

Progetto del corso di Business Intelligence per i Servizi Finanziari

Capitolo 1: Sommario dei dati utilizzati

Le sei azioni del mercato americano che ho deciso di analizzare sono **MSFT** (Microsoft) e **TXN** (Texas Instruments) per quanto riguarda il settore tecnologico, **JNJ** (Johnson & Johnson) e **PFE** (Pfizer) per il settore farmaceutico e **MA** (Mastercard) e **JPM** (J.P. Morgan) per il settore finanziario.

Ho deciso di scegliere la prima azienda visto il forte trend verso l'innovazione tecnologica che nel ventunesimo secolo ha portato nelle case di milioni di persone apparecchiature elettronico/informatiche di ogni tipologia: laptop, smartphone, smart TV, console, ecc. ... di cui Microsoft è una principale produttrice; la seconda azienda l'ho scelta viste le enormi sfide in ambito di intelligenza artificiale che l'uomo si è posto: per riuscire ad analizzare ed elaborare l'enorme mole di dati (Big Data) che ogni giorno vengono creati è necessario calcolo computazionale sempre maggiore e circuiti integrati sempre più piccoli di cui Texas Instruments è una tra le più grandi produttrici. La scelta di inserire aziende del settore farmaceutico proviene invece dalla necessità, sempre maggiore, di medicine, causata da due principali fattori: il costante incremento della popolazione mondiale, arrivata oramai a quota 8 miliardi di persone nel 2022 e la maggiore facilità con la quale malattie molto contagiose (quali covid19) si possono diffondere tra le popolazioni dei diversi stati. Infine, la scelta di aziende nell'ambito finanziario è sempre una conseguenza dello sviluppo tecnologico, che sta portato la gente ad un maggior utilizzo di carte di credito e bancomat a scapito del contante, favorito anche dall'enorme diffusione di negozi online.

Per poter analizzare queste azioni utilizzo la libreria *yfinance* e la funzione *download()*, che permette di scaricare da Yahoo Finance l'andamento dei prezzi di una stock compreso di prezzo di apertura, prezzo di chiusura, minimo, massimo, volume e prezzo aggiustato ai dividendi, rispetto ad un determinato arco temporale (in questo caso dal 2012-11-30 al 2022-11-30).

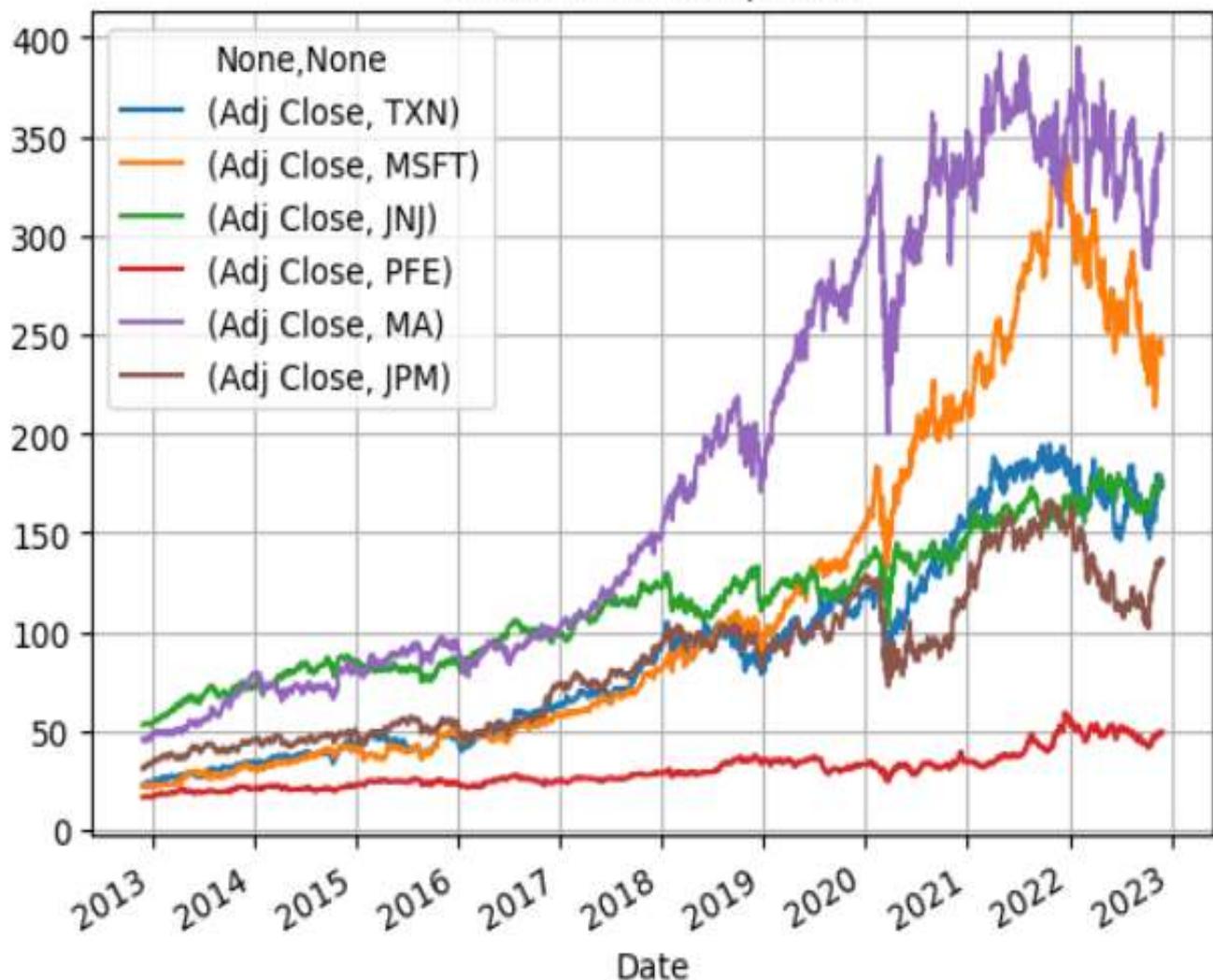
```
TXN = yf.download('TXN', start_date, end_date)
```

Questa libreria mi permette anche di creare un unico *DataFrame* che presenta l'andamento dei prezzi di più azioni nel tempo, e tramite la funzione *plot()* posso visualizzarli in un unico grafico e confrontarli.

```
df = yf.download(['TXN', 'MSFT', 'JNJ', 'PFE', 'MA', 'JPM'], start_date, end_date)
df1 = df[['Adj Close', 'TXN'], ('Adj Close', 'MSFT'), ('Adj Close', 'JNJ'), ('Adj Close', 'PFE'), ('Adj Close', 'MA'), ('Adj C
df1.plot(grid=True, title="Andamento dei prezzi")
```

Per analizzare i dati utilizzo il prezzo aggiustato alla chiusura, cioè il prezzo di chiusura di un'azione modificato dopo aver tenuto conto dei dividendi.

Andamento dei prezzi



Utilizzando la funzione `head()` posso visualizzare le prime 5 righe del *DataFrame*.

Date	TXN	MSFT	JNJ	PFE	MA	JPM
2012-11-30	22.582645	21.950119	53.113270	16.596138	46.064140	31.126062
2012-12-03	22.689936	21.793449	53.052334	16.642570	45.872803	30.921490
2012-12-04	22.689936	21.743973	53.212303	16.695637	45.568325	30.739647
2012-12-05	22.881510	21.991346	53.296066	17.007395	45.057415	31.216991
2012-12-06	22.843195	22.040813	53.357010	16.987490	45.328892	31.421570

Capitolo 2: Statistiche descrittive

Tramite i Data Frame creati calcolo il valore dei rendimenti di ogni azione scelta.

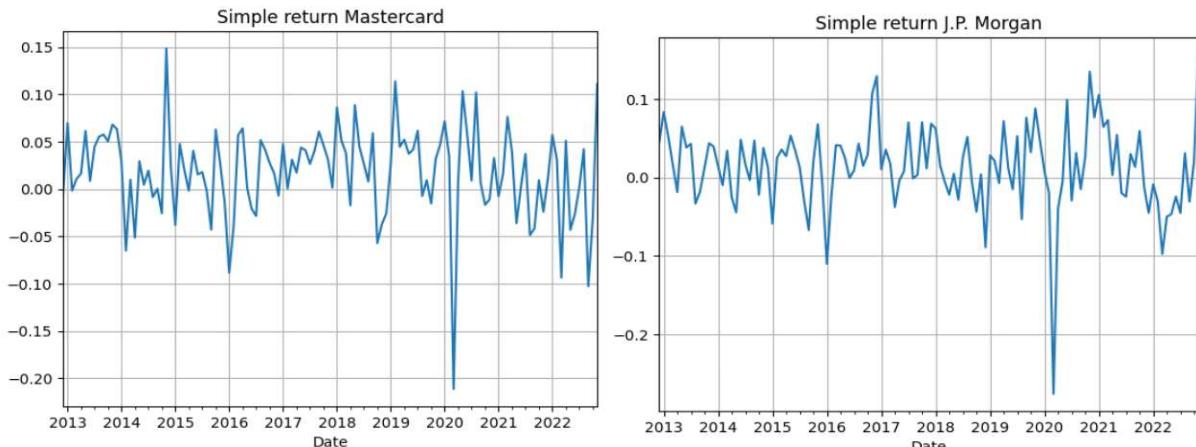
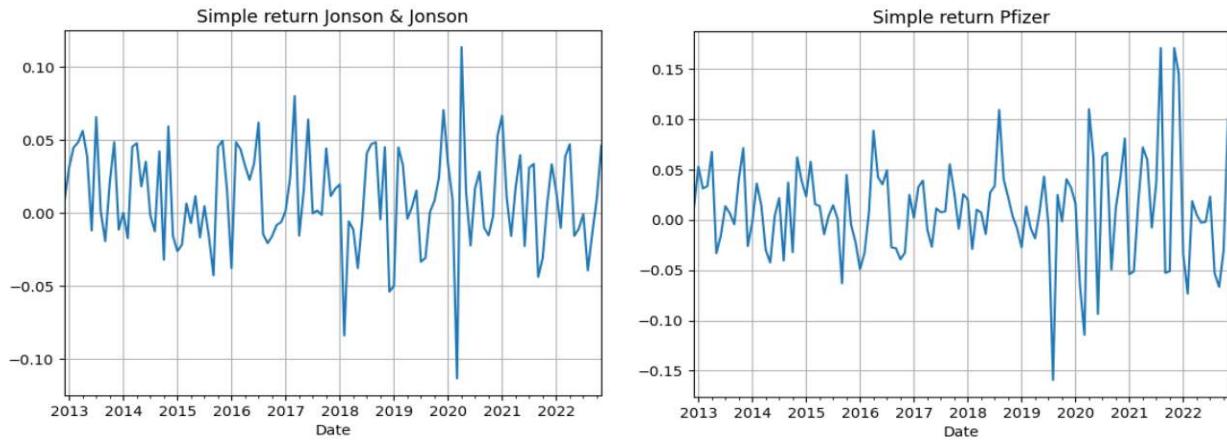
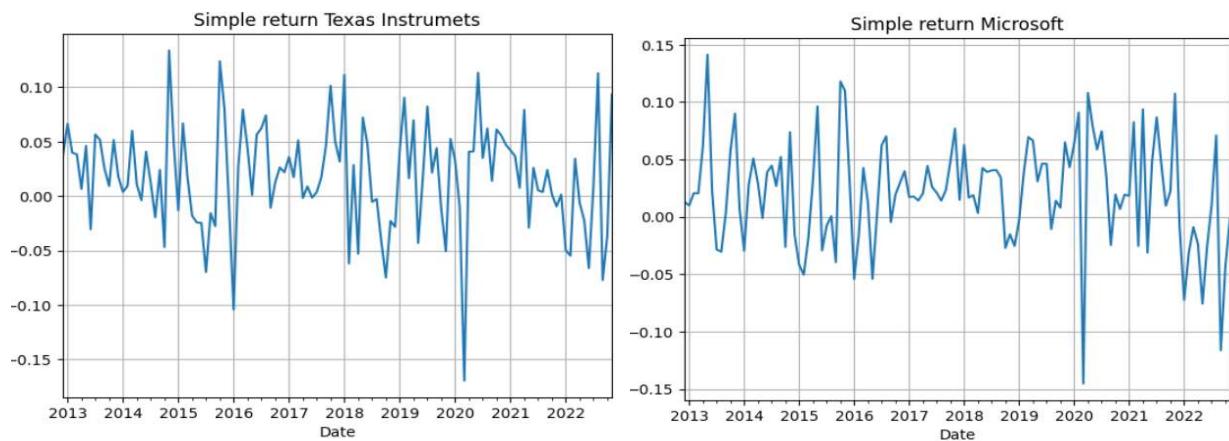
Utilizzando la funzione `resample('M')`, ottengo il prezzo medio mensile di ogni azione (ho scelto di raggruppare i prezzi mensilmente perché i grafici realizzati con i rendimenti giornalieri risultano incomprensibili).

```
TXN_month = TXN.resample('M').mean()
```

Successivamente, attraverso la funzione `pct_change()` (variazione percentuale di un giorno dal precedente) combinata con `dropna()` (escludo i dati nulli) ottengo la serie dei **ritorni semplici netti** di ogni azione e li mostro su dei grafici.

```
TXN_month_return = TXN_month['Adj Close'].pct_change().dropna()
```

