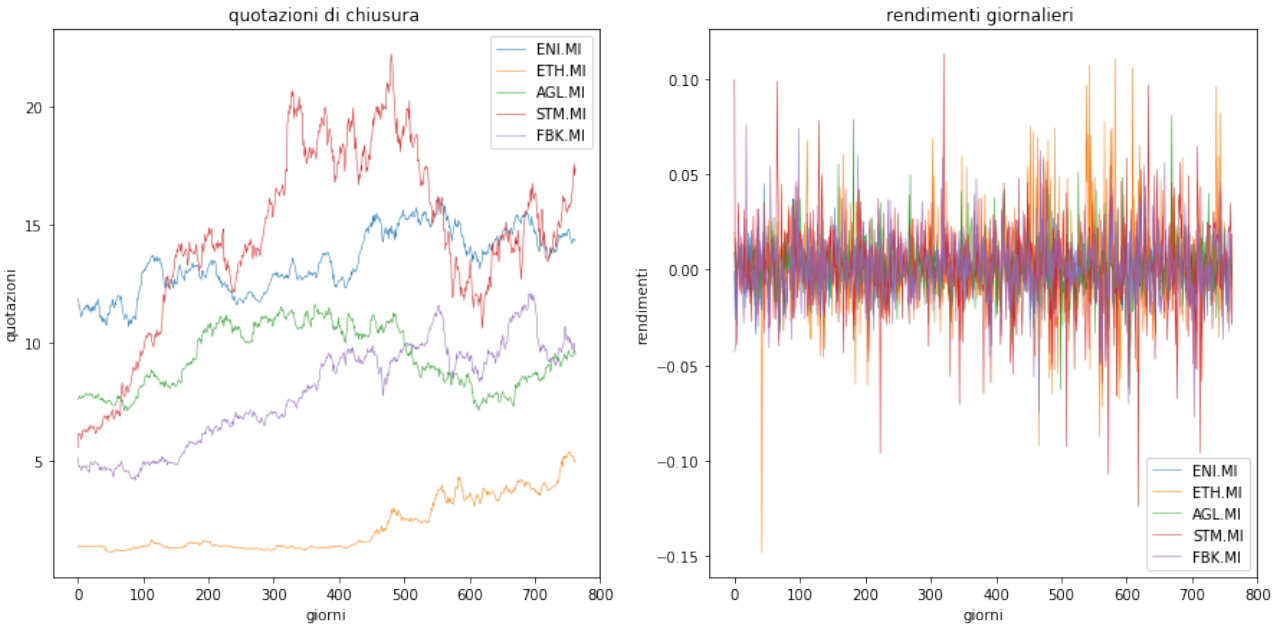
$$\delta_i(t) = log\left(\frac{P_{i,t+1}}{P_{i,t}}\right)$$

Dove:
$\delta_i(t)$ è il tasso istantaneo di rendimento del titolo i in t;
$P_{i,t}$ è la quotazione dell'i-esimo titolo all'istante t;
$P_{i,t+1}$ è la quotazione dell'i-esimo titolo all'istante t+1.

2.2 Grafico delle quotazioni di chiusura e dei rendimenti giornalieri

```
import matplotlib.pyplot as plt
x_dim_eni=lista_rendimenti_eni.shape
x_dim_eni =x_dim_eni[0]
print(x_dim_eni)
import numpy as np
giorni_rendimenti=np.arange(0,x_dim_eni,1)
giorni_quotazioni=np.arange(0,x_dim_eni+1,1)
fig,frame= plt.subplots(1, 2,figsize=(15, 7),)
plt.rcParams['lines.linewidth'] = 0.5
plt.rcParams['lines.markersize'] = 2
frame[0].plot(giorni_quotazioni,lista_chiusura_eni,label=nome_titoli[0])
frame[0].plot(giorni_quotazioni,lista_chiusura_eth,label=nome_titoli[1])
frame[0].plot(giorni_quotazioni,lista_chiusura_agl,label=nome_titoli[2])
frame[0].plot(giorni_quotazioni,lista_chiusura_stm,label=nome_titoli[3])
frame[0].plot(giorni_quotazioni,lista_chiusura_fbk,label=nome_titoli[4])
frame[0].set_xlabel('giorni')
frame[0].set_ylabel('quotazioni')
frame[0].set_title('quotazioni di chiusura ')

frame[1].plot(giorni_rendimenti,lista_rendimenti_eni,label=nome_titoli[0])
frame[1].plot(giorni_rendimenti,lista_rendimenti_eth,label=nome_titoli[1])
frame[1].plot(giorni_rendimenti,lista_rendimenti_agl,label=nome_titoli[2])
frame[1].plot(giorni_rendimenti,lista_rendimenti_stm,label=nome_titoli[3])
frame[1].plot(giorni_rendimenti,lista_rendimenti_fbk,label=nome_titoli[4])
frame[1].set_xlabel('giorni')
frame[1].set_ylabel('rendimenti')
frame[1].set_title('rendimenti giornalieri ')
frame[0].legend()
frame[1].legend()
plt.show()
```

```
762
```



Tramite l'utilizzo della libreria matplotlib ho creato due grafici riferiti ai 5 titoli, quello a sinistra raffigura le quotazioni di chiusura giornaliere, mentre quello a destra i rendimenti giornalieri.
Inizialmente definisco la variabile "x_dim_eni" e la pongo uguale alla dimensione del vettore "lista_rendimenti" (ad esempio di Eni, ma è indifferente perché i vettori hanno la stessa dimensione) tramite l'utilizzo della funzione numpy.shape.
Tale funzione fornisce una tupla con all'interno la dimensione di tale vettore (ovvero il numero di elementi), e la prendo in riferimento per stabilire la lunghezza dell'intervallo temporale dei grafici.
La dimensione del vettore dei rendimenti è 762 e quindi inferiore di quella delle quotazioni di chiusura (763), in quanto non è possibile calcolare il rendimento nel giorno più recente del campione (2019-7-26).
Quindi il grafico delle quotazioni di chiusura ha un intervallo temporale più lungo di un giorno di quello dei rendimenti.

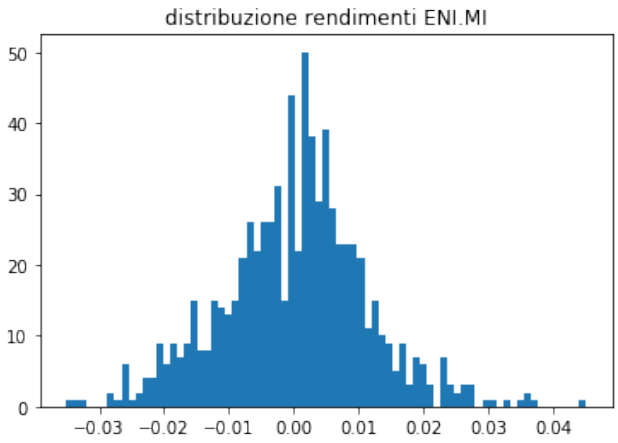
CAPITOLO 3: TEST DI NORMALITA' DELLE DISTRIBUZIONI

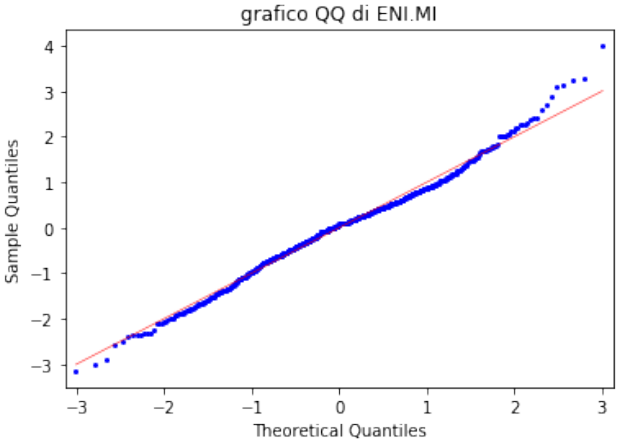
3.1 Lo Shapiro Wilk Test e il quantile-quantile plot

```
import matplotlib.pyplot as plt
import statsmodels.api as sm
import pylab
from scipy import stats

plt.hist(lista_rendimenti_eni, bins=75, density=False)
stringa = 'distribuzione rendimenti '+ nome_titoli[0]
plt.title(stringa)
plt.show()
Shapiro_Wilk_test= stats.shapiro(lista_rendimenti_eni)
p_value=Shapiro_Wilk_test[1]
print('il p-value è:',p_value)
if p_value <= 0.05:
    print("l'ipotesi nulla di normalità è rifiutata")
else:
    print("l'ipotesi nulla di normalità è accettata")
sm.qqplot(lista_rendimenti_eni,line='s',dist = stats.norm, fit = True)
stringa = 'grafico QQ di '+ nome_titoli[0]
plt.title(stringa)
pylab.show()

plt.hist(lista_rendimenti_eth, bins=75, density=False)
stringa = 'distribuzione rendimenti '+ nome_titoli[1]
plt.title(stringa)
plt.show()
Shapiro_Wilk_test= stats.shapiro(lista_rendimenti_eth)
p_value=Shapiro_Wilk_test[1]
print('il p-value è:',p_value)
if p_value <= 0.05:
    print("l'ipotesi nulla di normalità è rifiutata")
else:
    print("l'ipotesi nulla di normalità è accettata")
sm.qqplot(lista_rendimenti_eth,line='s',dist = stats.norm, fit = True)
stringa = 'grafico QQ di '+ nome_titoli[1]
plt.title(stringa)
pylab.show()
```