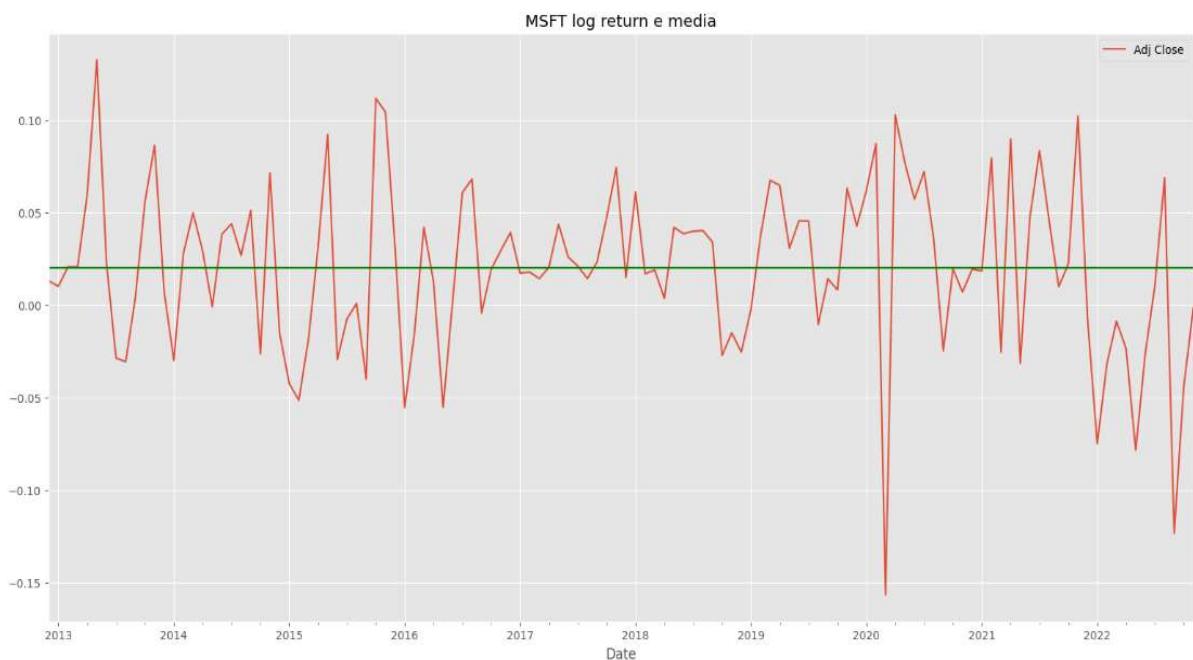
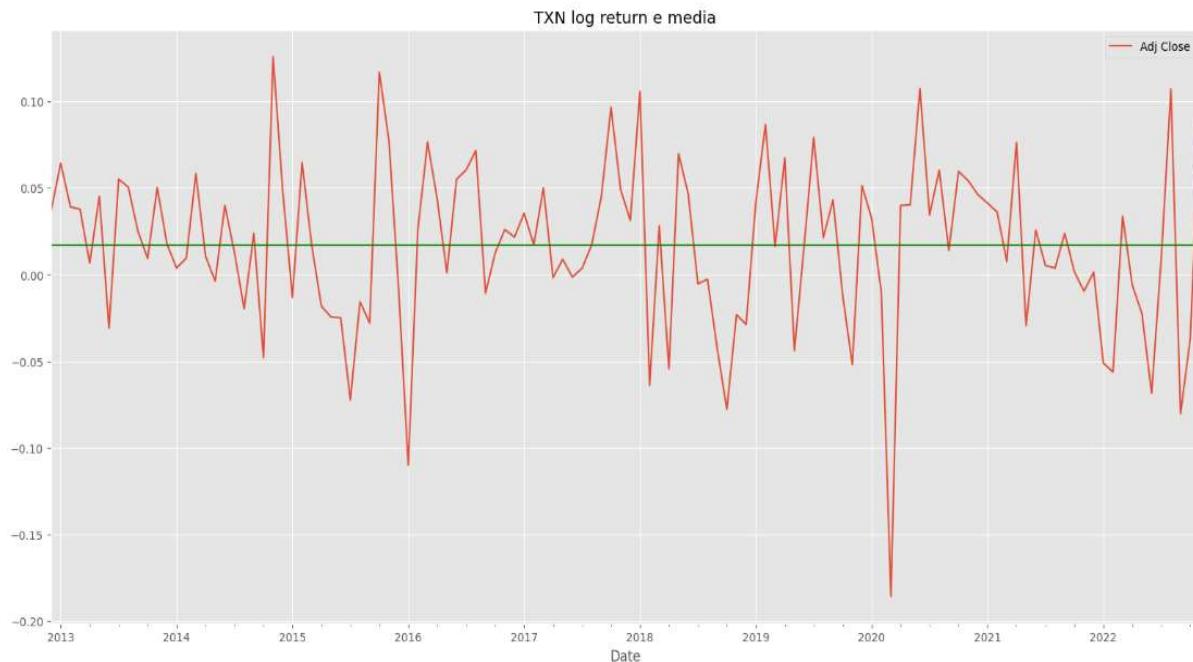
```
TXN_month_return.plot(grid=True, title="Simple return Texas Instrumets")
```

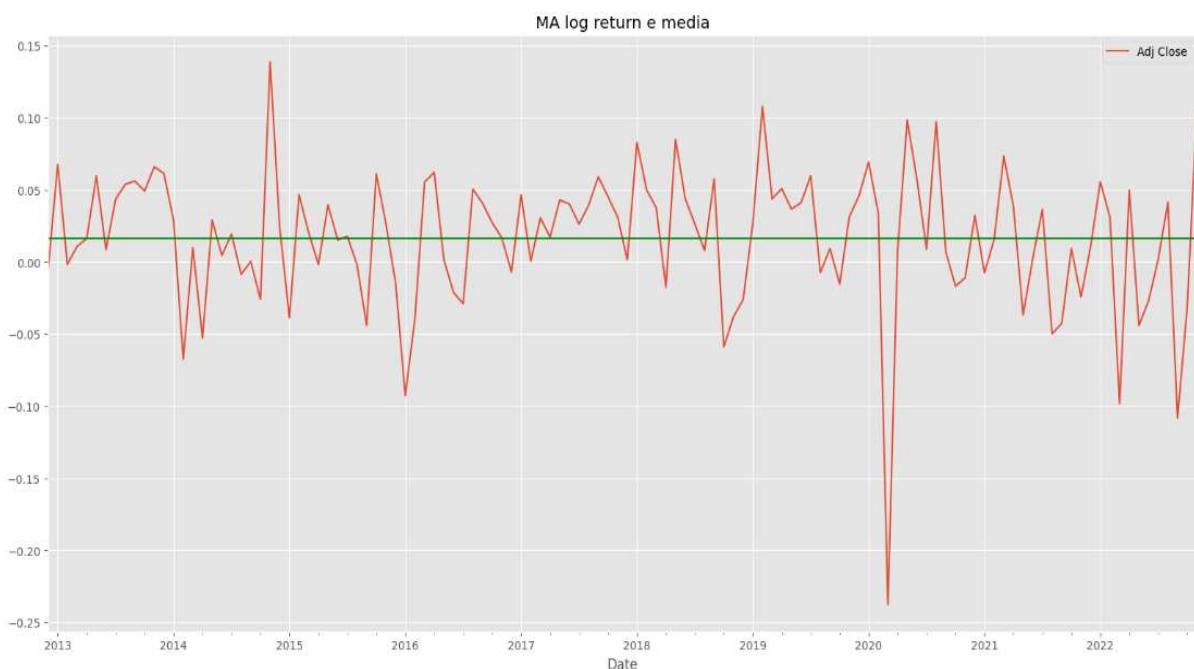
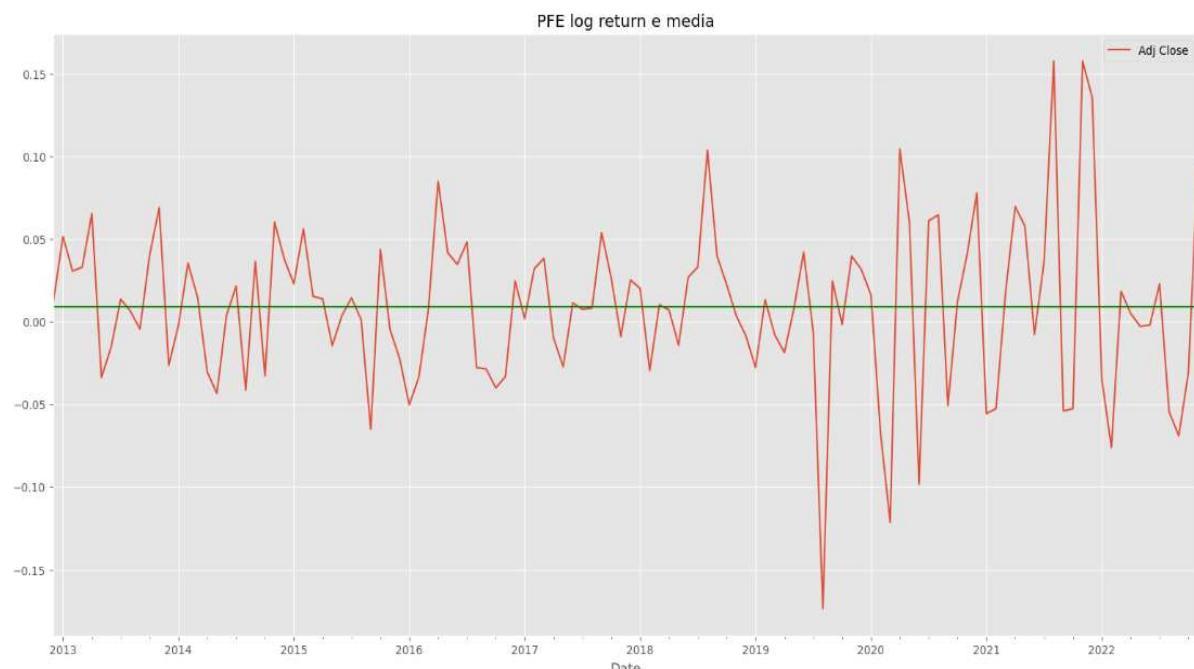
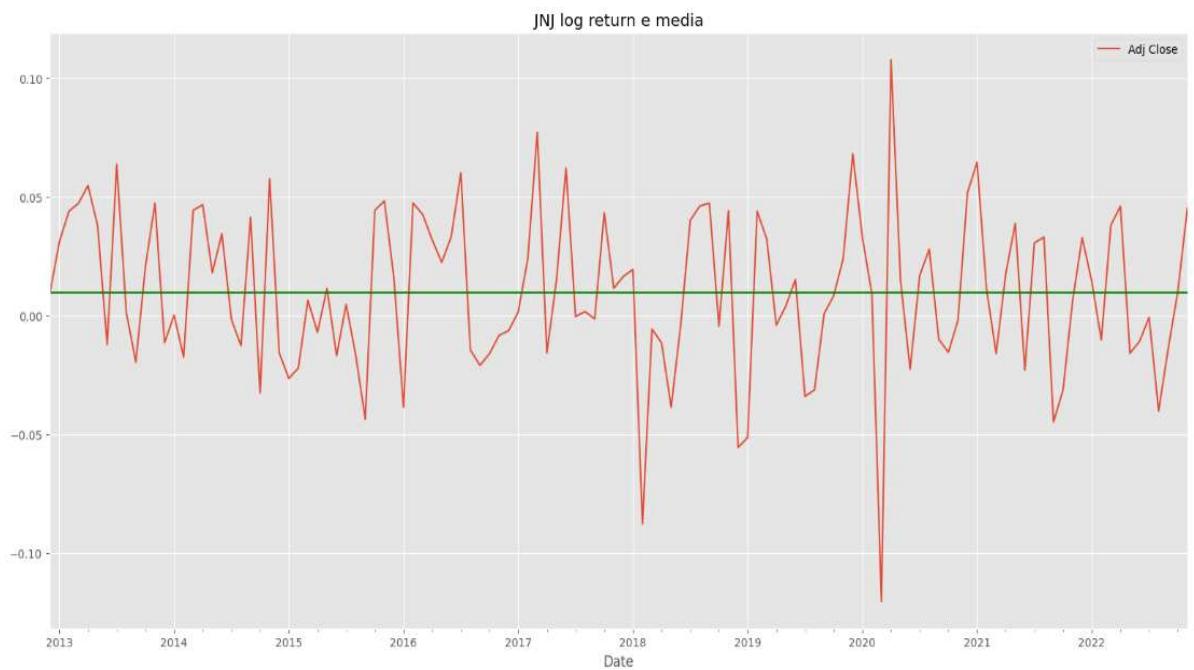


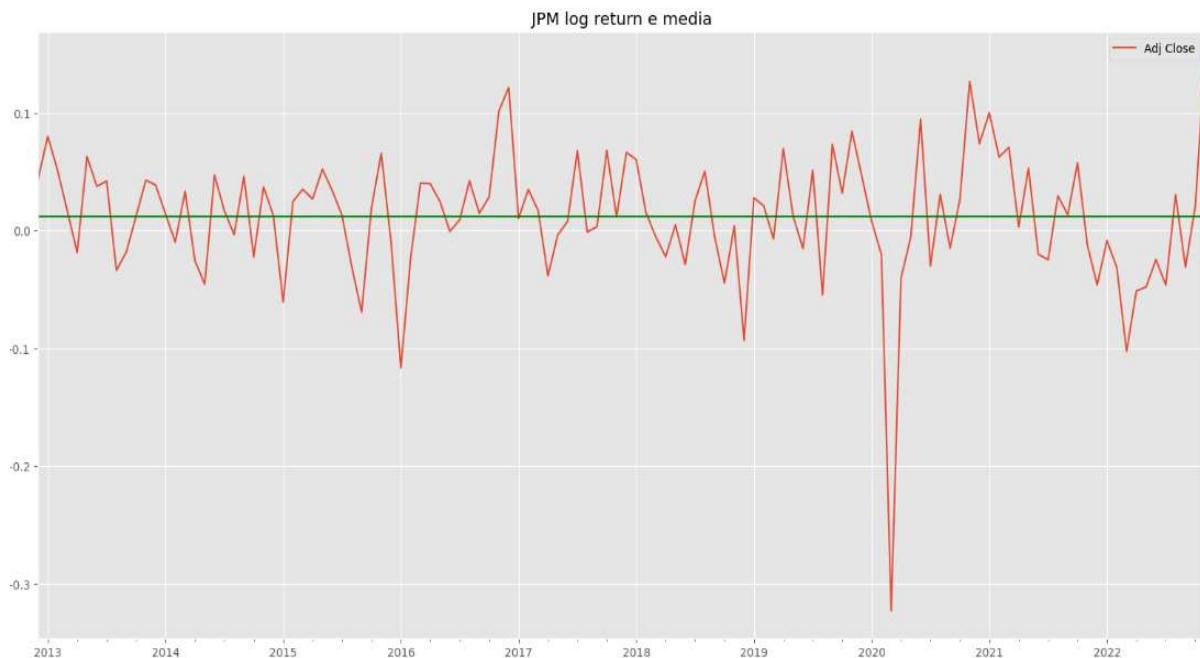
Per calcolare i **rendimenti logaritmici** utilizzo la funzione `log()` di *Numpy* e li mostro su di un grafico, al quale aggiungo la media di ogni rendimento per mostrare la variazione da essa:

```
TXN_log_return = np.log(TXN_month["Adj Close"] / TXN_month.shift(1)["Adj Close"]).dropna()
```

```
ax = TXN_log_return.plot(figsize=(19,9))
ax.axhline(TXN_log_return.mean(), c='g')
```







Tutte le serie storiche dei ritorni logaritmici presentano una elevata volatilità passando da ritorni del 10% a ritorni del -10%. Questo si verifica in modo più evidente nel periodo appena successivo all'inizio dell'anno 2020, caratterizzato dall'inizio della pandemia di COVID-19, dove osservo per ogni azione un ritorno negativo maggiore di 15% seguito da un repentino recupero positivo del 10%.

Un altro periodo da considerare è l'anno 2022, dove sono presenti valori negativi dei ritorni di tutte le azioni (esclusa JNJ), coincidente con l'inizio della guerra tra Russia e Ucraina; sempre in questo anno, verso la sua fine, invece riscontro un rialzo dei ritorni di ogni azione, presumibilmente per l'assestamento della situazione di guerra.

Nella serie dei ritorni di MSFT evidenzio due periodi di ritorni particolarmente elevati: nel primo trimestre del 2013 quando la società acquista la divisione Devices & Services di Nokia, azienda produttrice di cellulari, per circa 7.2 miliardi di dollari; e nella seconda metà dell'anno 2015, periodo nel quale è stato rilasciato al pubblico Windows 10, con oltre quattordici milioni di installazioni nelle prime 24 ore.

Per quanto riguarda il grafico di JNJ osservo un momento particolarmente negativo che caratterizza l'anno 2018, questo corrisponde a livello storico con il risarcimento danni da parte della società di 9 miliardi di dollari a persone colpite da cancro e tumori, causati da alcuni dei suoi prodotti.

A questo punto calcolo i **ritorni cumulati** e **composti** annui. Per questi è necessario raggruppare i dati per anni. Utilizzo la funzione *groupBy* con *frequenza = 'Y'* (è equivalente ad usare *resample('Y')*) e, come in precedenza, applico le funzioni *pct_change()* e *dropna()* per ottenere i ritorni semplici netti annui.

```
TXN_year = TXN.groupby(pd.Grouper(freq="Y")).mean()
TXN_year_return = TXN_year['Adj Close'].pct_change().dropna()
```

Per calcolare i ritorni **cumulati annui** utilizzo la funzione *cumprod()* di *Numpy* in questo modo:

```
TXN_cum_return = np.cumprod(1+TXN_year_return)
```

Ed ottengo che:

```
Ritorno cumulato annuo TXN: 7.145
Ritorno cumulato annuo MSFT: 12.1
Ritorno cumulato annuo JNJ: 3.164
Ritorno cumulato annuo PFE: 2.909
Ritorno cumulato annuo MA: 7.4013
Ritorno cumulato annuo JPM: 3.878
```

Questi valori indicano che se avessi investito un dollaro, all'inizio del periodo preso in considerazione, in una di queste azioni, a fine periodo, mi sarei ritrovato con una quantità di dollari corrispondente al valore mostrato sopra. Da notare che, se avessi investito in una sola di queste azioni, avrei almeno triplicato il mio investimento iniziale.

Mentre, per i **ritorni composti** utilizzo la formula:

```
TXN_compound_return = TXN_year_return.mean() - 0.5*(TXN_year_return.std())**2
```

Ed ottengo:

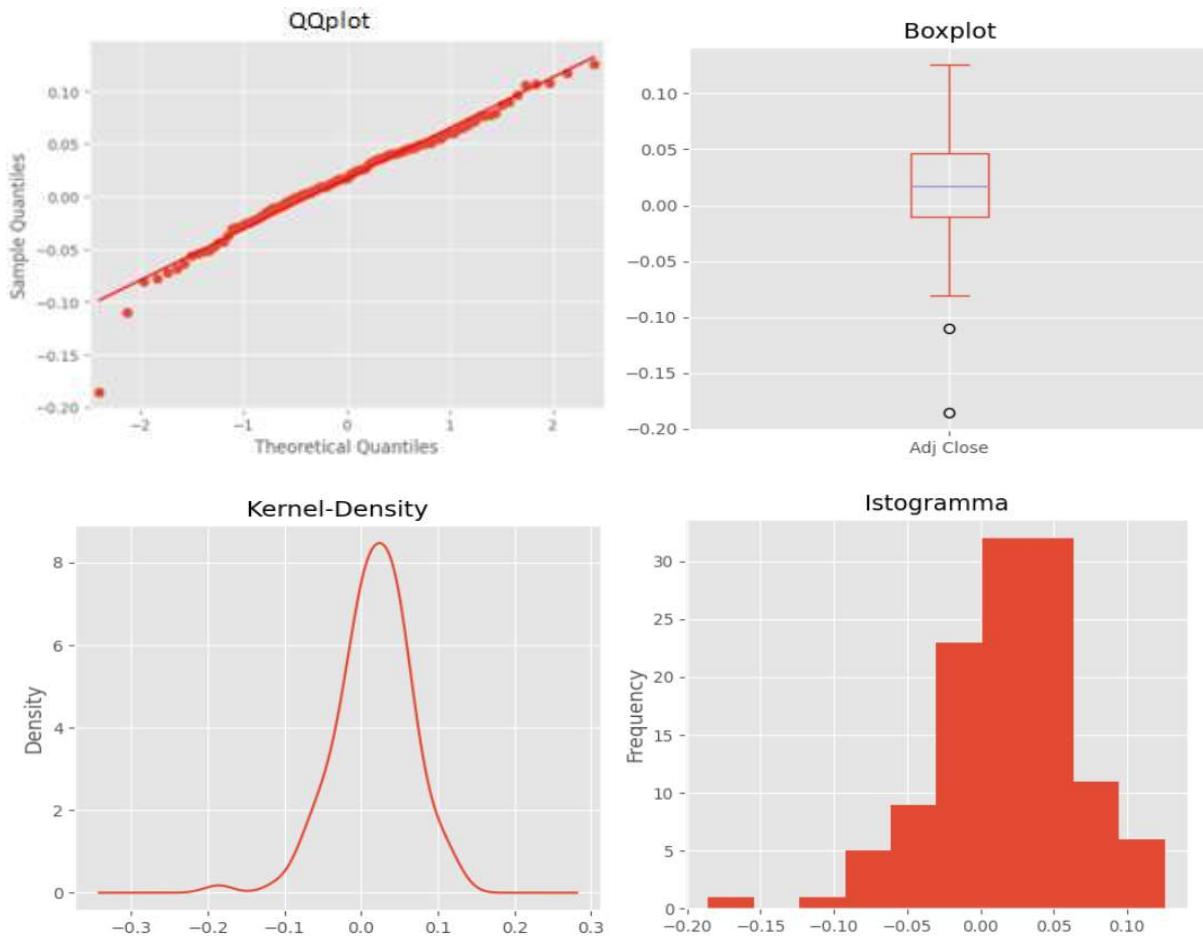
```
Ritorno composto TXN: 0.215
Ritorno composto MSFT: 0.28
Ritorno composto JNJ: 0.121
Ritorno composto PFE: 0.111
Ritorno composto MA: 0.2184
Ritorno composto JPM: 0.140
```

Ora per ogni serie di rendimenti logaritmici mostro i grafici diagnostici:

- **Istogramma:** mostra la frequenza con la quale si verificano i ritorni (rappresentazione discreta).
- **Boxplot:** mostra la mediana e i valori *outliers*.
- **QQplot:** mostra se la variabile osservata presenta una distribuzione normale, cioè se i punti di questa si addensano sulla diagonale.
- **Kernel – Density:** è una generalizzazione del metodo di stima dell'istogramma, con una rappresentazione delle frequenze continua.

```
import statsmodels.api as sm
sm.qqplot(TXN_log_return, line = 's')
plt.figure(figsize=(12, 10))
plt.subplot(221)
TXN_log_return.plot(kind='box', grid=True, title="Boxplot")
plt.subplot(222)
TXN_log_return.plot(kind='kde', grid=True, title="Kernel-Density")
plt.subplot(223)
TXN_log_return.plot(kind='hist', grid=True, title="Istogramma")
```

Texas Instruments:

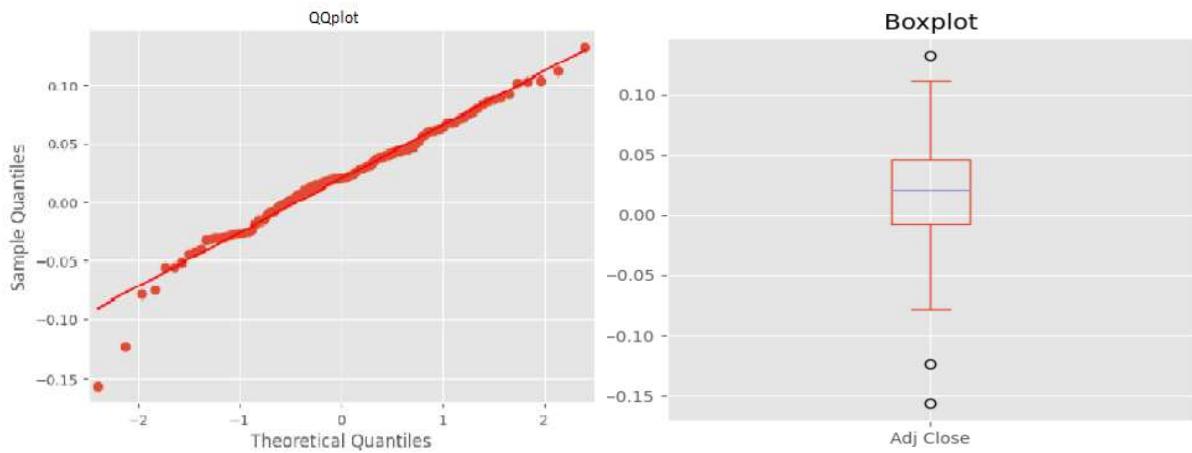


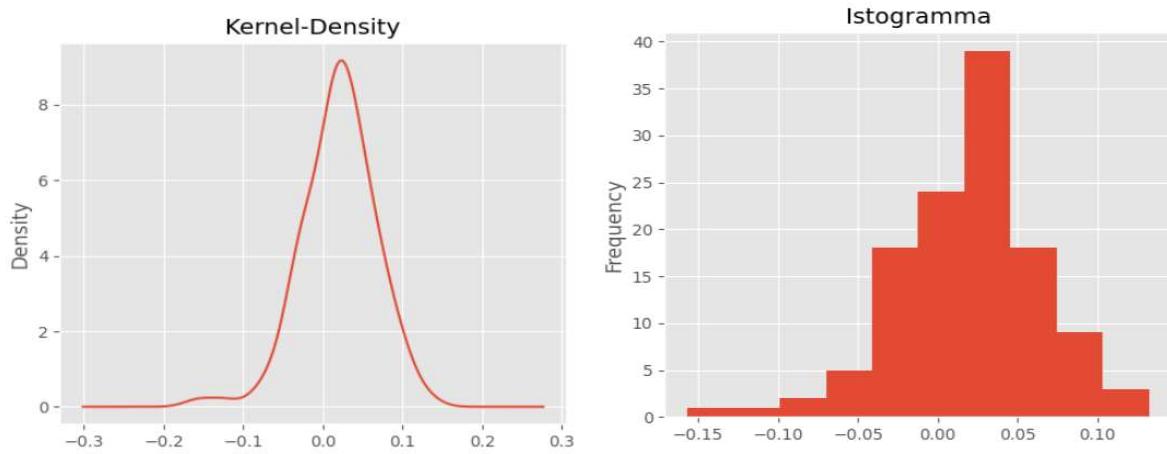
Nel QQplot noto che la distribuzione, a parte alcuni valori, segue quella di una normale.

Nel Boxplot sono presenti 2 valori *outliers* e un valore, corrispondente alla mediana, di poco inferiore a 2,5%.

Tramite i due grafici rimanenti ottengo i valori dei ritorni più frequenti nella serie: in questo caso sono compresi tra 0 e 5%; nell'istogramma posso anche notare un “buco” di valori nella serie dei ritorni al livello del -15%.

Microsoft:



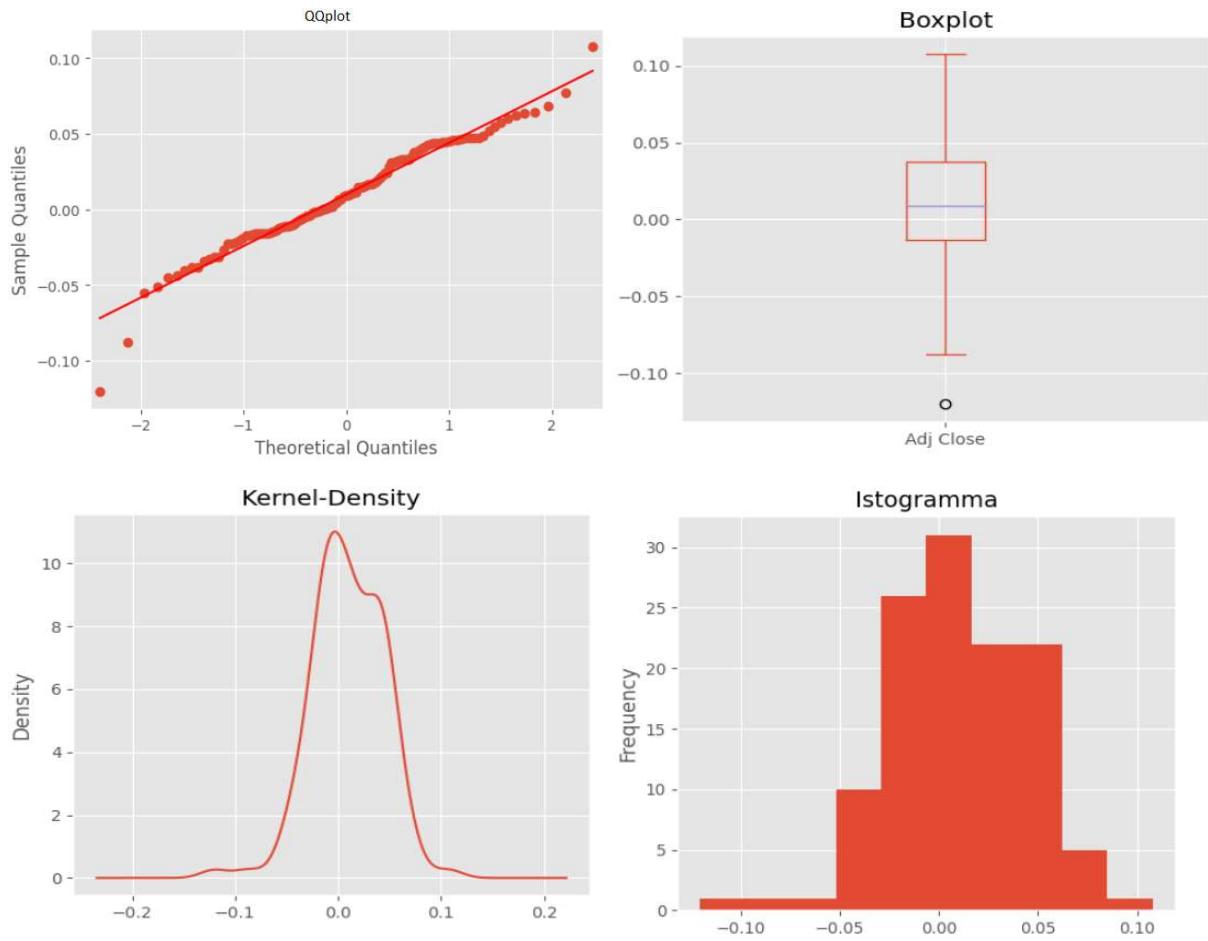


Il QQplot mostra che la distribuzione, a parte alcuni valori, segue quella di una normale.

Nel Boxplot sono presenti 3 valori *outliers* e un valore di mediana pari a 2,5%.

Gli ultimi due grafici mostrano che i valori dei ritorni più frequenti sono compresi tra 2% e 4,5%.