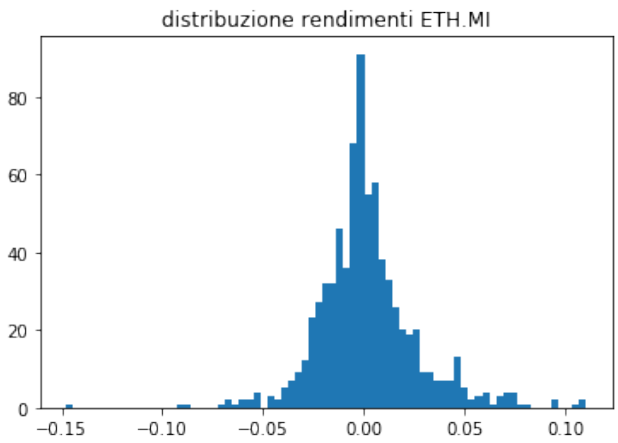
```
plt.hist(lista_rendimenti_agl, bins=75, density=False)
stringa = 'distribuzione rendimenti '+ nome_titoli[2]
plt.title(stringa)
plt.show()
Shapiro_Wilk_test= stats.shapiro(lista_rendimenti_agl)
p_value=Shapiro_Wilk_test[1]
print('il p-value è:',p_value)
if p_value <= 0.05:
    print("l'ipotesi nulla di normalità è rifiutata")
else:
    print("l'ipotesi nulla di normalità è accettata")
sm.qqplot(lista_rendimenti_agl, line='s',dist = stats.norm, fit = True)
stringa = 'grafico QQ di '+ nome_titoli[2]
plt.title(stringa)
pylab.show()
```

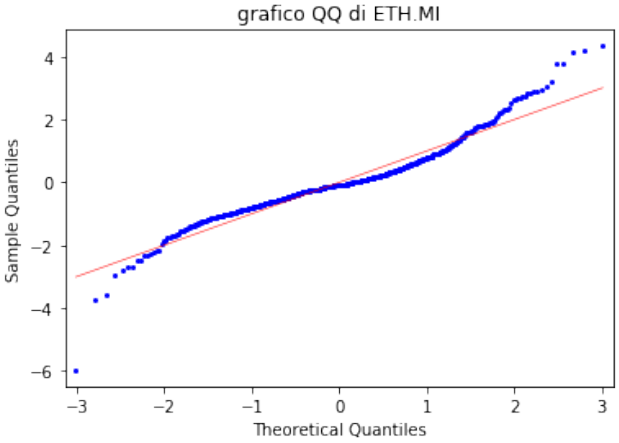
```
plt.hist(lista_rendimenti_stm, bins=75, density=False)
stringa = 'distribuzione rendimenti '+ nome_titoli[3]
plt.title(stringa)
plt.show()
Shapiro_Wilk_test= stats.shapiro(lista_rendimenti_stm)
p_value=Shapiro_Wilk_test[1]
print('il p-value è:',p_value)
if p_value <= 0.05:
    print("l'ipotesi nulla di normalità è rifiutata")
else:
    print("l'ipotesi nulla di normalità è accettata")
sm.qqplot(lista_rendimenti_stm, line='s',dist = stats.norm, fit = True)
stringa = 'grafico QQ di '+ nome_titoli[3]
plt.title(stringa)
pylab.show()

plt.hist(lista_rendimenti_fbk, bins=75, density=False)
stringa = 'distribuzione rendimenti '+ nome_titoli[4]
plt.title(stringa)
plt.show()
Shapiro_Wilk_test= stats.shapiro(lista_rendimenti_fbk)
p_value=Shapiro_Wilk_test[1]
print('il p-value è:',p_value)
if p_value <= 0.05:
    print("l'ipotesi nulla di normalità è rifiutata")
else:
    print("l'ipotesi nulla di normalità è accettata")
sm.qqplot(lista_rendimenti_fbk, line='s',dist = stats.norm, fit = True)
stringa = 'grafico QQ di '+ nome_titoli[4]
plt.title(stringa)
pylab.show()
```



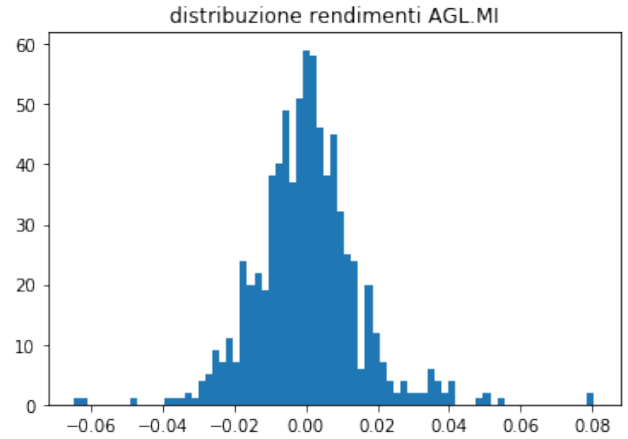


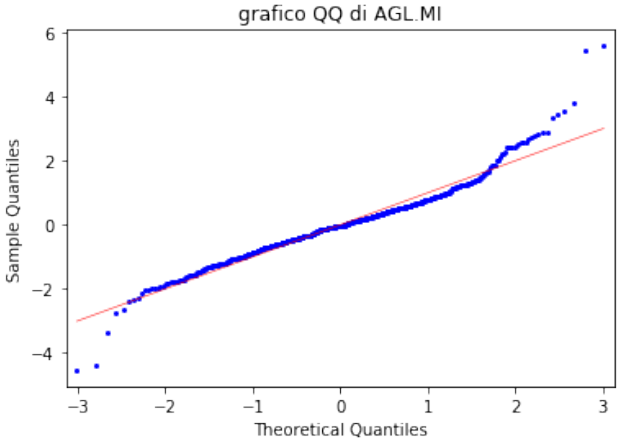

```
il p-value è: 0.00017246045172214508
l'ipotesi nulla di normalità è rifiutata
```



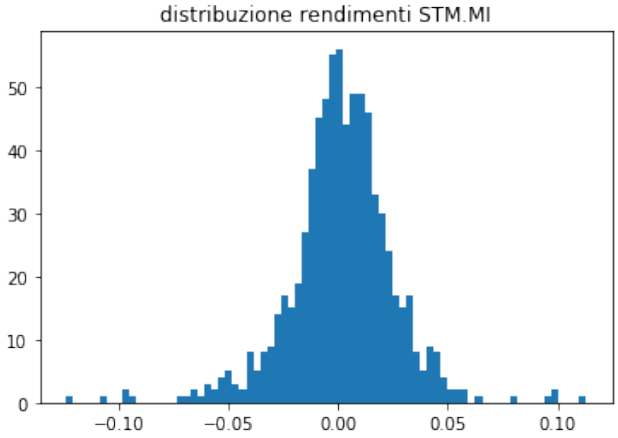


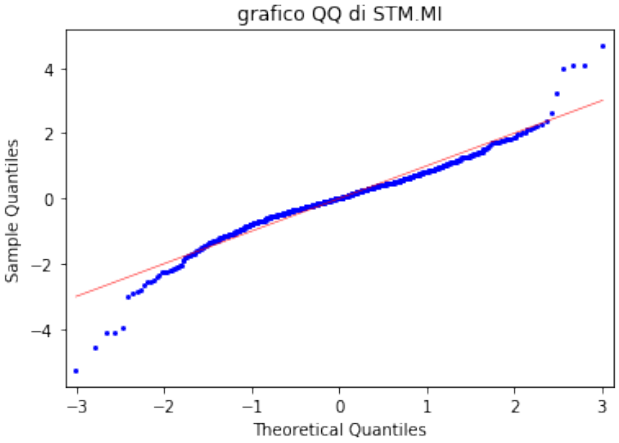

```
il p-value è: 1.524456645807665e-17
l'ipotesi nulla di normalità è rifiutata
```



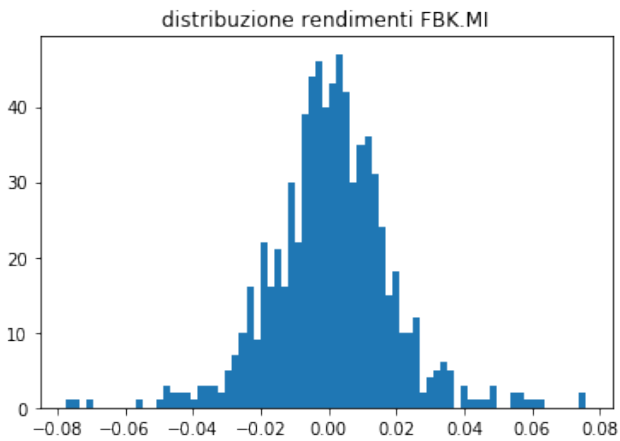


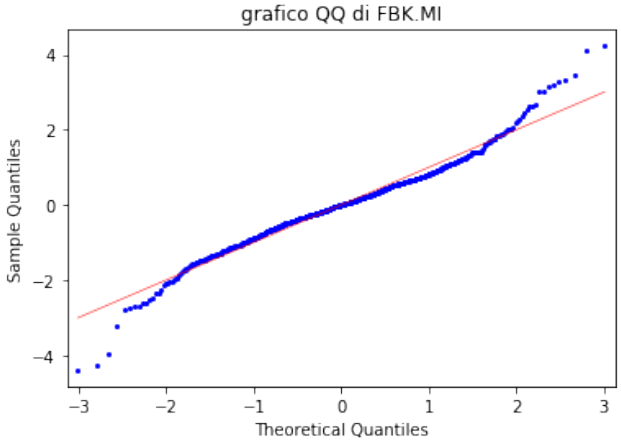

```
il p-value è: 8.558922342562225e-15
l'ipotesi nulla di normalità è rifiutata
```






```
il p-value è: 7.990138024415068e-15
l'ipotesi nulla di normalità è rifiutata
```






```
il p-value è: 5.1779830423370754e-11
l'ipotesi nulla di normalità è rifiutata
```