Johnson & Johnson:

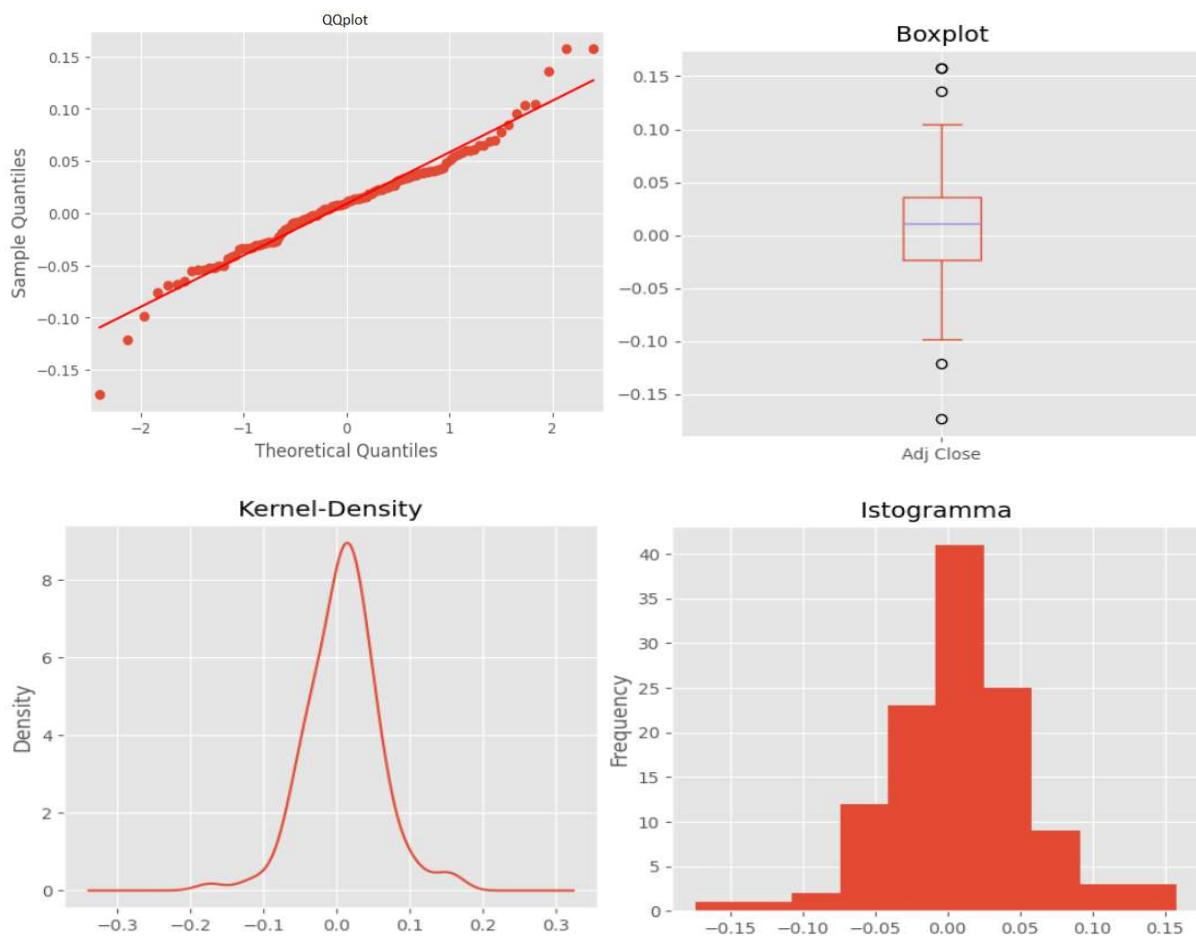


Il QQplot mostra che la distribuzione segue quella di una normale, a parte alcuni valori.

Nel Boxplot è presente un solo valore *outliers* e la mediana è all'incirca 1%.

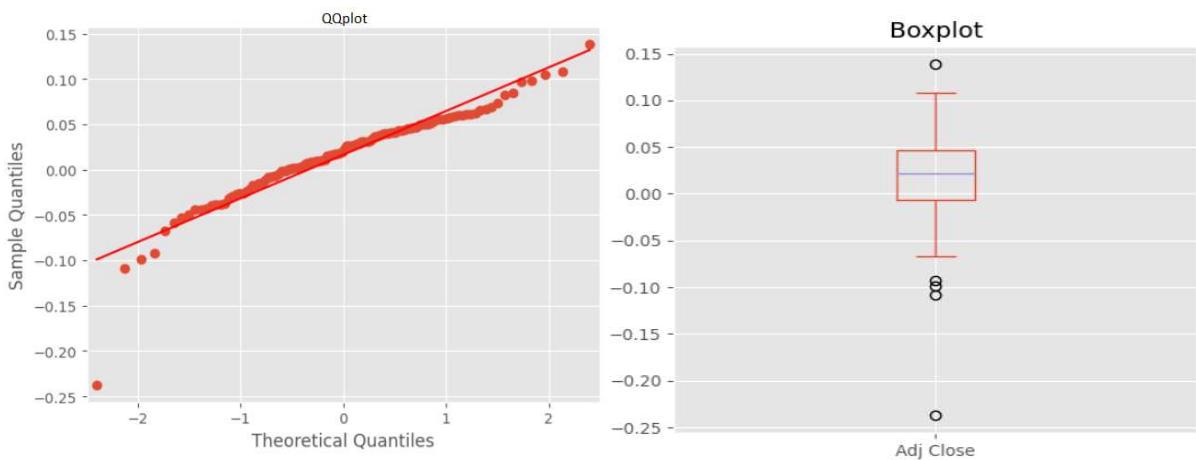
Gli altri due grafici mostrano che i valori dei ritorni più frequenti sono compresi tra lo 0% e il 2%, con una frequenza di 30.

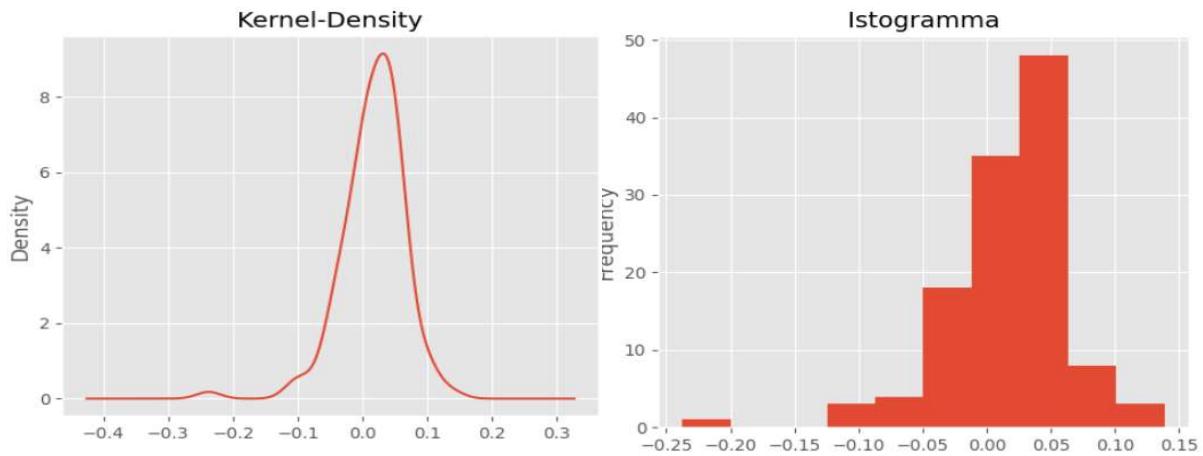
Pfizer:



Il QQplot mostra che la distribuzione segue quella di una normale, a meno di 5 valori; mentre il Boxplot presenta ben 4 valori *outliers* distribuiti equamente tra positivi e negativi, e una mediana pari a 1%. Dall'istogramma ottengo che i valori dei ritorni più frequenti nella serie, sono compresi tra 0% e 2%, con una frequenza appena superiore a 40.

Mastercard:



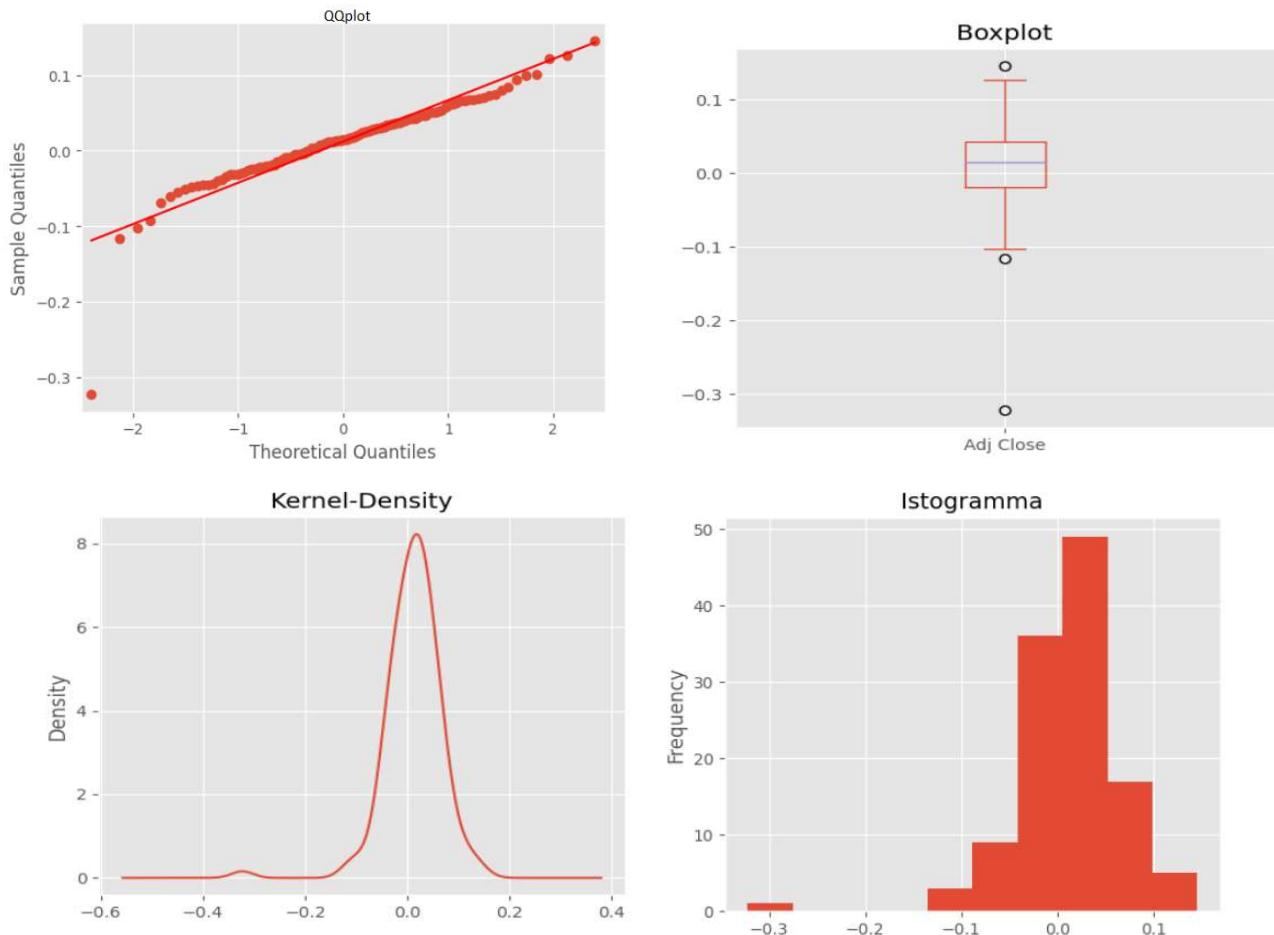


Nel QQplot osservo che la distribuzione segue quella di una normale a parte un valore che è nettamente inferiore a tutti gli altri.

Nel Boxplot è presente un elevato numero di valori *outliers* rispetto alle precedenti serie (5 outliers), e un valore di mediana pari a 2,5%.

I due grafici rimanenti mostrano che i valori dei ritorni più frequenti nella serie si aggirano intorno a 5%; nell'istogramma è presente anche un “buco” di valori nella serie dei ritorni.

J.P. Morgan:



Nel QQplot osservo che la distribuzione segue quella di una normale a parte un valore che è nettamente inferiore a tutti gli altri.

Il Boxplot mostra 3 valori *outliers* e una mediana che è all'incirca l'1%.

Tramite i due grafici rimanenti noto che i valori dei ritorni più frequenti nella serie sono compresi tra 0% e 5% con una frequenza che sfiora 50; nell'istogramma è presente, anche in questo caso, un "buco" di valori nella serie.

Per ogni serie di rendimenti logaritmici è possibile calcolare la **media**, la **varianza**, la **deviazione standard**, l'**asimmetria** e la **curtosi** (tutti i valori sono arrotondati a quattro cifre decimali) tramite i cinque metodi di *Pandas*:

1. *Mean()* per la media;
2. *Var()* per la varianza;
3. *Std()* per la deviazione standard;
4. *Skew()* per l'asimmetria;
5. *Kurtosis()* per la curtosi.

```
Media ritorni logaritmici TXN: 0.0169
Media ritorni logaritmici MSFT: 0.0198
Media ritorni logaritmici JNJ: 0.0098
Media ritorni logaritmici PFE: 0.0089
Media ritorni logaritmici MA: 0.0165
Media ritorni logaritmici JPM: 0.0121
```

Da questi valori noto che tutte le azioni hanno un rendimento medio maggiore di 0; quella che ha reso maggiormente è MSFT con un valore di quasi il 2%, mentre quella che ha reso meno è PFE con un valore appena sotto l'uno per cento (0,89%)

```
Varianza ritoni logaritmici TXN: 0.0023
Varianza ritoni logaritmici MSFT: 0.0021
Varianza ritoni logaritmici JNJ: 0.0012
Varianza ritoni logaritmici PFE: 0.0025
Varianza ritoni logaritmici MA: 0.0023
Varianza ritoni logaritmici JPM: 0.003
```

Il calcolo della varianza fornisce il valore del rischio delle varie azioni, in questo caso la più rischiosa è JPM e quella meno rischiosa è JNJ.

```
Deviazione standard ritorni logaritmici TXN: 0.0484  
Deviazione standard ritorni logaritmici MSFT: 0.046  
Deviazione standard ritorni logaritmici JNJ: 0.0342  
Deviazione standard ritorni logaritmici PFE: 0.0496  
Deviazione standard ritorni logaritmici MA: 0.0484  
Deviazione standard ritorni logaritmici JPM: 0.0549
```

La deviazione standard permette di ottenere i dati relativi alla volatilità; in questo caso l'azione più volatile è quella di J.P. Morgan mentre quella meno volatile è quella di Johnson & Johnson; in accordo con i risultati portati dal calcolo della varianza.

```
Asimmetria ritorni logaritmici TXN: -0.6825  
Asimmetria ritorni logaritmici MSFT: -0.5692  
Asimmetria ritorni logaritmici JNJ: -0.3872  
Asimmetria ritorni logaritmici PFE: -0.0414  
Asimmetria ritorni logaritmici MA: -1.3574  
Asimmetria ritorni logaritmici JPM: -1.8309
```

Tutte le azioni presentano un'asimmetria negativa, questo significa che la media dei valori calcolata precedentemente si trova a sinistra (è inferiore) al valore di moda (valore più frequente).

```
Kurtosi ritorni logaritmici TXN: 1.9372  
Kurtosi ritorni logaritmici MSFT: 1.5556  
Kurtosi ritorni logaritmici JNJ: 1.2309  
Kurtosi ritorni logaritmici PFE: 1.9714  
Kurtosi ritorni logaritmici MA: 5.8159  
Kurtosi ritorni logaritmici JPM: 10.8416
```

Tutte le distribuzioni presentano un valore di curtosi maggiore di zero, ad indicare che le curve che disegnano sono più appuntite rispetto alla distribuzione normale (leptocurtiche).

Per poter ottenere la **matrice di covarianza** tra tutte le coppie di azioni selezionate devo applicare la funzione `cov()` sul Data Frame che rappresenta i ritorni logaritmici di tutti i titoli.

```
df4_log_return.cov()
```

	TXN	MSFT	JNJ	PFE	MA	JPM
TXN	0.000294	0.000182	0.000079	0.000081	0.000172	0.000150
MSFT	0.000182	0.000288	0.000080	0.000085	0.000182	0.000136
JNJ	0.000079	0.000080	0.000124	0.000084	0.000085	0.000081
PFE	0.000081	0.000085	0.000084	0.000190	0.000094	0.000096
MA	0.000172	0.000182	0.000085	0.000094	0.000295	0.000171
JPM	0.000150	0.000136	0.000081	0.000096	0.000171	0.000285

Questa matrice ha sulla diagonale principale i valori della varianza di tutte le azioni che coincidono con i valori della varianza calcolati precedentemente.