Analizzare la distribuzione dei rendimenti dei titoli permette di valutare più accuratamente il rischio degli stessi.
Se la distribuzione è normale, la standard deviation è una misura di rischio affidabile, mentre nel caso contrario tale misura subisce delle distorsioni.
Per verificare se le distribuzioni dei rendimenti dei titoli possano essere considerate normali eseguo lo Shapiro-Wilk test attraverso la funzione scipy.stats.shapiro(x) (tale funzione restituisce un vettore con all'interno la statistica test e il p-value).
Questo strumento ha la potenza del test maggiore (probabilità di respingere l'ipotesi nulla quando è falsa[1]) rispetto a tutti gli altri concernenti i test di normalità, e ha risultati affidabili per campioni contenenti osservazioni inferiori a 5000 unità.[2]
Ai fini del test prendo in considerazione il p-value, il quale è definito come il minimo livello di significatività per il quale l'ipotesi nulla può essere respinta[3].
L'ipotesi nulla del test in questione è che i rendimenti siano distribuiti normalmente e la verifico fissando un livello di significatività di 0,05.
Se il p-value è inferiore a 0,05 l'ipotesi nulla è rifiutata e c'è evidenza che i rendimenti dei titoli non provengano da una popolazione normalmente distribuita.
L'esito complessivo dei test effettuato sui rendimenti dei 5 titoli è il rifiuto dell'ipotesi nulla, e quindi i rendimenti dei titoli non provengono da distribuzioni normali.
Al livello grafico, è possibile osservare nella schermata di output a sinistra le distribuzioni dei rendimenti dei titoli e a destra il grafico quantile-quantile (QQ).
Quest'ultimo mette in relazione i quantili della distribuzione dei rendimenti osservati con quelli di una distribuzione normale con stessa media e deviazione standard (tale ridimensionamento della distribuzione normale lo effettuo con il parametro line='s' della funzione statsmodels.qqplot).
La linea rossa rappresenta i valori attesi dei quantili in caso di distribuzione normale, se i valori dei quantili della distribuzione in esame si discostano da essa si manifesta un sospetto di non normalità della distribuzione.

[1]Monti A. C. Introduzione alla statistica, Edizioni Scientifiche, 2008 (2° edizione) p. 341

[2] Razali, N. M. & Wah, Y. B. (2011) Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests, Journal of Statistical Modeling and Analytics, Vol. 2, pp. 25 e 32.

[3] Monti A. C. Introduzione alla statistica, Edizioni Scientifiche, 2008 (2° edizione) p. 337

3.2 Studio della forma delle distribuzioni con la skewness e la curtosi

```
from scipy.stats import kurtosis
from scipy.stats import skew
import scipy.stats as scs
import pandas as pd

kurtosis_eni=kurtosis(lista_rendimenti_eni,fisher=True)
kurtosistest_eni=scs.kurtosistest(lista_rendimenti_eni)
skewness_eni=skew(lista_rendimenti_eni)
skewtest_eni=scs.skewtest(lista_rendimenti_eni)

kurtosis_eth=kurtosis(lista_rendimenti_eth,fisher=True)
kurtosistest_eth=scs.kurtosistest(lista_rendimenti_eth)
skewness_eth=skew(lista_rendimenti_eth)
skewtest_eth=scs.skewtest(lista_rendimenti_eth)

kurtosis_agl=kurtosis(lista_rendimenti_agl,fisher=True)
kurtosistest_agl=scs.kurtosistest(lista_rendimenti_agl)
skewness_agl=skew(lista_rendimenti_agl)
skewtest_agl=scs.skewtest(lista_rendimenti_agl)

kurtosis_stm=kurtosis(lista_rendimenti_stm,fisher=True)
kurtosistest_stm=scs.kurtosistest(lista_rendimenti_stm)
skewness_stm=skew(lista_rendimenti_stm)
skewtest_stm=scs.skewtest(lista_rendimenti_stm)

kurtosis_fbk=kurtosis(lista_rendimenti_fbk,fisher=True)
kurtosistest_fbk=scs.kurtosistest(lista_rendimenti_fbk)
skewness_fbk=skew(lista_rendimenti_fbk)
skewtest_fbk=scs.skewtest(lista_rendimenti_fbk)
```

```
statistiche_eni = {'skewness':[skewness_eni,skewtest_eni[0],skewtest_eni[1]],
            'Kurtosis':[kurtosis_eni,kurtosistest_eni[0],kurtosistest_eni[1]]}
statistiche_eni_frame=pd.DataFrame.from_dict(statistiche_eni, orient='index',
            columns=['Value','t-statistic','p-value'])

statistiche_eth = {'skewness':[skewness_eth,skewtest_eth[0],skewtest_eth[1]],
            'Kurtosis':[kurtosis_eth,kurtosistest_eth[0],kurtosistest_eth[1]]}
statistiche_eth_frame=pd.DataFrame.from_dict(statistiche_eth,orient='index',
            columns=['Value','t-statistic','p-value'])

statistiche_agl = {'skewness':[skewness_agl,skewtest_agl[0],skewtest_agl[1]],
            'Kurtosis':[kurtosis_agl,kurtosistest_agl[0],kurtosistest_agl[1]]}
statistiche_agl_frame=pd.DataFrame.from_dict(statistiche_agl,orient='index',
            columns=['Value','t-statistic','p-value'])

statistiche_stm = {'skewness':[skewness_stm,skewtest_stm[0],skewtest_stm[1]],
            'Kurtosis':[kurtosis_stm,kurtosistest_stm[0], kurtosistest_stm[1]]}
statistiche_stm_frame=pd.DataFrame.from_dict(statistiche_stm,orient='index',
            columns=['Value','t-statistic','p-value'])

statistiche_fbk = {'skewness':[skewness_fbk,skewtest_fbk[0],skewtest_fbk[1]],
            'Kurtosis':[kurtosis_fbk,kurtosistest_fbk[0],kurtosistest_fbk[1]]}
statistiche_fbk_frame=pd.DataFrame.from_dict(statistiche_fbk,orient='index',
            columns=['Value','t-statistic','p-value'])
```

```
print('ENI',statistiche_eni_frame)
print('ETH',statistiche_eth_frame)
print('AGL',statistiche_agl_frame)
print('STM',statistiche_stm_frame)
print('FBK',statistiche_fbk_frame)
```

```
ENI              Value  t-statistic   p-value
skewness  0.066691     0.757986  0.448459
Kurtosis  0.704785     3.164668  0.001553
ETH              Value  t-statistic       p-value
skewness  0.394509     4.334313  1.462159e-05
Kurtosis  3.893165     8.632839  5.984745e-18
AGL              Value  t-statistic       p-value
skewness  0.557702     5.943699  2.786603e-09
Kurtosis  3.849766     8.591257  8.602322e-18
STM              Value  t-statistic       p-value
skewness -0.334270    -3.706975  2.097493e-04
Kurtosis  3.686247     8.430347  3.446420e-17
FBK              Value  t-statistic       p-value
skewness  0.041039     0.466738  6.406872e-01
Kurtosis  2.360047     6.803617  1.020245e-11
```

La deviazione dalla normalità di una distribuzione dei rendimenti può essere quantificata tramite due indici che ne descrivono la forma: la skewness e la curtosi.
La skewness è un indice che indica l'ammontare e la direzione dell'asimmetria di una distribuzione rispetto a quella Gaussiana (la quale ha skewness uguale a 0).
Se l'indice di skewness è maggiore di 0, la distribuzione è asimmetrica verso destra (la coda più lunga e piatta è posizionata a destra), e da ciò ne deriva che la standard deviation sovrastima il rischio, poiché i rendimenti molto elevati (che non sono fonte di preoccupazione per gli investitori) ne aumentano la volatilità.[4]
Nel caso opposto, e di maggiore importanza per gli investitori, se l'indice di skewness è negativo, e quindi la distribuzione asimmetrica verso sinistra, la standard deviation sottostima il rischio.[5]
La curtosi misura lo spessore o la sottigliezza delle code di una distribuzione rispetto alla distribuzione Gaussiana, a tale misura in accordo con la definizione di Fisher viene sottratto 3, in modo da risultare 0 in caso di distribuzione normale.
Una curtosi maggiore di 0 indica che la standard deviation sottostima la probabilità di avere rendimenti molto bassi o molto alti.[6]
Dal test delle ipotesi emerge che per tutte le distribuzioni dei rendimenti la curtosi sia significativamente diversa da 0 (p-value<0,05), lo stesso vale per la skewness ad eccezione del titolo ENI; quindi anche in questo caso l'ipotesi nulla di normalità delle distribuzioni è rigettata.

[4] Bodie, kane, Marcus. Investments, McGraw-Hill, 2014 (10° edizione) pag. 138

[5] Bodie, kane, Marcus. Investments, McGraw-Hill, 2014 (10° edizione) pag. 138

[6] Bodie, kane, Marcus. Investments, McGraw-Hill, 2014 (10° edizione) pag. 138

In particolare, tutte le distribuzioni dei rendimenti hanno una curtosi significativamente maggiore di 0, ovvero sono di natura Leptocurtica (ciò significa che le loro code tendono asintoticamente a zero più lentamente della Gaussiana, e quindi producono più outliers rispetto ad essa.)[7].
Per quanto riguarda l'indice di skewness, la distribuzione dell'azione Eni risulta essere assimilabile a quella normale avendo un p-value maggiore di 0,05, la distribuzione dell'azione STM presenta un valore significativamente negativo, quindi è asimmetrica verso sinistra, mentre le altre distribuzioni presentano un valore significativamente positivo, quindi possiamo concludere che esse siano asimmetriche verso destra.
Da ciò si deduce che la deviazione standard può non rappresentare in modo veritiero il rischio dei titoli, quindi è opportuno introdurre un'ulteriore misura di rischio che permetta di svincolarsi dall'assunzione di normalità della distribuzione e quantificare la vulnerabilità a rendimenti molto negativi: il value at risk (adottando il metodo della simulazione storica).

[7] Grami A. Probability, Random Variables, Statistics, and Random Processes, Wiley, 2019 paragrafo 8.5

CAPITOLO 4: IL VALUE AT RISK DEI TITOLI

```
import pandas as pd
from tabulate import tabulate
var_90=pd.DataFrame(lista_rendimenti_eni)[0].quantile(0.1)
var_95=pd.DataFrame(lista_rendimenti_eni)[0].quantile(0.05)
var_99=pd.DataFrame(lista_rendimenti_eni)[0].quantile(0.01)
print (tabulate([['90%',var_90],['95%',var_95],['99%',var_99]],
        headers=['livello di confidenza','value at risk ENI']))

var_90=pd.DataFrame(lista_rendimenti_eth)[0].quantile(0.1)
var_95=pd.DataFrame(lista_rendimenti_eth)[0].quantile(0.05)
var_99=pd.DataFrame(lista_rendimenti_eth)[0].quantile(0.01)
print (tabulate([['90%',var_90],['95%',var_95],['99%',var_99]],
        headers=['livello di confidenza','value at risk ETH']))

var_90=pd.DataFrame(lista_rendimenti_agl)[0].quantile(0.1)
var_95=pd.DataFrame(lista_rendimenti_agl)[0].quantile(0.05)
var_99=pd.DataFrame(lista_rendimenti_agl)[0].quantile(0.01)
print (tabulate([['90%',var_90],['95%',var_95],['99%',var_99]],
        headers=['livello di confidenza','value at risk AGL']))

var_90=pd.DataFrame(lista_rendimenti_stm)[0].quantile(0.1)
var_95=pd.DataFrame(lista_rendimenti_stm)[0].quantile(0.05)
var_99=pd.DataFrame(lista_rendimenti_stm)[0].quantile(0.01)
print (tabulate([['90%',var_90],['95%',var_95],['99%',var_99]],
        headers=['livello di confidenza','value at risk STM']))

var_90=pd.DataFrame(lista_rendimenti_fbk)[0].quantile(0.1)
var_95=pd.DataFrame(lista_rendimenti_fbk)[0].quantile(0.05)
var_99=pd.DataFrame(lista_rendimenti_fbk)[0].quantile(0.01)
print (tabulate([['90%',var_90],['95%',var_95],['99%',var_99]],
        headers=['livello di confidenza','value at risk FBK']))
```

```
livello di confidenza      value at risk ENI
-----------------------  -------------------
90%                              -0.0148685
95%                              -0.0189314
99%                              -0.0261498
livello di confidenza      value at risk ETH
-----------------------  -------------------
90%                              -0.0238429
95%                              -0.0317963
99%                              -0.0601905
livello di confidenza      value at risk AGL
-----------------------  -------------------
90%                              -0.0160235
95%                              -0.0210954
99%                              -0.0315147
livello di confidenza      value at risk STM
-----------------------  -------------------
90%                              -0.0255207
95%                              -0.0370206
99%                              -0.0654826
livello di confidenza      value at risk FBK
-----------------------  -------------------
90%                              -0.0197079
95%                              -0.0260603
99%                              -0.0461455
```

Il Value at Risk (VaR) rappresenta la massima perdita potenziale di un asset o di un portafoglio dato un determinato orizzonte temporale e uno specifico intervallo di confidenza.[8]
Per calcolare tale valore adotto il metodo della simulazione storica (non parametrico) il quale opera direttamente sui dati storici e non in base a ipotesi sulla distribuzione dei rendimenti.
Nonostante tale metodologia permetta di superare l'assunzione di normalità delle distribuzioni dei rendimenti, la sua validità dipende da un'ipotesi molto forte, ovvero che i dati storici contengano tutti i possibili scenari che si possono verificare nel futuro.
Inoltre, se c'è un trend crescente (decrescente) di volatilità durante l'intervallo temporale del campione, il VaR risulterà sottostimato (sovrastimato), poiché tutti i rendimenti sono ponderati nello stesso modo.[9]
Per calcolare il VaR utilizzo la funzione pandas.quantile, con la quale ottengo il valore del rendimento corrispondente a un determinato quantile di una distribuzione, ovvero sottraendo a uno il quantile, a un determinato livello di confidenza (ad esempio a un quantile del 10% corrisponde un livello di confidenza del 90%).
Posso affermare, ad esempio per l'azione ENI, che la massima perdita potenziale giornaliera con una probabilità del 90% equivale a un rendimento di -0.0149.

[8] A. Damodaran 2011
[9] Kim D., Francis J. C. Modern Portfolio Theory: Foundations, Analysis, and New Developments, John Wiley & Sons, 2013 paragrafo 11.5.2

CAPITOLO 5: IL VALORE ATTESO E LA VARIANZA

5.1 Calcolo del valore atteso dei titoli

```
vettore_rendimenti_medi = []
rendimenti_medi=lista_rendimenti_eni.mean()
vettore_rendimenti_medi.append([rendimenti_medi])

rendimenti_medi=lista_rendimenti_eth.mean()
vettore_rendimenti_medi.append([rendimenti_medi])

rendimenti_medi=lista_rendimenti_agl.mean()
vettore_rendimenti_medi.append([rendimenti_medi])

rendimenti_medi=lista_rendimenti_stm.mean()
vettore_rendimenti_medi.append([rendimenti_medi])

rendimenti_medi=lista_rendimenti_fbk.mean()
vettore_rendimenti_medi.append([rendimenti_medi])

print (vettore_rendimenti_medi)
rendimento_massimo=vettore_rendimenti_medi[0]

for i in vettore_rendimenti_medi:
    if i>rendimento_massimo:
        rendimento_massimo=i[0]
rendimento_minimo=vettore_rendimenti_medi[0]

for i in vettore_rendimenti_medi:
    if i<rendimento_minimo:
        rendimento_minimo=i[0]
print (rendimento_massimo,rendimento_minimo)
```

```
[[0.00024720780138471954], [0.0017138388709156298], [0.0003080882958049988], [0.001494413846586
2627], [0.0007994921412453481]]
0.0017138388709156298 [0.00024720780138471954]
```

Il valore atteso della variabile casuale rendimento è ottenuto facendo la media aritmetica dei rendimenti storici assegnando a ognuno di essi un uguale probabilità di verificarsi, ovvero:

$$E[\tilde{\delta}] = \sum_{t=1}^{n} p(t)\,\delta(t) = \frac{1}{762}\sum_{t=1}^{762} \delta(t)$$

Dove:
n è il numero di rendimenti (nel caso in esame pari a 762);
$p(t)$ è la probabilità assegnata all'osservazione t.
Definisco ora la variabile "vettore_rendimenti_medi" come un vettore vuoto, per poi riempirlo successivamente con i valori attesi dei rendimenti dei 5 titoli.
Tramite la funzione numpy.mean calcolo le medie aritmetiche dei rendimenti dei titoli, e poi attraverso la funzione numpy.append inserisco i valori così ottenuti nel "vettore_rendimenti_medi".
Poi utilizzando due cicli di for trovo il valore massimo e minimo del "vettore_rendimenti_medi", ciò sarà utile successivamente per stabilire la lunghezza dell'asse "E[x]" del grafico valore atteso vs varianza.

5.2 Calcolo della varianza come misura di rischio

```
vettore_varianze=[]
varianze=lista_rendimenti_eni.var()
vettore_varianze.append([varianze])

varianze=lista_rendimenti_eth.var()
vettore_varianze.append([varianze])

varianze=lista_rendimenti_agl.var()
vettore_varianze.append([varianze])

varianze=lista_rendimenti_stm.var()
vettore_varianze.append([varianze])

varianze=lista_rendimenti_fbk.var()
vettore_varianze.append([varianze])

print (vettore_varianze)
varianza_massima=vettore_varianze[0][0]

for i in vettore_varianze:
    if i>varianza_massima:
        varianza_massima=i[0]
varianza_minima=vettore_varianze[0][0]

for i in vettore_varianze:
    if i<varianza_minima:
        varianza_minima=i
print (varianza_minima, varianza_massima)
```

```
[[0.00012625855597972935], [0.000627116649508125], [0.0002071769479212936], [0.0005682199021482
84], [0.00031597959499342 85]]
0.00012625855597972935 0.000627116649508125
```

H.M. Markowitz nel 1952 propose la varianza come misura del rischio d'investimento per cui: a varianza via via più elevata si associa rischio via via più elevato.[10]
La varianza è il valore atteso del quadrato degli scarti dei valori della variabile casuale dal suo valore atteso:

$$Var\left[\tilde{\delta}_i\right] = \sum_{t=1}^{762} p(t)\left(\delta(t) - \bar{\delta}\right)^2$$

Dove:
$\bar{\delta}$ è il tasso istantaneo di rendimento medio.
In questo blocco di codice faccio sostanzialmente la stessa cosa di quello precedente ma questa volta riferendomi alle varianze.
Utilizzo la funzione numpy.var per ottenere le varianze dei rendimenti dei titoli e poi inserisco tali valori nel "vettore_varianze".

---

[10] Bortot P., Magnani U., Olivieri G., A. Rossi F., Torrigiani. Matematica Finanziaria, Monduzzi, 1998 (2° edizione) p. 443

Il valore massimo e minimo del “vettore_varianze” saranno utili successivamente per stabilire la lunghezza dell’asse “Var(x)” del grafico valore atteso vs varianza.