Dalla matrice noto che tutti i valori ottenuti sono positivi, e da questo deduco che tutte le azioni si muovono nella stessa direzione. Posso ora mettere in evidenza se la covarianza di una security, rispetto ad una dello stesso settore, sia maggiore della covarianza rispetto a quella che rappresenta un altro settore:

- TXN ha il valore di covarianza maggiore nei confronti di MSFT (stesso settore)
- MSFT ha il valore di covarianza maggiore sia rispetto a TXN (stesso settore) sia rispetto a MA (altro settore)
- JNJ ha il valore di covarianza maggiore rispetto a MA (altro settore)
- PFE ha il valore di covarianza maggiore nei confronti di JPM (altro settore)
- MA ha il valore di covarianza maggiore nei confronti di MSFT (altro settore)
- JPM ha il valore di covarianza maggiore rispetto a MA (stesso settore)

Un altro valore da prendere in considerazione è la **correlazione** tra due azioni. Utilizzo la funzione `corr()` sul Data Frame con i ritorni logaritmici, e ottengo, come per la covarianza, una matrice con tutti i valori:

```
cor=df4_log_return.corr()
```

	TXN	MSFT	JNJ	PFE	MA	JPM
TXN	1.000000	0.624553	0.411631	0.343311	0.583831	0.518163
MSFT	0.624553	1.000000	0.422886	0.362172	0.624319	0.475360
JNJ	0.411631	0.422886	1.000000	0.546408	0.444948	0.429589
PFE	0.343311	0.362172	0.546408	1.000000	0.396162	0.412733
MA	0.583831	0.624319	0.444948	0.396162	1.000000	0.590256
JPM	0.518163	0.475360	0.429589	0.412733	0.590256	1.000000

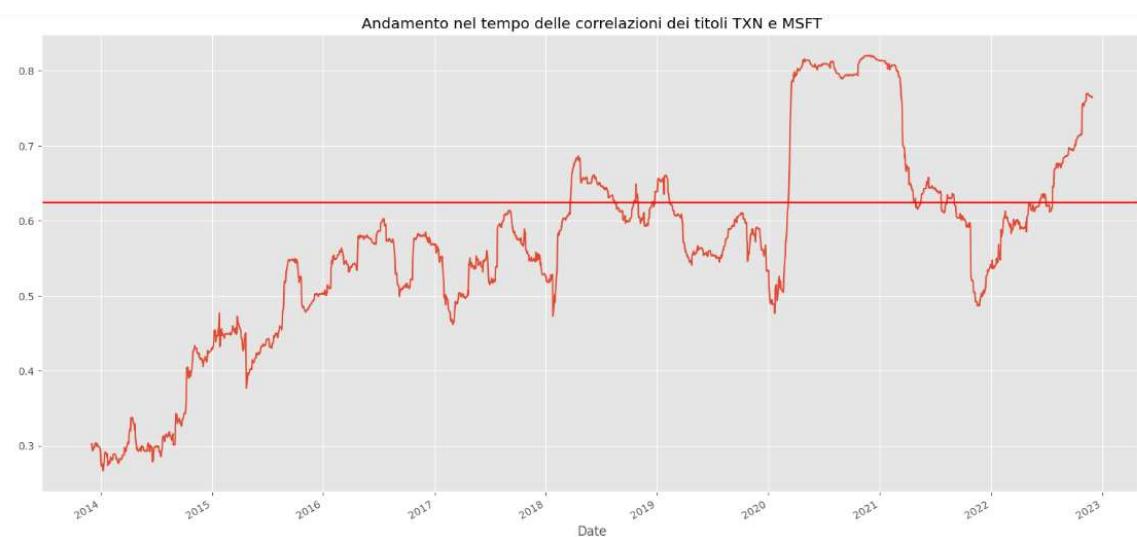
I valori sulla diagonale principale rappresentano il grado di correlazione di un'azione con sé stessa, per questo motivo sono tutti uguali a uno (valore massimo).

Le azioni più correlate sono, in ordine, TXN con MSFT, MSFT con MA e JPM con MA, da evidenziare che due su tre di queste coppie rappresentano azioni che appartengono allo stesso settore.

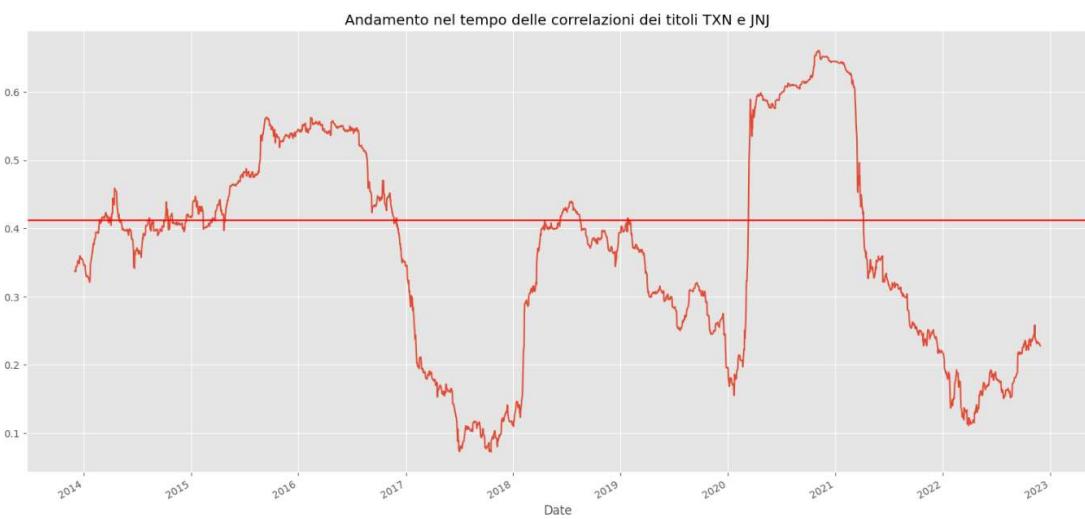
Le azioni meno correlate sono invece PFE con TXN, MSFT con PFE e MA con PFE, da sottolineare che queste tre coppie sono composte da azioni di settori differenti.

Ora mostro i grafici dell'andamento nel tempo della correlazione di ogni azione con l'altra, e la relativa correlazione media, presa dalla matrice precedente.

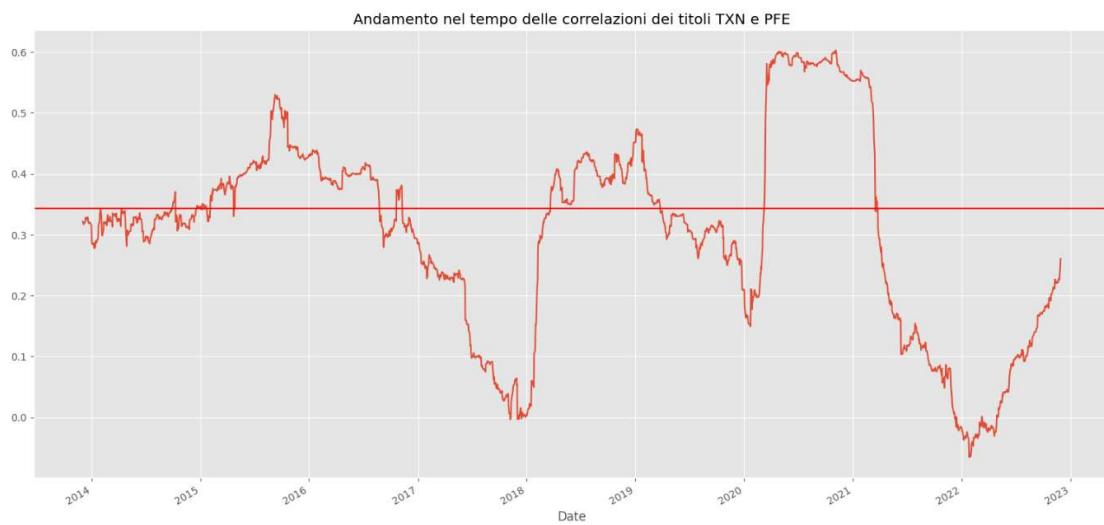
```
ax = df4_log_return[['Adj Close', 'PFE']].rolling(252).corr(df4_log_return[['Adj Close', 'TXN']]).plot(figsize=(19,9))
ax.axhline(cor.iloc[0, 3], c='r')
plt.title('Andamento nel tempo delle correlazioni dei titoli TXN e PFE')
```



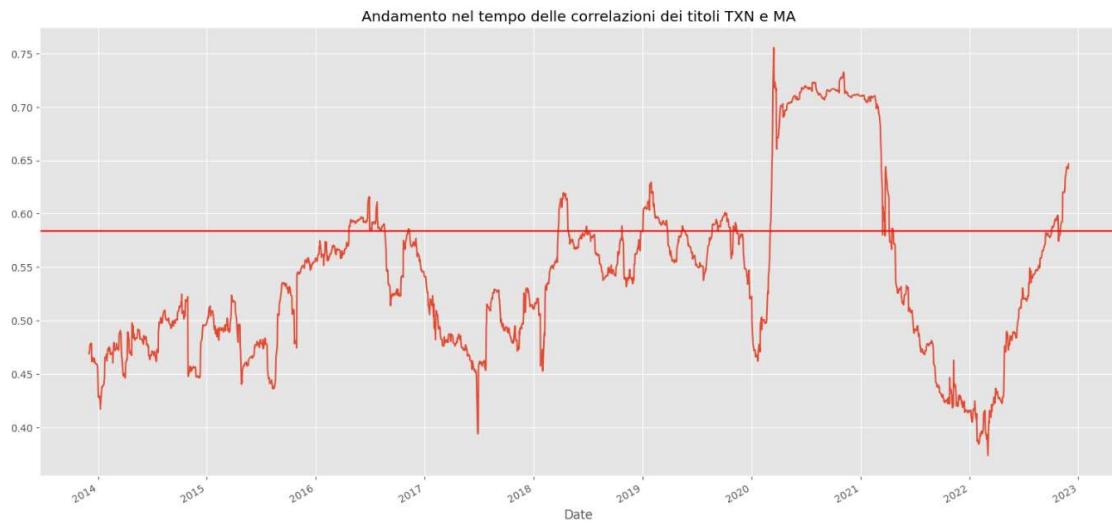
Questo grafico conferma la tendenza di MSFT e TXN ad essere correlate sempre più nel tempo.



Da questo grafico noto una forte volatilità nella correlazione dei rendimenti di TXN e JNJ che passa, ad esempio, nell'anno 2021 da un valore di 0,65 a un valore di 0,2; mai però negativa.



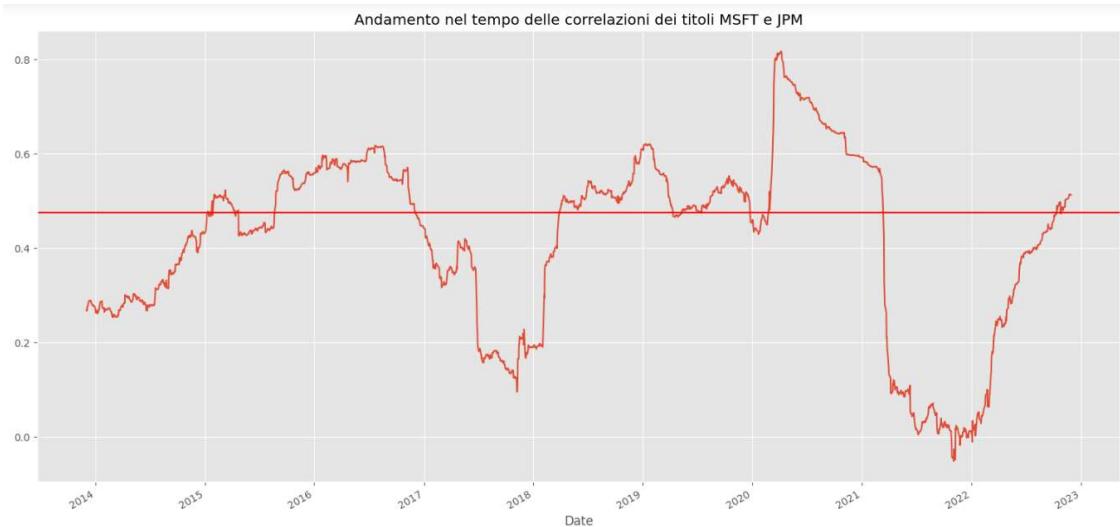
Anche questo grafico mostra una forte volatilità nella correlazione dei rendimenti tra TXN e PFE, con valori anche negativi all'inizio dell'anno 2022.



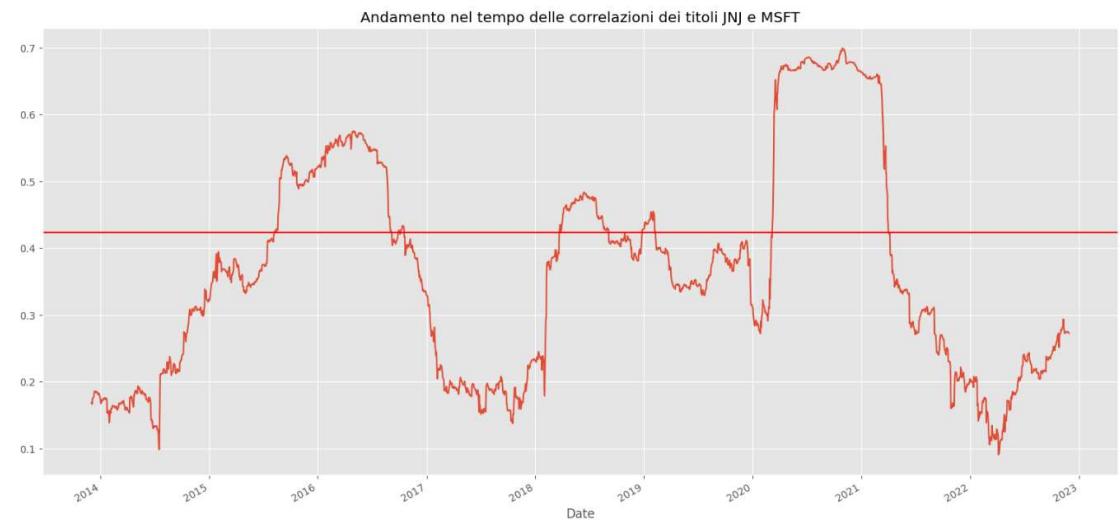
Il grafico mostra una maggiore stabilità, rispetto ai precedenti, del valore della correlazione nel tempo.