5.3 Una tabella di sintesi dei risultati

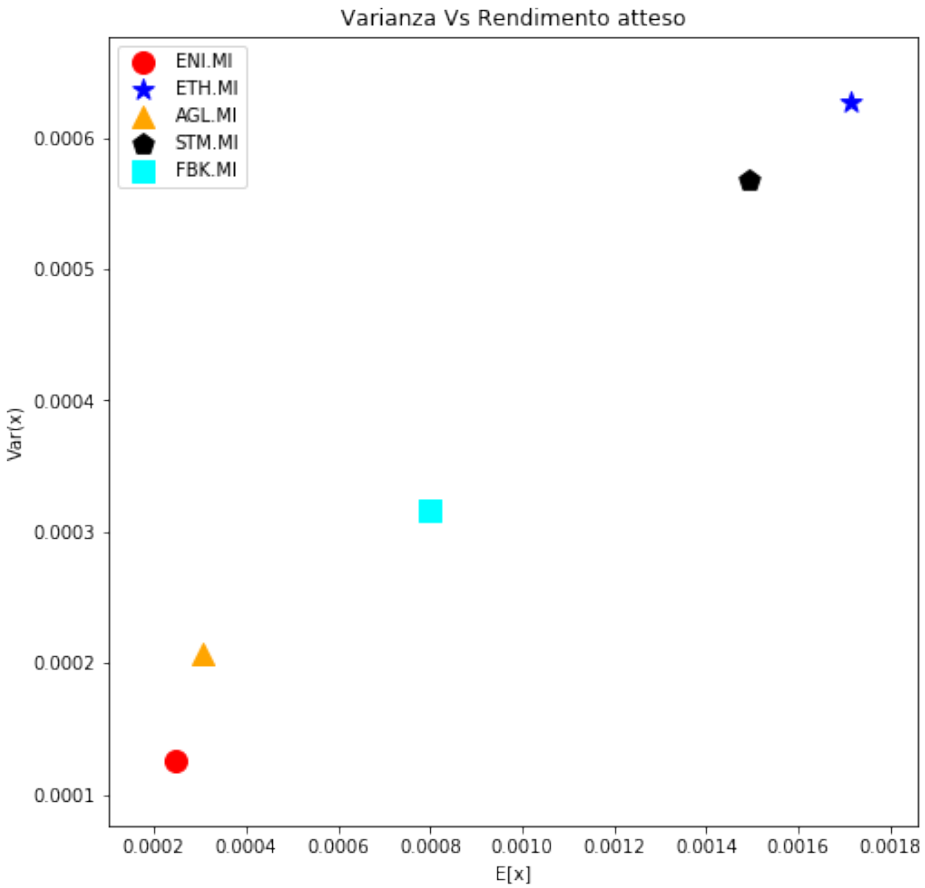
```
import pandas as pd
momenti={nome_titoli[0]:[vettore_rendimenti_medi[0],vettore_varianze[0]],
         nome_titoli[1]:[vettore_rendimenti_medi[1],vettore_varianze[1]],
         nome_titoli[2]:[vettore_rendimenti_medi[2],vettore_varianze[2]],
         nome_titoli[3]:[vettore_rendimenti_medi[3],vettore_varianze[3]],
         nome_titoli[4]:[vettore_rendimenti_medi[4],vettore_varianze[4]]}

momenti_DataFrame=pd.DataFrame.from_dict(momenti,orient='index',
                  columns=['valore atteso','varianza'])
momenti_DataFrame
```

	valore atteso	varianza
ENI.MI	[0.00024720780138471954]	[0.00012625855597972935]
ETH.MI	[0.0017138388709156298]	[0.000627116649508125]
AGL.MI	[0.0003080882958049988]	[0.0002071769479212936]
STM.MI	[0.0014944138465862627]	[0.000568219902148284]
FBK.MI	[0.0007994921412453481]	[0.0003159795949934285]

Questo DataFrame esprime una sintesi raggruppando il valore atteso e la varianza di ogni titolo.

5.4 Grafico valore atteso versus varianza

```
fig, ax = plt.subplots(figsize=(8, 8))
plt.rcParams['lines.markersize'] = 12
extension_x=(rendimento_massimo-rendimento_minimo)*0.1
extension_y=(varianza_massima-varianza_minima)*0.1
colori=['red','blue','orange','black','cyan']
simboli=['o','*','^','p','s']
ax.set_xlim(rendimento_minimo-extension_x,rendimento_massimo+extension_x)
ax.set_ylim(varianza_minima-extension_y,varianza_massima+extension_y)
ctr=0
while ctr<5:
    ax.scatter(vettore_rendimenti_medi[ctr],vettore_varianze[ctr], c=colori[ctr],
               marker=simboli[ctr], label=nome_titoli[ctr])
    ctr=ctr+1
ax.set_xlabel('E[x]')
ax.set_ylabel('Var(x)')
ax.legend()
ax.set_title('Varianza Vs Rendimento atteso')
plt.show()
```



Il grafico evidenzia una relazione lineare crescente tra valore atteso e varianza dei 5 titoli rappresentati.
Con le variabili “extension_x” e “extension_y” definisco dei margini di sicurezza affinché gli assi del grafico siano sufficientemente capienti da contenere tutti i punti e poi con le funzioni set_xlim e set_ylim fisso i valori per gli estremi degli assi cartesiani.

CAPITOLO 6: LA COVARIANZA E L'INDICE DI CORRELAZIONE

6.1 La matrice delle covarianze

```
import pandas as pd
data = {nome_titoli[0]: [0, 0, 0, 0, 0],
        nome_titoli[1]: [0, 0, 0, 0, 0],
        nome_titoli[2]: [0, 0, 0, 0, 0],
        nome_titoli[3]: [0, 0, 0, 0, 0],
        nome_titoli[4]: [0, 0, 0, 0, 0]}
tab_cov=pd.DataFrame.from_dict(data, orient='index' ,columns=[nome_titoli[0],
                nome_titoli[1], nome_titoli[2],nome_titoli[3],nome_titoli[4]])
import numpy as np2
x1=np2.array([lista_rendimenti_eni,lista_rendimenti_eth])
print (x1.shape)
tab=np2.cov(x1)
print (tab)
tab_cov.iloc[0,0]=tab[0,0]
tab_cov.iloc[1,1]=tab[1,1]
tab_cov.iloc[0,1]=tab[0,1]
tab_cov.iloc[1,0]=tab[1,0]

x1=np2.array([lista_rendimenti_eni,lista_rendimenti_agl])
tab=np2.cov(x1)
tab_cov.iloc[2,2]=tab[1,1]
tab_cov.iloc[0,2]=tab[0,1]
tab_cov.iloc[2,0]=tab[1,0]

x1=np2.array([lista_rendimenti_eni,lista_rendimenti_stm])
tab=np2.cov(x1)
tab_cov.iloc[3,3]=tab[1,1]
tab_cov.iloc[0,3]=tab[0,1]
tab_cov.iloc[3,0]=tab[1,0]

x1=np2.array([lista_rendimenti_eni,lista_rendimenti_fbk])
tab=np2.cov(x1)
tab_cov.iloc[4,4]=tab[1,1]
tab_cov.iloc[0,4]=tab[0,1]
tab_cov.iloc[4,0]=tab[1,0]

x1=np2.array([lista_rendimenti_eth,lista_rendimenti_agl])
tab=np2.cov(x1)
tab_cov.iloc[1,2]=tab[0,1]
tab_cov.iloc[2,1]=tab[1,0]

x1=np2.array([lista_rendimenti_eth,lista_rendimenti_stm])
tab=np2.cov(x1)
tab_cov.iloc[1,3]=tab[0,1]
tab_cov.iloc[3,1]=tab[1,0]

x1=np2.array([lista_rendimenti_eth,lista_rendimenti_fbk])
tab=np2.cov(x1)
tab_cov.iloc[1,4]=tab[0,1]
tab_cov.iloc[4,1]=tab[1,0]
```

```
x1=np2.array([lista_rendimenti_agl,lista_rendimenti_stm])
tab=np2.cov(x1)
tab_cov.iloc[2,3]=tab[0,1]
tab_cov.iloc[3,2]=tab[1,0]

x1=np2.array([lista_rendimenti_agl,lista_rendimenti_fbk])
tab=np2.cov(x1)
tab_cov.iloc[2,4]=tab[0,1]
tab_cov.iloc[4,2]=tab[1,0]

x1=np2.array([lista_rendimenti_stm,lista_rendimenti_fbk])
tab=np2.cov(x1)
tab_cov.iloc[3,4]=tab[0,1]
tab_cov.iloc[4,3]=tab[1,0]
tab_cov

import math as mat
tab_corr=tab_cov.copy()
i=0
j=0
while i<5:
    j=0
    while j<5:

        prodotto_varianze=tab_cov.iloc[i,i]*tab_cov.iloc[j,j]
        tab_corr.iloc[i,j]=tab_cov.iloc[i,j]/mat.sqrt(prodotto_varianze)
        j=j+1
    i=i+1

tab_cov.style.set_caption('<b><i>matrice covarianze')
```

```
(2, 762)
[[1.26424467e-04 6.17487410e-05]
 [6.17487410e-05 6.27940719e-04]]
```

matrice covarianze

	ENI.MI	ETH.MI	AGL.MI	STM.MI	FBK.MI
ENI.MI	0.000126424	6.17487e-05	2.64712e-05	6.88064e-05	9.01293e-05
ETH.MI	6.17487e-05	0.000627941	3.73974e-05	0.000119343	8.64846e-05
AGL.MI	2.64712e-05	3.73974e-05	0.000207449	7.05847e-05	5.99885e-05
STM.MI	6.88064e-05	0.000119343	7.05847e-05	0.000568967	0.000123519
FBK.MI	9.01293e-05	8.64846e-05	5.99885e-05	0.000123519	0.000316395

La covarianza è il valore atteso del prodotto degli scarti dei rendimenti di due titoli dai loro valori medi.
Se vi è maggiore probabilità di concordanza nel segno degli scarti la covarianza è positiva, altrimenti se vi è discordanza la covarianza è negativa.
Se i rendimenti manifestano una dipendenza lineare, la covarianza dice se tale relazione è crescente o decrescente.