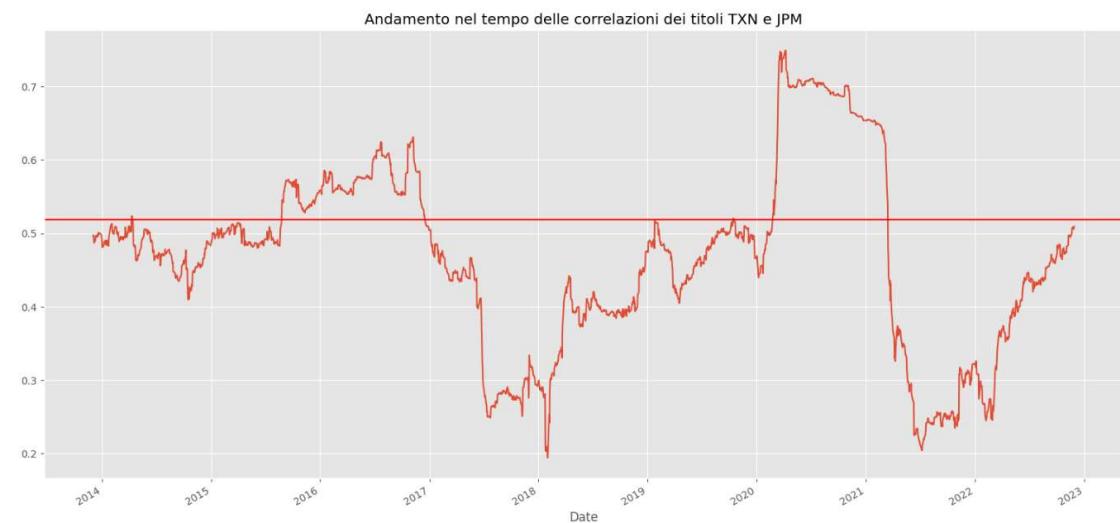
Anche in questo è presente una forte volatilità della correlazione tra PFE e MSFT, con più valori negativi



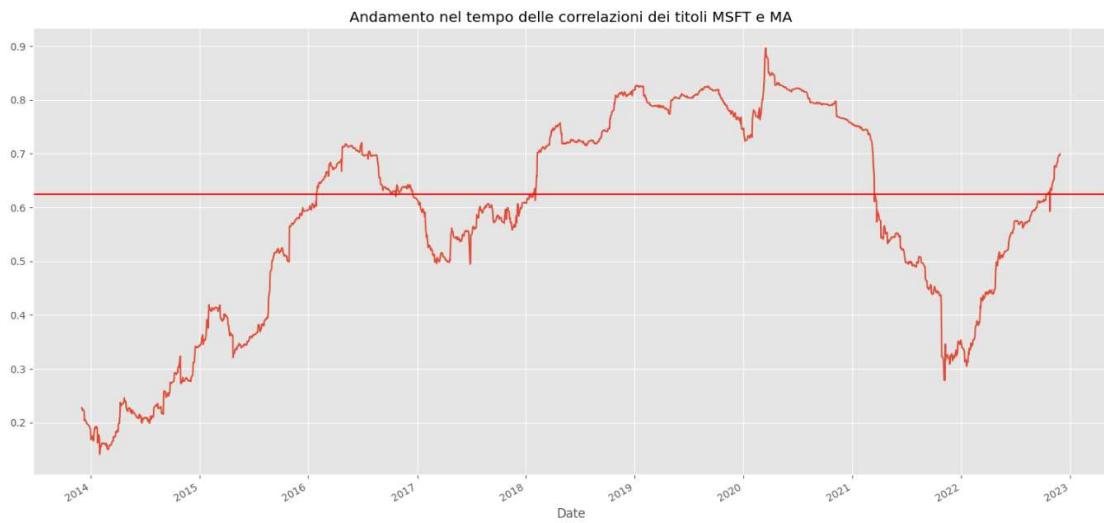
Come sopra.



Come sopra, ma con soli valori positivi.



Come sopra.



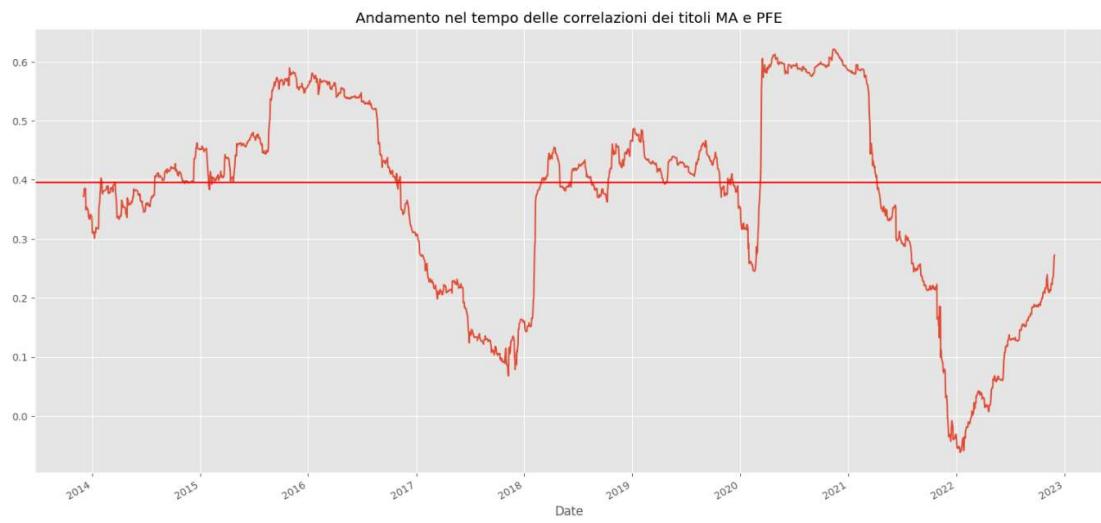
Rispetto alle altre coppie di azioni, MSFT e MA mostrano una tendenza crescente di correlazione, con un massimo nell'anno 2020 per poi crollare a fine 2021.



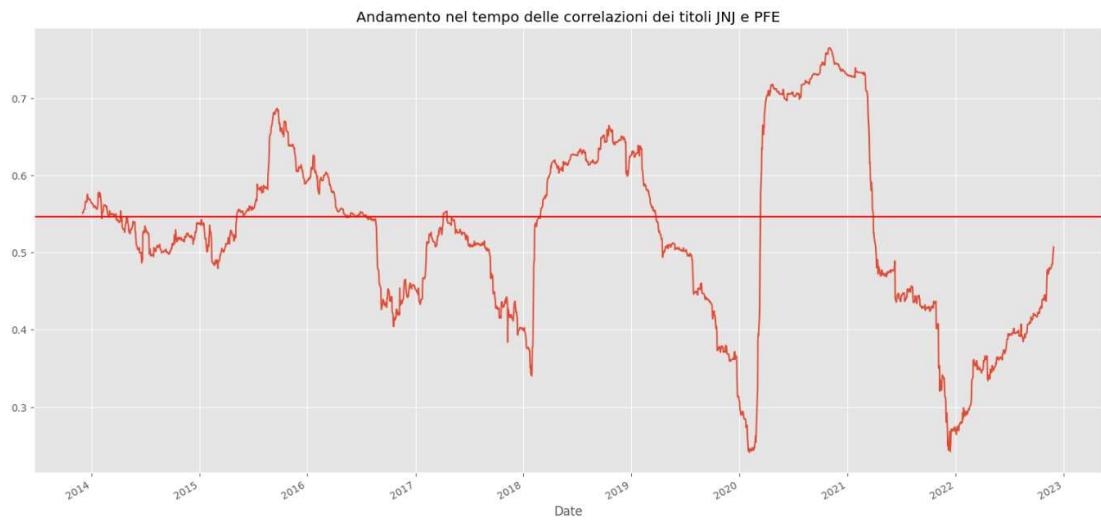
Anche in questo caso è presente una forte volatilità.



Come sopra, con valori anche negativi.



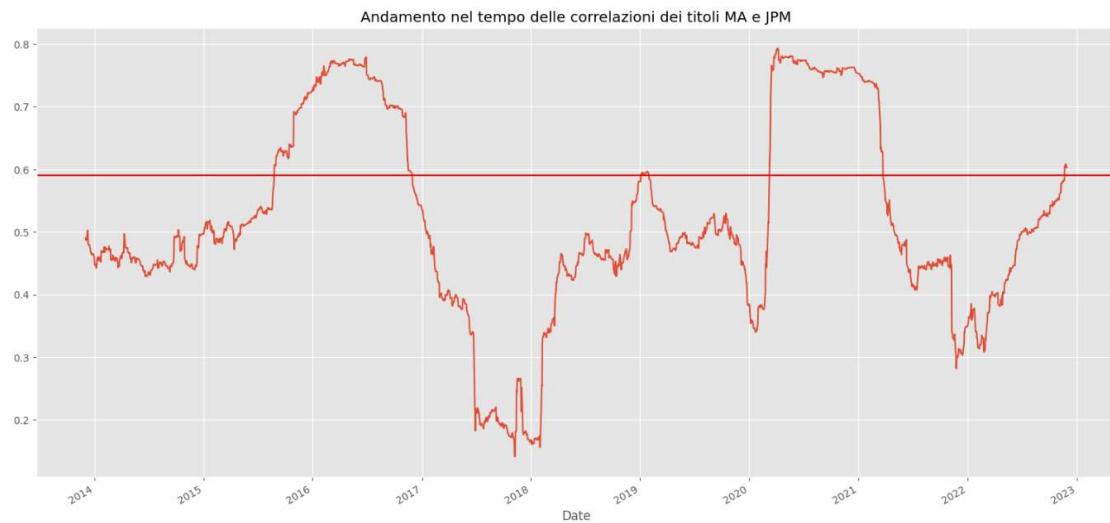
Come sopra.



Come sopra, nonostante aziende dello stesso settore.



In questo grafico è presente meno volatilità, specialmente all'inizio della serie storica.

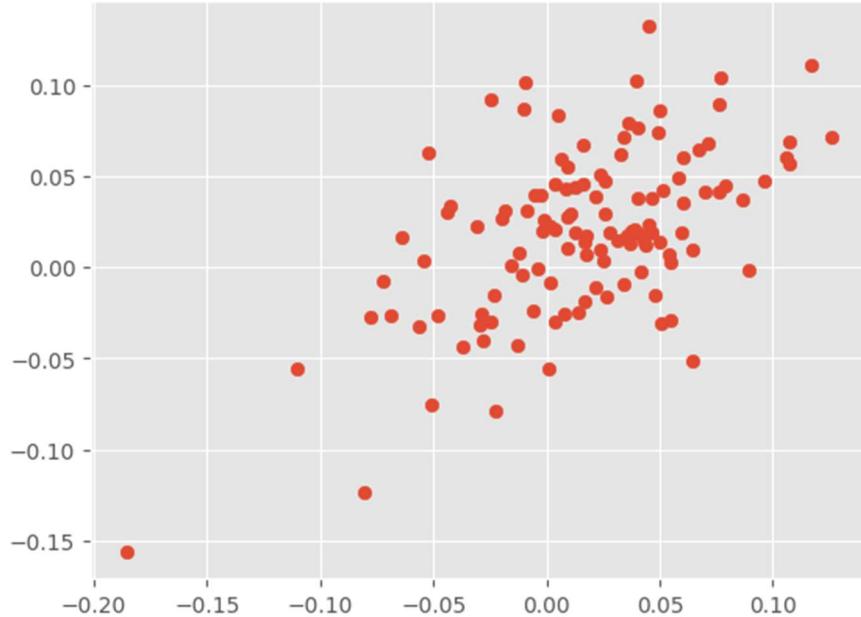


Anche nel grafico sopra riportato è presente una forte volatilità, nonostante aziende dello stesso settore.

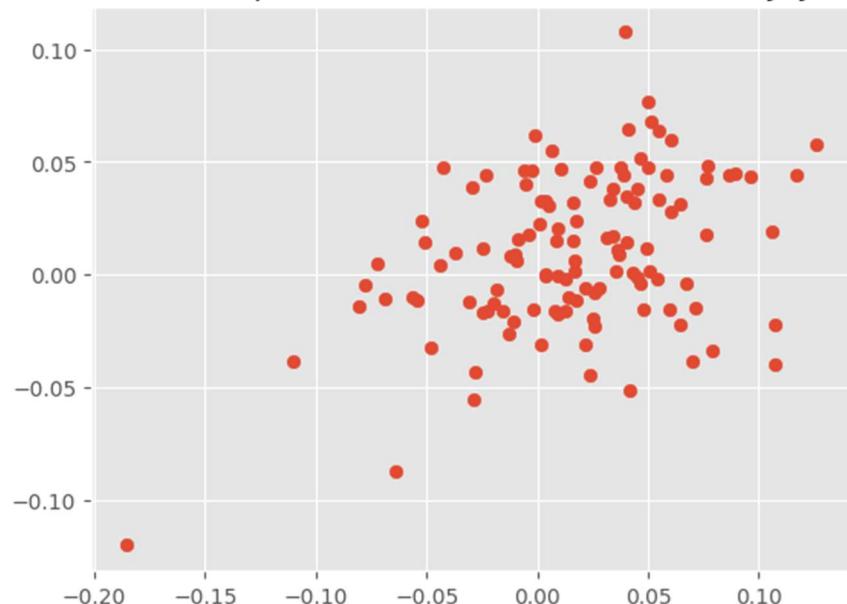
Il valore della correlazione di ogni coppia di azioni aumenta e, in molti casi, raggiunge i massimi, all'inizio dell'anno 2020, in concomitanza con l'arrivo della pandemia di Covid-19; questo ad indicare come in quel periodo tutte le azioni abbiano avuto un simile crollo dei ritorni.

Ora genero gli **Scatter Plot** (grafico a dispersione) di tutte le combinazioni di rendimenti logaritmici. Questi grafici permettono di trovare una relazione lineare tra due serie, che verrebbe descritta da una linea retta che taglia con ampiezza di 45° il grafico; viceversa verrebbe a formarsi una “nuvola” di punti.

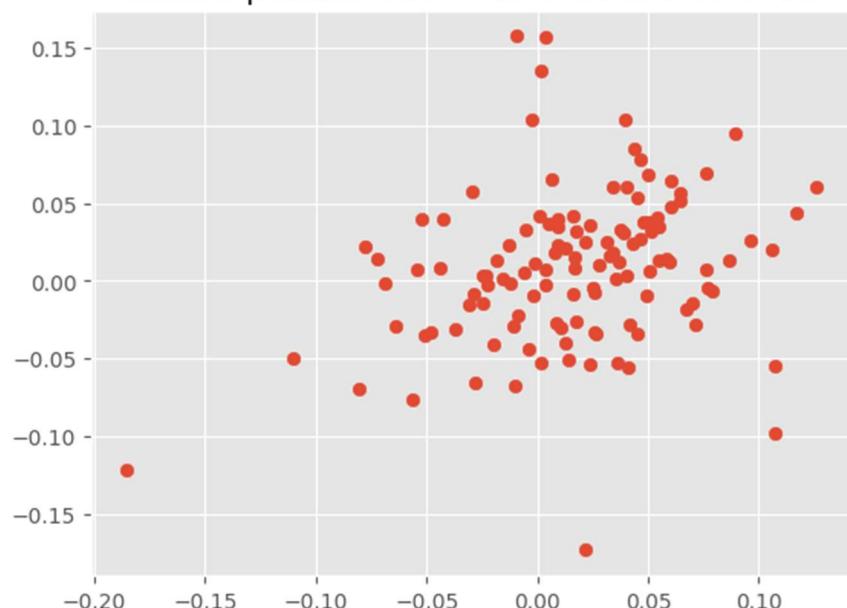
Scatter plot tra i ritorni dei titoli di TXN e MSFT



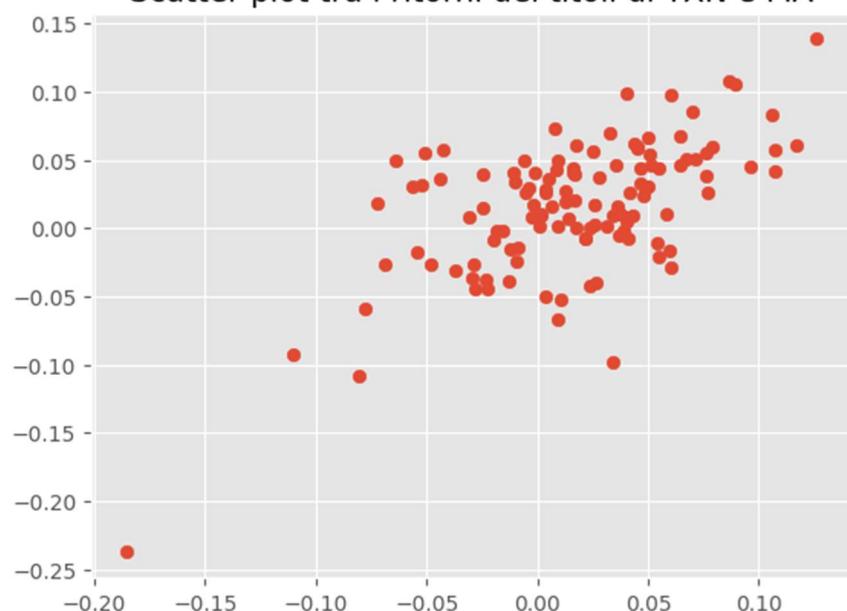
Scatter plot tra i ritorni dei titoli di TXN e JNJ



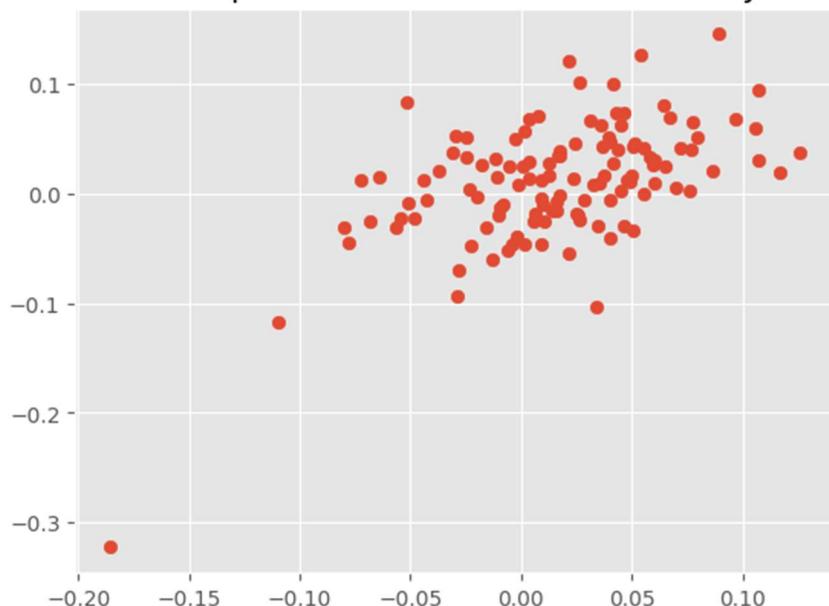
Scatter plot tra i ritorni dei titoli di TXN e PFE



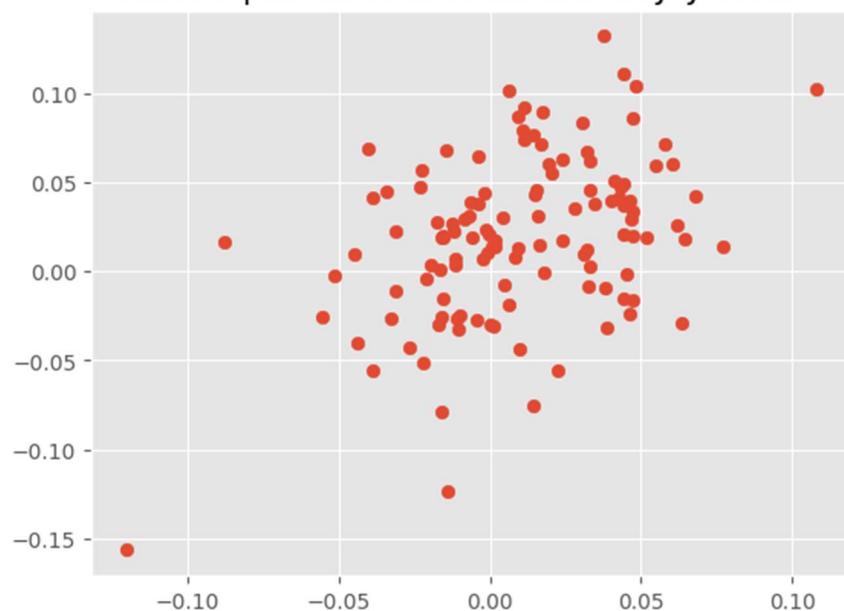
Scatter plot tra i ritorni dei titoli di TXN e MA



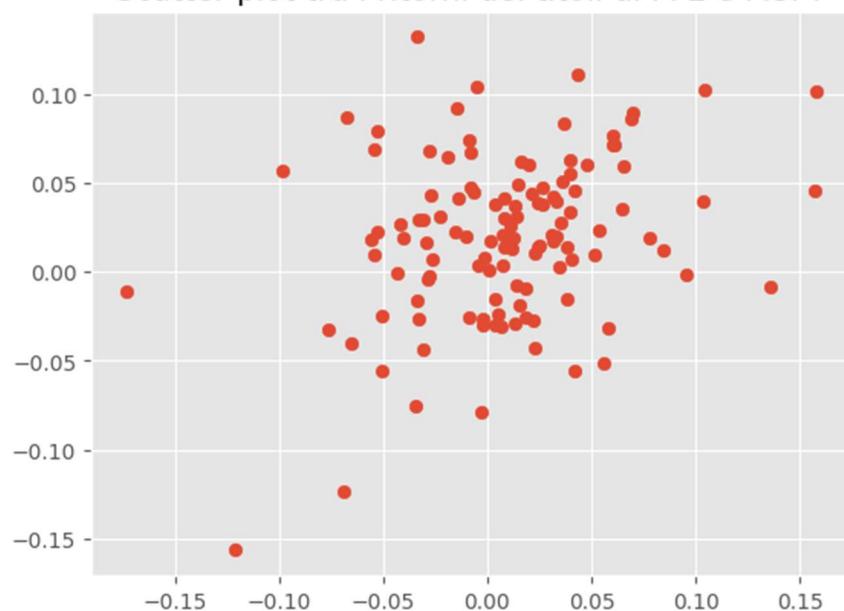
Scatter plot tra i ritorni dei titoli di TXN e JPM



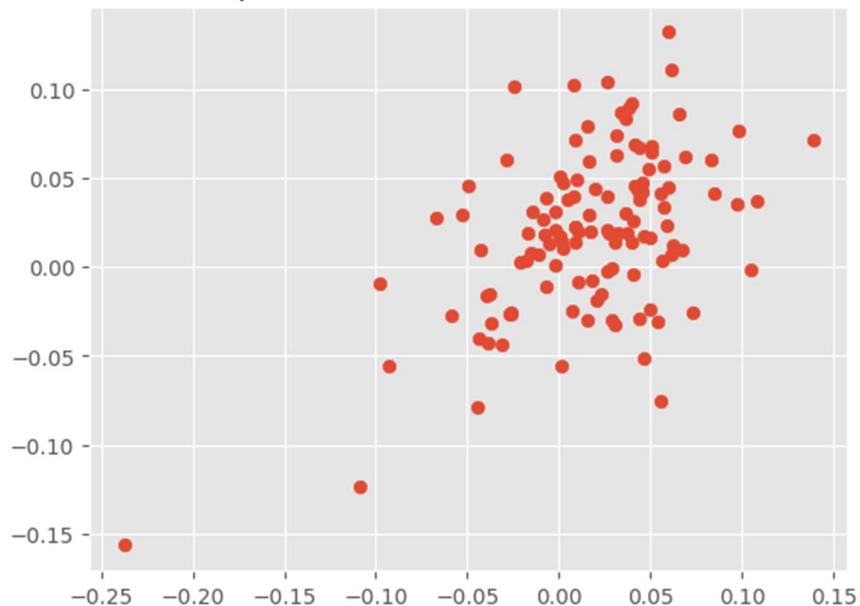
Scatter plot tra i ritorni dei titoli di JNJ e MSFT



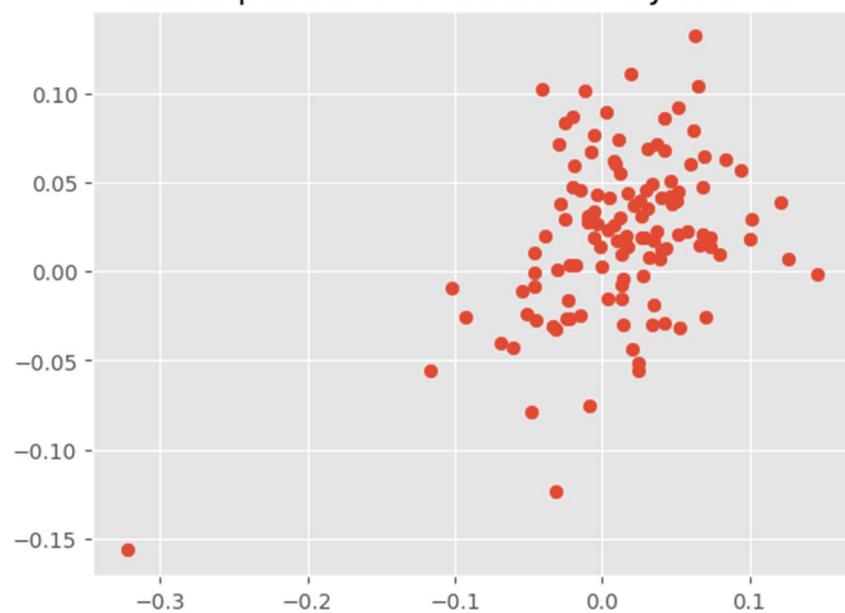
Scatter plot tra i ritorni dei titoli di PFE e MSFT



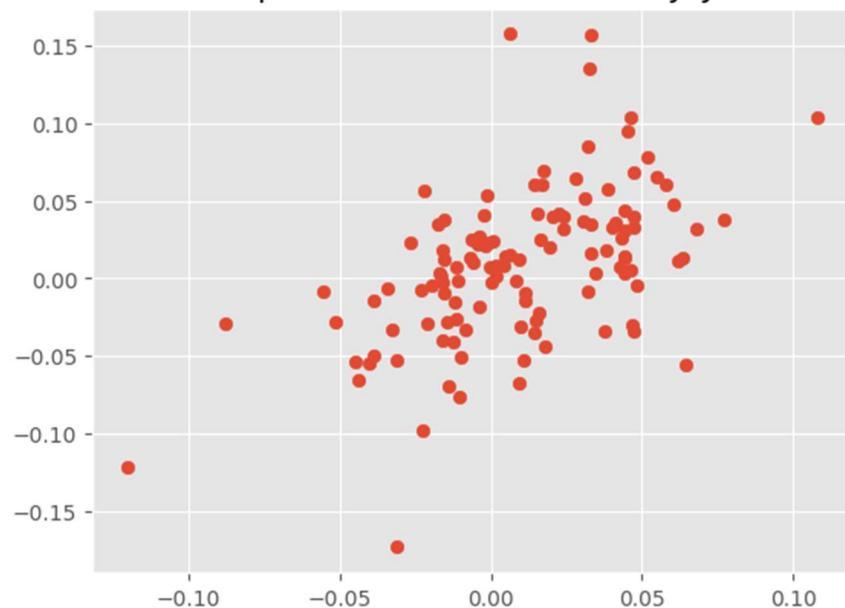
Scatter plot tra i ritorni dei titoli di MA e MSFT



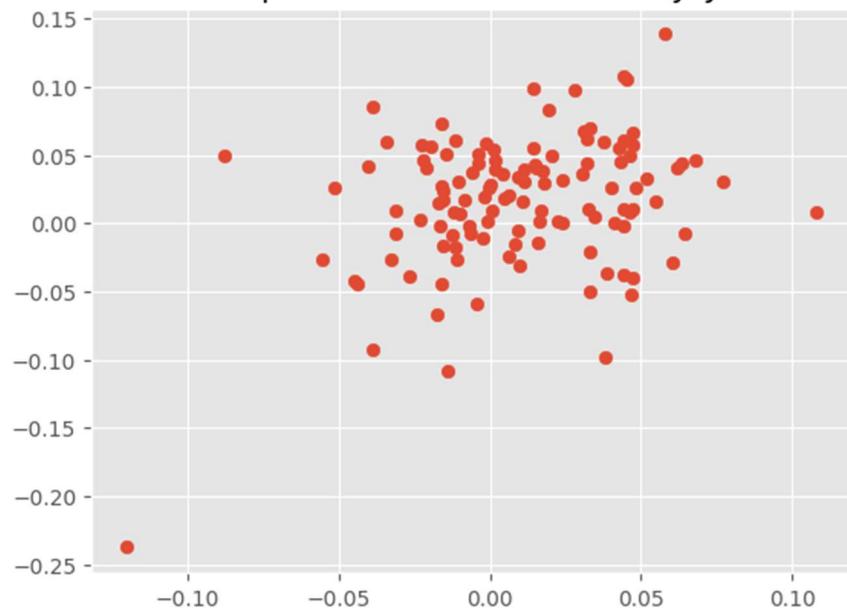
Scatter plot tra i ritorni dei titoli di JPM e MSFT



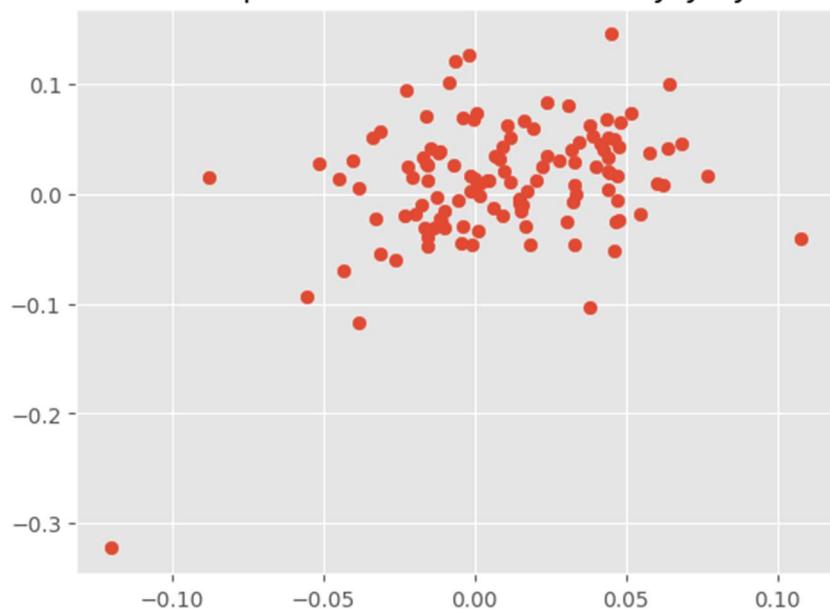
Scatter plot tra i ritorni dei titoli di JNJ e PFE