Se i rendimenti sono indipendenti fra di loro la covarianza è nulla, tuttavia non è necessariamente vero il contrario (una covarianza nulla non implica che i rendimenti siano indipendenti).
Infatti, se la covarianza è nulla i rendimenti sono incorrelati, ovvero non hanno un legame lineare (potrebbero esserci forme di dipendenza non lineare).[11]
La covarianza tra il titolo i e j è pari a:

$$Cov[\tilde{\delta}_i, \tilde{\delta}_j] = \sum_{t=1}^{762} p(t)(\delta_i(t) - \bar{\delta}_i)(\delta_j(t) - \bar{\delta}_j)$$

Definisco la variabile "data" come un dizionario i cui valori sono sequenze di zeri e le cui chiavi i nomi dei titoli.
Poi creo un DataFrame identificato dalla variabile "tab_cov" le cui etichette di riga e di colonna sono rappresentate dai nomi dei titoli e composto da valori tutti pari a 0 (i quali saranno sostituiti con i valori corretti).
La variabile "x1" indica la matrice numpy composta dai rendimenti di due titoli, e nell'output è riportata la sua dimensione, ovvero (2, 762).
Attraverso la funzione numpy.cov ottengo la matrice delle covarianze riferita ai rendimenti di due titoli, e la indico con la variabile "tab".
Nell'output ho stampato per fini espositivi la matrice di covarianze riferite ai rendimenti dei titoli ENI e ETH; nella diagonale principale sono indicate le varianze, mentre in quella secondaria la covarianza.
Essere a conoscenza delle posizioni di tali valori è fondamentale per riempire correttamente la matrice di covarianze di tutti i titoli.
Con la funzione DataFrame.iloc specifico tramite indicizzazione la posizione del DataFrame "tab_cov" da riempire con un determinato valore della matrice di covarianze di due titoli.
Definisco con la variabile "tab_corr" la matrice degli indici di correlazione, e per comodità la pongo uguale temporaneamente alla matrice delle covarianze tramite la funzione DataFrame.copy (tale funzione non altera la matrice delle covarianze una volta sostituiti i valori), per poi modificarne i valori tramite due cicli di while inserendo in tal modo i valori corretti.

[11] Monti A. C. Introduzione alla statistica, Edizioni Scientifiche, 2008 (2° edizione) pp. 161-163

6.2 La matrice degli indici di correlazione

```
tab_corr.style.set_caption('<b><i>matrice indici di correlazione')
```

matrice indici di correlazione

	ENI.MI	ETH.MI	AGL.MI	STM.MI	FBK.MI
ENI.MI	1	0.219156	0.163456	0.256549	0.450646
ETH.MI	0.219156	1	0.103616	0.199662	0.194028
AGL.MI	0.163456	0.103616	1	0.205452	0.234152
STM.MI	0.256549	0.199662	0.205452	1	0.291123
FBK.MI	0.450646	0.194028	0.234152	0.291123	1

Il coefficiente di correlazione è un indice di intensità del legame lineare tra due rendimenti, esso si ottiene dividendo la covarianza per il prodotto delle due standard deviation dei titoli, è quindi una misura standardizzata; inoltre può assumere valori tra -1 e 1.
Quanto più ρ_{ij} è in valore assoluto vicino all'unità, tanto più forte è l'intensità del legame lineare fra i rendimenti.
Mentre, quanto più ρ_{ij} si avvicina allo zero, tanto più debole è l'intensità del legame lineare.
Il coefficiente di correlazione lineare, ρ_{ij}, è[12]

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

La matrice degli indici di correlazione è un utile strumento nella fase di costruzione del portafoglio, poiché permette di constatare il beneficio dell'effetto diversificazione.
Infatti, se i valori all'interno della matrice sono minori di 1 (a esclusione della diagonale principale che si riferisce a un solo titolo) i titoli non sono perfettamente correlati, e quindi la diversificazione ha effetto.
In particolare, più bassi sono i coefficienti di correlazione all'interno della matrice, e più ha effetto la diversificazione in termini di riduzione del rischio di portafoglio.

[12] Monti A. C. Introduzione alla statistica, Edizioni Scientifiche, 2008 (2° edizione) pp. 164-167

CAPITOLO 7: IL BETA

7.1 I dati storici dell'indice di mercato

```
import pandas_datareader.data as web
nome_indice='ITLMS.MI'
dati_indice=web.get_data_yahoo(nome_indice[0], data_inizio, data_fine,
                                                    interval='d')
dati_indice.shape
```

```
(756, 6)
```

Ora importo i dati storici dell'indice del mercato azionario FTSE Italia All-Share Index (codice di Borsa: ITLMS.MI).
Lo scopo è quello di calcolare il beta dei 5 titoli con tale indice di mercato.
Tuttavia, si manifesta un problema di compatibilità tra le dimensioni dei DataFrame dei dati storici dei titoli e quella dei dati storici dell'indice di mercato.
Infatti, i DataFrame dei titoli hanno una dimensione di (763, 6), mentre quella dell'indice è (756, 6).

7.2 Matrice delle covarianze tra titoli e indice di mercato

```
import pandas as nuovo_indice
data = {nome_indice: [0, 0, 0, 0, 0,0]}
tab_cov_index=pd.DataFrame.from_dict(data, orient='index' ,columns=[nome_indice,
   nome_titoli[0], nome_titoli[1], nome_titoli[2],nome_titoli[3],nome_titoli[4]])

nuovo_indice, nuovo_titoli = dati_indice.align(dati_eni,join='inner',axis=0)
print ("dimensione DataFrame indice e titoli dopo l'allineamento",
                            nuovo_indice.shape, nuovo_titoli.shape)
quotazioni=nuovo_titoli.values
quotazioni_index=nuovo_indice.values
lista_chiusura_indice=quotazioni_index[:,5]
lista_chiusura_titoli=quotazioni[:,5]
x1= np.empty((2,0), float)

i=0
lunghezza=lista_chiusura_titoli.size
while (i<lunghezza-1):
    rendimento_titolo=np.log(lista_chiusura_titoli[i+1]/lista_chiusura_titoli[i])
    rendimento_indice=np.log(lista_chiusura_indice[i+1]/lista_chiusura_indice[i])
    x1=np.append(x1, [[rendimento_titolo],[rendimento_indice]], axis=1)
    i=i+1

print ('dati presenti=', lunghezza)
print ('dimensione matrice rendimenti', x1.shape)
```

```
tab=np2.cov(x1)
print("matrice covarianze tra ENI e indice",tab)
tab_cov_index.iloc[0,0]=tab[1,1]
tab_cov_index.iloc[0,1]=tab[0,1]

nuovo_indice, nuovo_titoli = dati_indice.align(dati_eth,join='inner',axis=0)
quotazioni=nuovo_titoli.values
quotazioni_index=nuovo_indice.values
lista_chiusura_indice=quotazioni_index[:,5]
x1= np.empty((2,0), float)
lista_chiusura_titoli=quotazioni[:,5]

i=0
lunghezza=lista_chiusura_titoli.size
while (i<lunghezza-1):
    rendimento_titolo=np.log(lista_chiusura_titoli[i+1]/lista_chiusura_titoli[i])
    rendimento_indice=np.log(lista_chiusura_indice[i+1]/lista_chiusura_indice[i])
    x1=np.append(x1, [[rendimento_titolo],[rendimento_indice]], axis=1)
    i=i+1

tab=np2.cov(x1)
tab_cov_index.iloc[0,2]=tab[0,1]

nuovo_indice, nuovo_titoli = dati_indice.align(dati_agl,join='inner',axis=0)
quotazioni=nuovo_titoli.values
quotazioni_index=nuovo_indice.values
lista_chiusura_indice=quotazioni_index[:,5]
x1= np.empty((2,0), float)
lista_chiusura_titoli=quotazioni[:,5]
```

```
i=0
lunghezza=lista_chiusura_titoli.size
while (i<lunghezza-1):
    rendimento_titolo=np.log(lista_chiusura_titoli[i+1]/lista_chiusura_titoli[i])
    rendimento_indice=np.log(lista_chiusura_indice[i+1]/lista_chiusura_indice[i])
    x1=np.append(x1, [[rendimento_titolo],[rendimento_indice]], axis=1)
    i=i+1

tab=np2.cov(x1)
tab_cov_index.iloc[0,3]=tab[0,1]

nuovo_indice, nuovo_titoli = dati_indice.align(dati_stm,join='inner',axis=0)
quotazioni=nuovo_titoli.values
quotazioni_index=nuovo_indice.values
lista_chiusura_indice=quotazioni_index[:,5]
x1= np.empty((2,0), float)
lista_chiusura_titoli=quotazioni[:,5]

i=0
lunghezza=lista_chiusura_titoli.size
while (i<lunghezza-1):
    rendimento_titolo=np.log(lista_chiusura_titoli[i+1]/lista_chiusura_titoli[i])
    rendimento_indice=np.log(lista_chiusura_indice[i+1]/lista_chiusura_indice[i])
    x1=np.append(x1, [[rendimento_titolo],[rendimento_indice]], axis=1)
    i=i+1

tab=np2.cov(x1)
tab_cov_index.iloc[0,4]=tab[0,1]

nuovo_indice, nuovo_titoli = dati_indice.align(dati_fbk,join='inner',axis=0)
quotazioni=nuovo_titoli.values
quotazioni_index=nuovo_indice.values
lista_chiusura_indice=quotazioni_index[:,5]
x1= np.empty((2,0), float)
lista_chiusura_titoli=quotazioni[:,5]

i=0
lunghezza=lista_chiusura_titoli.size
while (i<lunghezza-1):
    rendimento_titolo=np.log(lista_chiusura_titoli[i+1]/lista_chiusura_titoli[i])
    rendimento_indice=np.log(lista_chiusura_indice[i+1]/lista_chiusura_indice[i])
    x1=np.append(x1, [[rendimento_titolo],[rendimento_indice]], axis=1)
    i=i+1

tab=np2.cov(x1)
tab_cov_index.iloc[0,5]=tab[0,1]

tab_cov_index.style.set_caption('<b><i>covarianze tra titoli e indice di mercato')
```

```
dimensione DataFrame indice e titoli dopo l'allineamento (743, 6) (743, 6)
dati presenti= 743
dimensione matrice rendimenti (2, 742)
matrice covarianze tra ENI e indice [[1.31670383e-04 2.17195411e-05]
 [2.17195411e-05 2.34068762e-03]]
```

covarianze tra titoli e indice di mercato

	ITLMS.MI	ENI.MI	ETH.MI	AGL.MI	STM.MI	FBK.MI
ITLMS.MI	0.00234069	2.17195e-05	0.000119624	2.65728e-05	0.000149948	7.31331e-05

Definisco con la variabile “tab_cov_index” il DataFrame che accoglierà i valori delle covarianze tra i titoli e l’indice di mercato.
Per risolvere il problema di compatibilità tra DataFrame sopracitato, utilizzo la funzione pandas.Dataframe.align, la quale permette di allineare due DataFrame rispetto a un determinato asse (scegliendo axis=0 ho effettuato un allineamento per righe).
Il risultato sono due nuovi DataFrame perfettamente compatibili poiché della stessa dimensione, ovvero (743, 6) come riportato nella schermata output.
Definisco con la variabile “x1” la matrice numpy che accoglierà i rendimenti dell’indice di mercato e di un titolo e la riempio adottando un ciclo di while.
Poi calcolo la matrice delle covarianze tra i rendimenti di un titolo e l’indice di mercato e inserisco i valori di interesse tramite indicizzazione nel DataFrame finale (“tab_cov_index”).

7.3 Calcolo dei beta dei titoli

```
beta=tab_cov_index.div(tab_cov_index.iloc[0,0]).drop(columns='ITLMS.MI')*100
beta=beta.rename(index={'ITLMS.MI':'beta'})
beta.style.set_caption('<b><i>beta dei titoli %')
```

beta dei titoli %

	ENI.MI	ETH.MI	AGL.MI	STM.MI	FBK.MI
beta	0.927913	5.11064	1.13526	6.40617	3.12443

Il coefficiente $\beta_i = \frac{Cov(\delta_i, R_M)}{\sigma_M^2}$ è dato dal rapporto tra la covarianza, del rendimento del titolo i-esimo con il rendimento di mercato, ed il rischio di mercato.
Il beta, variazione media del rendimento di un titolo a seguito della variazione del rendimento dell'indice di mercato, è la misura adeguata del rischio sistematico dell'attività con il mercato.[13]
Esso indica quindi quanto il rendimento di un titolo è sensibile ai movimenti di mercato.
Le azioni con un beta maggiore di uno tendono ad amplificare i movimenti del mercato, mentre le azioni con un beta compreso tra 0 e 1 tendono a muoversi nella stessa direzione del mercato, ma con intensità minore.
Tramite la funzione DataFrame.div eseguo la divisione tra la covarianza del titolo con l'indice di mercato, e la varianza dell'indice di mercato stesso per ottenere il beta.
Si può notare come le azioni ETH, AGL, STM e FBK amplifichino i movimenti del mercato avendo un beta maggiore dell'1%, mentre ENI si muove sempre nella stessa direzione del mercato ma con intensità minore.
Si può affermare, ad esempio per l'azione ETH, che se il futuro sarà come il passato, quando il mercato cresce dell'1%, tale azione cresce in media dell'5.11%.

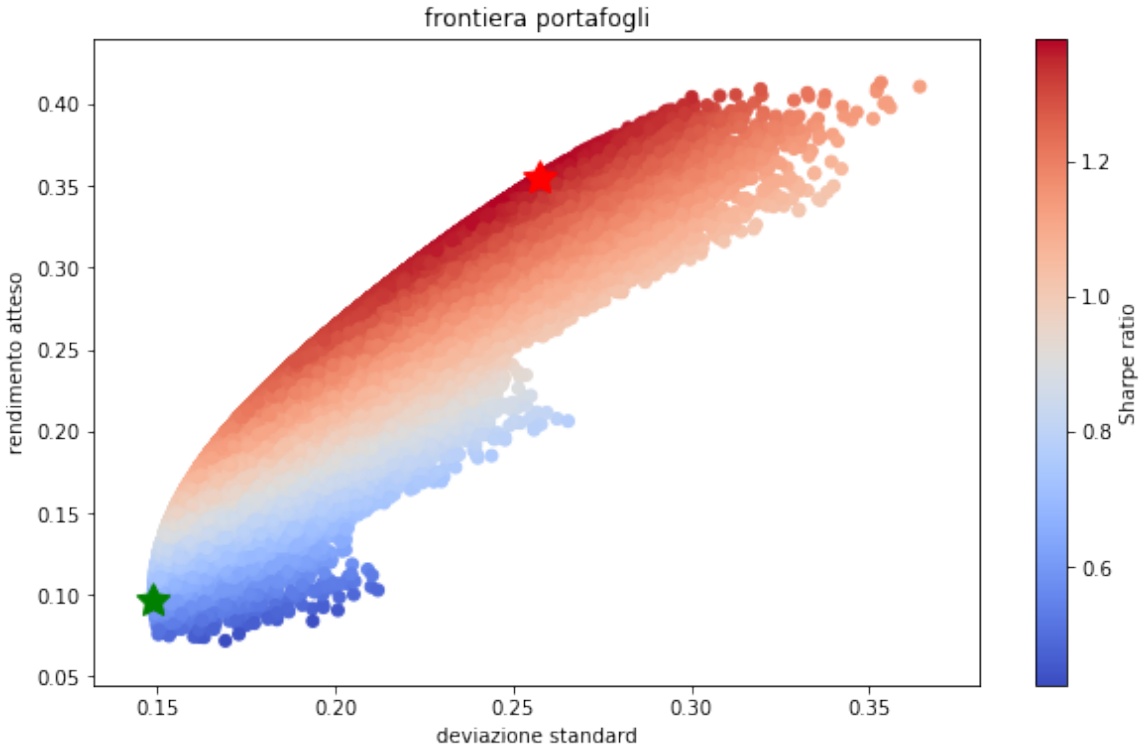
[13] Bortot P., Magnani U., Olivieri G., A. Rossi F., Torrigiani. Matematica Finanziaria, Monduzzi, 1998 (2° edizione) pp. 546-547

CAPITOLO 8: LA FRONTIERA DEI PORTAFOGLI

8.1 Stima della frontiera dei portafogli possibili con la simulazione Monte Carlo

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

rendimenti= np.empty((5,0), float)
rendimenti= np.append(rendimenti,[(lista_rendimenti_eni.mean()*252)])
rendimenti= np.append(rendimenti,[(lista_rendimenti_eth.mean()*252)])
rendimenti= np.append(rendimenti,[(lista_rendimenti_agl.mean()*252)])
rendimenti= np.append(rendimenti,[(lista_rendimenti_stm.mean()*252)])
rendimenti= np.append(rendimenti,[(lista_rendimenti_fbk.mean()*252)])

num_portafogli= 1000000
risultati = np.zeros((4+len(nome_titoli)-1,num_portafogli))
for i in range(num_portafogli):
    pesi= np.array(np.random.random(5))
    pesi /= np.sum(pesi)

    rendimento_portafoglio=np.sum(rendimenti*pesi)
    std_dev_portafoglio=np.sqrt(np.dot(pesi.T,np.dot(tab_cov,pesi)))*np.sqrt(252)

    risultati[0,i] = rendimento_portafoglio
    risultati[1,i] = std_dev_portafoglio
    risultati[2,i] = risultati[0,i] / risultati[1,i]

    for j in range(len(pesi)):
        risultati[j+3,i] = pesi[j]

frame_risultati= pd.DataFrame(risultati.T,columns=['ret','stdev','sharpe',
 nome_titoli[0],nome_titoli[1],nome_titoli[2],nome_titoli[3],nome_titoli[4]])

max_sharpe_port = frame_risultati.iloc[frame_risultati['sharpe'].idxmax()]
min_var_port = frame_risultati.iloc[frame_risultati['stdev'].idxmin()]
```

```
plt.figure (figsize =(10,6))
plt.scatter ( frame_risultati.stdev , frame_risultati.ret ,
             c = frame_risultati.sharpe , marker = '.' , cmap = 'coolwarm' )
plt.xlabel ( 'deviazione standard')
plt.ylabel ( 'rendimento atteso' )
plt.colorbar ( label = 'Sharpe ratio' )
plt.title('frontiera portafogli')
plt.scatter(max_sharpe_port[1],max_sharpe_port[0],marker=(5,1,0),color='r',s=300)
plt.scatter(min_var_port[1],min_var_port[0],marker=(5,1,0),color='g',s=300)
```



Ora lo scopo dell'analisi è tracciare la frontiera dei portafogli possibili, e per farlo adotto la simulazione Monte Carlo (facendo un milione di simulazioni), la quale assegna diversi valori casuali ai pesi dei titoli all'interno del portafoglio creando così diverse composizioni dello stesso, sotto il vincolo che la somma dei pesi sia uguale a uno.
Per fare ciò, utilizzo la funzione numpy.random.random che restituisce un vettore della dimensione desiderata con all'interno i numeri casualmente generati.
Fisso una dimensione pari a 5 per tale vettore poiché altrettanti sono i pesi dei titoli all'interno del portafoglio.
Poi calcolo il rendimento e la deviazione standard dei portafogli in funzione dei pesi di volta in volta casualmente generati.
Il rendimento di un portafoglio è ottenuto come media dei rendimenti dei titoli ponderati per i pesi generati dalla simulazione.

$$\bar{R}_P = \sum_{i=1}^{5} w_i \bar{\delta}_i$$

Mentre la varianza di portafoglio è ottenuta sommando le covarianze all'interno della matrice delle covarianze, e ponderando ogni covarianza per il prodotto dei pesi dei titoli a cui essa si riferisce.

$$\sigma_P^2 = \sum_{i=1}^{5} \sum_{j=1}^{5} w_i w_j \sigma_{ij}$$

Calcolo tale statistica attraverso la funzione numpy.dot, moltiplicando inizialmente il vettore riga dei pesi per la matrice delle covarianze, e successivamente moltiplicando il vettore riga così ottenuto per il vettore colonna (facendo il trasposto del vettore pesi iniziale) degli stessi pesi.
La deviazione standard di portafoglio annualizzata si ottiene facendo la radice quadrata della varianza e moltiplicando tale valore per la radice quadrata di 252, ovvero i giorni di trading.

Inserisco i rendimenti e standard deviation dei portafogli generati all'interno della matrice numpy "risultati".
Dividendo la prima riga per la seconda riga di tale matrice, ovvero i rendimenti e le deviazioni standard, si ottengono le Sharpe Ratio dei singoli portafogli (per semplicità assumo che il tasso di interesse privo di rischio sia pari a 0) e le inserisco nella terza riga della matrice.

$$Sharpe\ ratio = \frac{\bar{R}_P - r_f}{\sigma_P}$$

Poi inserisco i pesi dei titoli di ogni portafoglio generato all'interno della matrice "risultati" a partire dalla sua terza riga tramite un ciclo di for.
Converto la matrice "risultati" in un pandas DataFrame trasposto rispetto ad essa.
Poi tramite le funzioni DataFrame.idxmax e DataFrame.idxmin trovo rispettivamente i valori del portafoglio con la massima Sharpe Ratio e quelli del portafoglio di minima varianza.

8.2 Il portafoglio con la massima Sharpe ratio e il portafoglio di minima varianza

```
results_max = pd.DataFrame(data=max_sharpe_port.rename(''))
results_max.style.set_caption('<b><i>portafoglio massima Sharpe Ratio')
```

portafoglio massima Sharpe Ratio	
ret	0.355215
stdev	0.257129
sharpe	1.38146
ENI.MI	0.000180699
ETH.MI	0.405627
AGL.MI	0.0145192
STM.MI	0.354632
FBK.MI	0.225041

```
results_min = pd.DataFrame(data=min_var_port.rename(''))
results_min.style.set_caption('<b><i>portafoglio minima varianza')
```

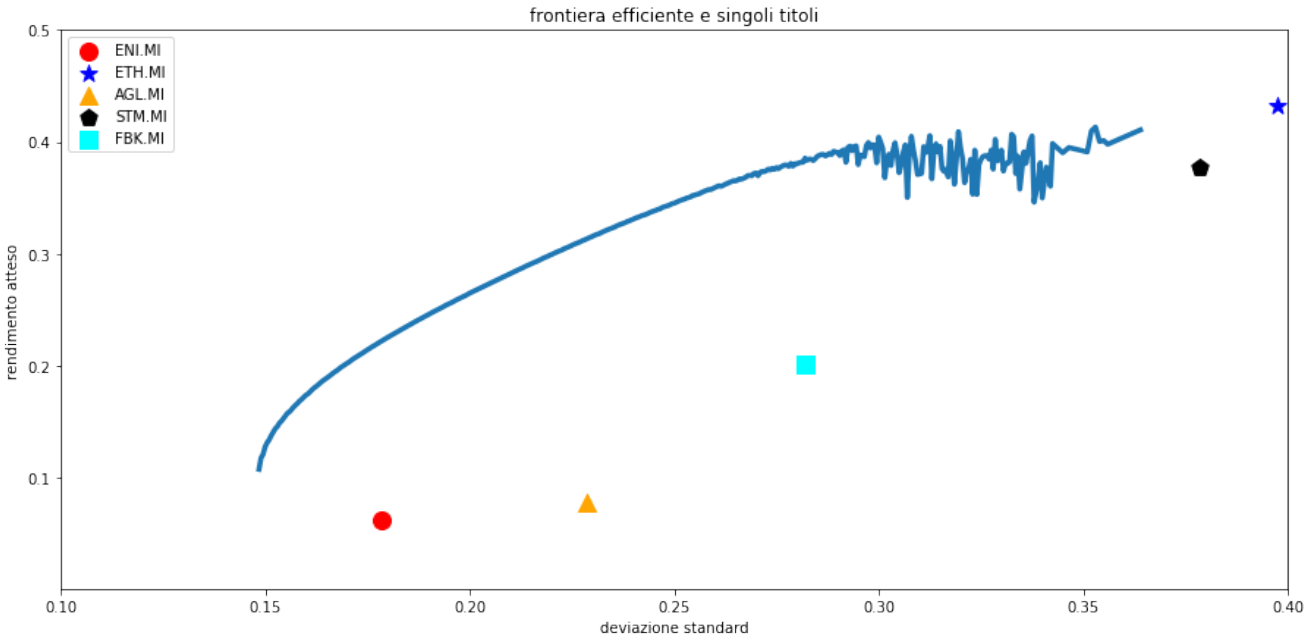
portafoglio minima varianza	
ret	0.0968816
stdev	0.148823
sharpe	0.650985
ENI.MI	0.562007
ETH.MI	0.0495304
AGL.MI	0.337347
STM.MI	0.0227829
FBK.MI	0.0283322

I DataFrame proiettati rappresentano i valori del portafoglio con la massima Sharpe ratio e del portafoglio di minima varianza.
In particolare, le prime tre righe di ogni tabella indicano i valori del rendimento atteso, standard deviation e Sharpe ratio del portafoglio, mentre le righe seguenti indicano i pesi dei titoli di cui il portafoglio è composto.

8.3 Stima della frontiera efficiente

```
front_eff=np.zeros((2,1000))
i=0
while i<1000:
    front_eff[0,i]=i*0.0005
    i=i+1
i=0
while i<1000000:
    dev=risultati[1,i]
    posizione =int(dev/0.0005)

    rendimento=risultati[0,i]
    if rendimento>front_eff[1,posizione]:
        front_eff[1,posizione]=rendimento
    i=i+1
i=0

while i<front_eff.shape[1]:
    if front_eff[1,i]==0:
        front_eff= np.delete(front_eff,i,1)
        i=i-1
    i=i+1

fig,frame= plt.subplots(1, 1,figsize=(15, 7),)
plt.rcParams['lines.linewidth'] = 3.5
plt.rcParams['lines.markersize'] = 12
frame.plot(front_eff[0],front_eff[1])
frame.set_ylim(0.001,0.50)
frame.set_xlim(0.1,0.40)
```

```
ctr=0
while ctr<5:
    y=np.take(vettore_rendimenti_medi,ctr)
    y=y*252
    x=np.take(vettore_varianze,ctr)
    x=x*252
    x=np.sqrt(x)
    frame.scatter(x,y, c=colori[ctr], marker=simboli[ctr],
                label=nome_titoli[ctr])
    ctr=ctr+1
plt.title('frontiera efficiente e singoli titoli')
frame.legend()
plt.xlabel ( 'deviazione standard')
plt.ylabel ( 'rendimento atteso' )
plt.show()
```



Tale grafico mette a confronto la frontiera efficiente stimata attraverso la simulazione Monte Carlo e i singoli titoli.
La frontiera efficiente, o frontiera di minima varianza, è l'insieme di portafogli ottimali che massimizzano il rendimento atteso per un dato livello rischio, oppure minimizzano il rischio atteso dato un livello di rendimento (è la parte di frontiera che giace al di sopra del portafoglio di minima varianza).
Il grafico dimostra come i portafogli composti da un singolo titolo siano inefficienti, in quanto dominati dalla frontiera efficiente con portafogli aventi un maggiore rendimento atteso per un dato livello di rischio (beneficio della diversificazione).
Dal grafico si nota come per elevati valori della standard deviation la frontiera efficiente perda la sua forma teorica, ciò è dovuto al rumore statistico causato dalla bassa concentrazione di portafogli a nord-est della frontiera originati dalla simulazione Monte Carlo (e quindi minore la probabilità che il portafoglio con il massimo rendimento corrisponda al valore teorico di frontiera).
Con la variabile "front_eff" definisco una matrice bidimensionale contenente gli estremi inferiori degli intervalli di deviazione standard e i rendimenti massimi dei portafogli simulati con volatilità all'interno di tali estremi.
Una volta ultimata la ricerca dei massimi valori di rendimento per ogni intervallo di deviazione standard, traccio la curva di frontiera corrispondente a tali valori.

BIBLIOGRAFIA

Brealey R. A., Myers S. C., Sandri S. Principi di Finanza Aziendale, McGraw-Hill, Milano, 2015 (7° edizione)

Bortot P., Magnani U., Olivieri G., A. Rossi F., Torrigiani. Matematica Finanziaria, Monduzzi, 1998 (2° edizione)

Bodie, kane, Marcus. Investments, McGraw-Hill, 2014 (10° edizione)

Monti A. C. Introduzione alla statistica, Edizioni Scientifiche, 2008 (2° edizione)

Hilpisch Y. Python for Finance, Oreilly, 2019 (2° edizione)

VanderPlas J. Python Data Science Handbook, O'Reilly, 2016 (1° edizione)

Razali, N. M. & Wah, Y. B. (2011) Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests, Journal of Statistical Modeling and Analytics, Vol. 2, pp. 21-33.

Grami A. Probability, Random Variables, Statistics, and Random Processes, Wiley, 2019

Kim D., Francis J. C. Modern Portfolio Theory: Foundations, Analysis, and New Developments, John Wiley & Sons, 2013

SITOGRAFIA

https://it.finance.yahoo.com/quote/ENI.MI?p=ENI.MI&.tsrc=fin-srch

https://it.finance.yahoo.com/quote/ETH.MI?p=ETH.MI&.tsrc=fin-srch

https://it.finance.yahoo.com/quote/AGL.MI?p=AGL.MI&.tsrc=fin-srch

https://it.finance.yahoo.com/quote/STM.MI?p=STM.MI&.tsrc=fin-srch

https://it.finance.yahoo.com/quote/FBK.MI?p=FBK.MI&.tsrc=fin-srch

https://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html

https://docs.scipy.org/doc/numpy/contents.html

https://matplotlib.org/3.1.1/users/index.html

https://docs.scipy.org/doc/scipy/reference/stats.html

https://medium.com/@rrfd/testing-for-normality-applications-with-python-6bf06ed646a9

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html

https://www.statsmodels.org/stable/generated/statsmodels.graphics.gofplots.qqplot.html

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skew.html

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skewtest.html

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kurtosis.html

https://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.stats.kurtosistest.html

https://blog.quantinsti.com/calculating-value-at-risk-in-excel-python/

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.quantile.html

https://pythonforfinance.net/2017/01/21/investment-portfolio-optimisation-with-python/

https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.random.html