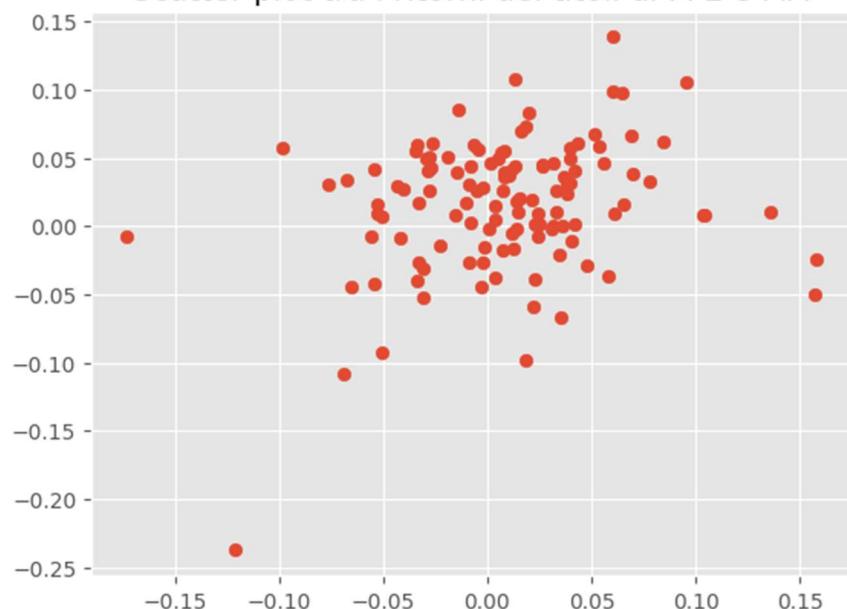
Scatter plot tra i ritorni dei titoli di JNJ e MA



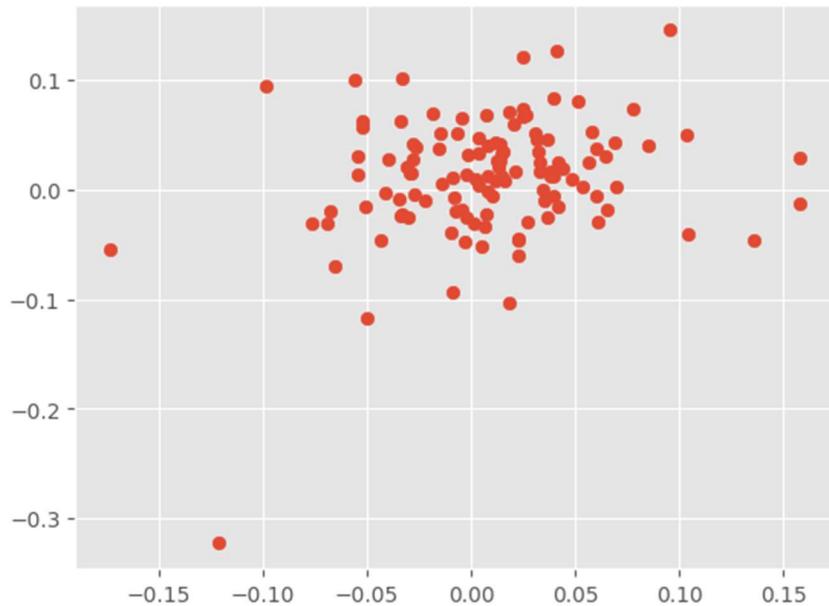
Scatter plot tra i ritorni dei titoli di JNJ e JPM



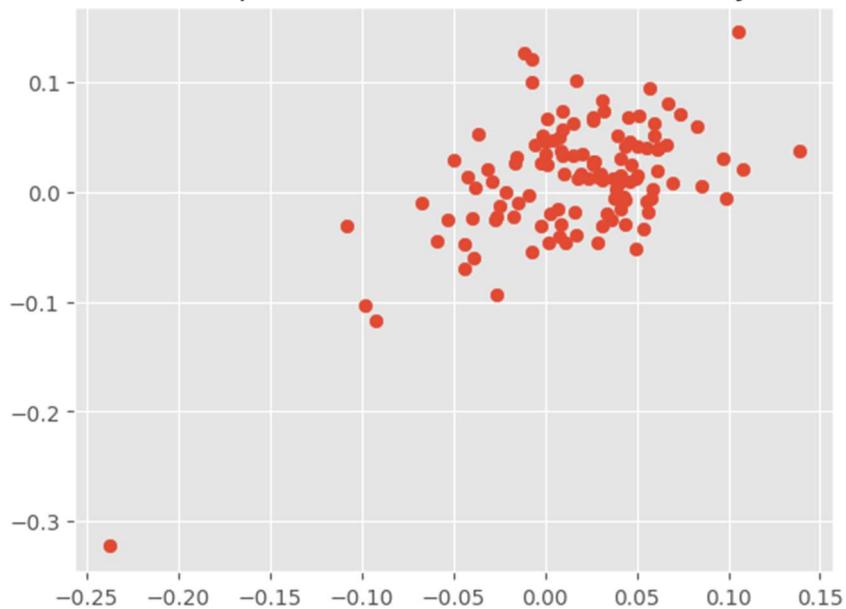
Scatter plot tra i ritorni dei titoli di PFE e MA



Scatter plot tra i ritorni dei titoli di PFE e JPM



Scatter plot tra i ritorni dei titoli di MA e JPM



Questi grafici mostrano come non ci sia una vera e propria relazione lineare tra due rendimenti, compresi quelli che mettono a confronto azioni dello stesso settore.

Per tutte queste serie di rendimenti, si distingue un punto che è nettamente separato dal resto, che si trova nella parte inferiore sinistra del grafico ed è il valore avuto nel marzo 2020, nel periodo iniziale della pandemia.

In tutti questi grafici posso osservare come la maggior parte dei punti siano compresi nel quadrato formato dalle linee di ascissa 0,1 e -0,1 e ordinata 0,1 e -0,1. Perciò posso concludere che la dispersione dei punti delle coppie di azioni sia, a meno di qualche elemento, simile.

Capitolo 3: Analisi di previsione

In questo capitolo utilizzo l'algoritmo statistico **ARIMA** per costruire dei modelli di previsione dei prezzi delle serie storiche.

Per applicare un modello ARIMA dovrò scegliere tre **iperparametri** p, d, q :

- I valori correnti di una serie di dati vengono correlati con i valori precedenti nella stessa serie per produrre il componente AR, noto come p ;
- i valori correnti di un termine di errore casuale vengono correlati ai valori precedenti per produrre il componente MA, q ;
- il componente I (simbolizzato da d) viene aggiunto per correggere la differenza di stazionarietà tramite la differenziazione.

Per prima cosa divido la serie storica in tre:

- una serie di **training**, utilizzata per costruire il modello e formata dagli 80 valori dei ritorni mensili iniziali;
- una serie di **validazione**, utilizzata per ricercare la combinazione di *iperparametri* migliori (quella che mostra il *mean squared error* minore) e formata dai 10 valori dei ritorni successivi alla prima serie;
- una serie di **test**, utilizzata per fare la previsione e formata dai successivi 30 valori dei ritorni (di cui gli ultimi 10 verranno utilizzati per confrontare i valori reali con quelli previsti).



Nel codice sotto riportato, creo un metodo, chiamato *evaluate_models()*, che per ogni differente combinazione dei tre *iperparametri*, presi con un valore nell'insieme { 0, 1, 2 }, crea un modello ARIMA chiamando un altro metodo, *evaluate_arima_model()*. Al suo interno, tramite la funzione ARIMA creo un modello sui valori di training e ne valuto la bontà calcolando l'errore quadratico medio delle previsioni con i valori effettivi presenti nei dati di validation. Confronto tutti questi valori calcolati per ottenere la combinazione di parametri che minimizzi l'errore.

```
def evaluate_arima_model(arima_order):
    modelARIMA = ARIMA(training_data, order=arima_order)
    model_fit = modelARIMA.fit()
    predicted = model_fit.predict(start = 81 ,end = 90)
    error = mean_squared_error(validation_data, predicted)
    return error

#valuto tutte le possibili combinazioni con un valore dei parametri compreso tra 0 e 2 e le confronto
def evaluate_models():
    p_values = [0, 1, 2]
    d_values = [0, 1, 2]
    q_values = [0, 1, 2]
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    mse = evaluate_arima_model(order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                        print('ARIMA{} MSE={:.7f} {}'.format(order,mse))
                except:
                    continue
    print('Best ARIMA{} MSE={:.7f} {}'.format(best_cfg, best_score))
```

Per ogni azione, chiamo la funzione `evaluate_models()` cambiando i valori di training e validation. Questa mi stampa tutte le combinazioni dei parametri provate, con a fianco l'errore quadratico medio calcolato:

```
ARIMA(0, 0, 0) MSE=2955.9479491
ARIMA(0, 0, 1) MSE=2948.4842502
ARIMA(0, 0, 2) MSE=2775.5206053
ARIMA(0, 1, 0) MSE=152.0885750
ARIMA(0, 1, 1) MSE=147.2764082
ARIMA(0, 1, 2) MSE=177.4612079
ARIMA(0, 2, 0) MSE=89.5156147
ARIMA(0, 2, 1) MSE=81.9323520
ARIMA(0, 2, 2) MSE=81.4785029
ARIMA(1, 0, 0) MSE=170.2665722
ARIMA(1, 0, 1) MSE=169.1251828
ARIMA(1, 0, 2) MSE=229.1791576
ARIMA(1, 1, 0) MSE=145.7806113
ARIMA(1, 1, 1) MSE=143.3364161
ARIMA(1, 1, 2) MSE=186.7372877
ARIMA(1, 2, 0) MSE=255.9162657
ARIMA(1, 2, 1) MSE=81.4951340
ARIMA(1, 2, 2) MSE=82.8969915
ARIMA(2, 0, 0) MSE=173.2406672
ARIMA(2, 0, 1) MSE=173.4193729
ARIMA(2, 0, 2) MSE=234.3065492
ARIMA(2, 1, 0) MSE=166.8180913
ARIMA(2, 1, 1) MSE=174.1912950
ARIMA(2, 1, 2) MSE=161.8696430
ARIMA(2, 2, 0) MSE=207.3488681
ARIMA(2, 2, 1) MSE=91.0867552
ARIMA(2, 2, 2) MSE=93.5116912
Best ARIMA(0, 2, 2) MSE=81.4785029
```

Nell'ultima riga è presente la combinazione migliore: nel caso di TXN è **(0, 2, 2)**.

Ora creo nuovamente il modello, con il valore di ordine trovato:

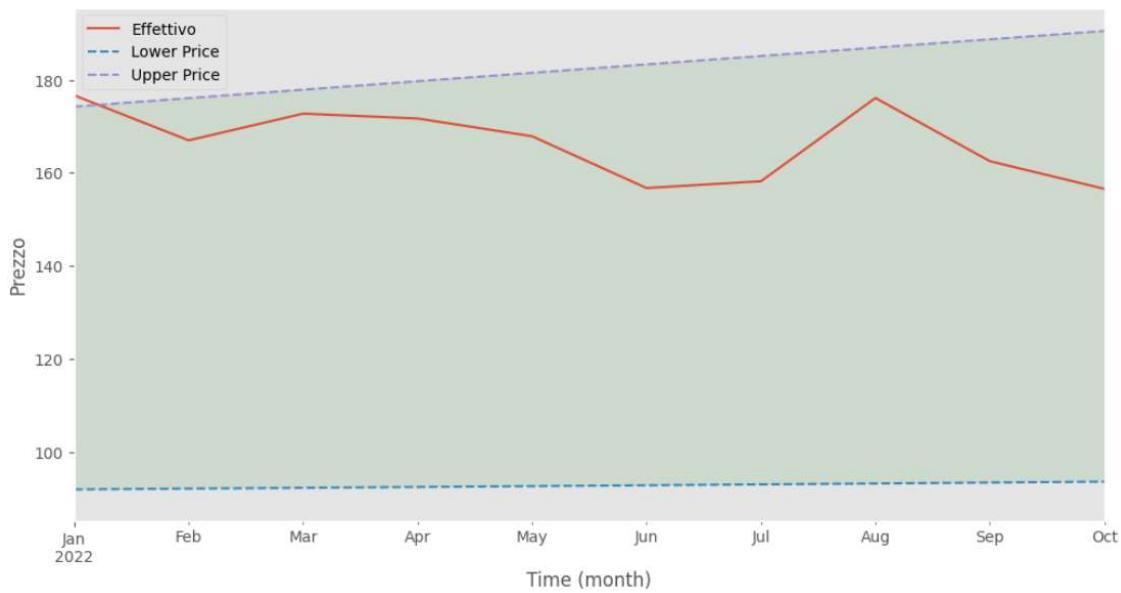
```
model = ARIMA(training_data, order = (0,2,2))
model_fit = model.fit()
model_fit.summary()
```

Dep. Variable:	y	No. Observations:	80			
Model:	ARIMA(0, 2, 2)	Log Likelihood:	-197.498			
Date:	Wed, 18 Jan 2023	AIC:	400.997			
Time:	18:32:06	BIC:	408.067			
Sample:	0	HQIC:	403.827			
	- 80					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ma.L1	-0.9413	2.856	-0.330	0.742	-6.538	4.656
ma.L2	-0.0584	0.221	-0.264	0.792	-0.492	0.375
sigma2	8.7741	25.331	0.346	0.729	-40.873	58.421
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	4.63			
Prob(Q):	0.98	Prob(JB):	0.10			
Heteroskedasticity (H):	11.10	Skew:	-0.06			
Prob(H) (two-sided):	0.00	Kurtosis:	4.19			

Applico il metodo `get_prediction()` al modello ARIMA creato, per prevedere i prezzi dal 91° mese fino al 120°, ovvero il range di prezzi di cui fanno parte i dati di testing. Questo metodo restituirà un intervallo di valori entro i quali il modello prevede che il prezzo dell'azione si muoverà nel futuro.

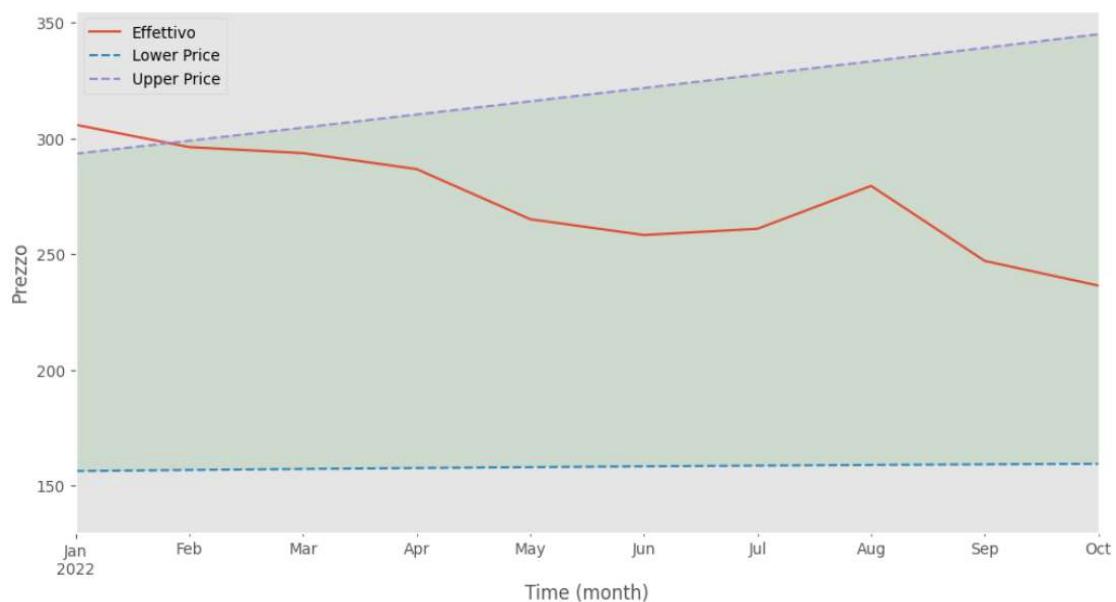
```
prediction = model_fit.get_prediction(start=91, end=120)
```

Per concludere, confronto gli ultimi 10 valori ottenuti nella previsione, con gli ultimi 10 mesi di valori effettivi presenti nel *testing_data* in un grafico:



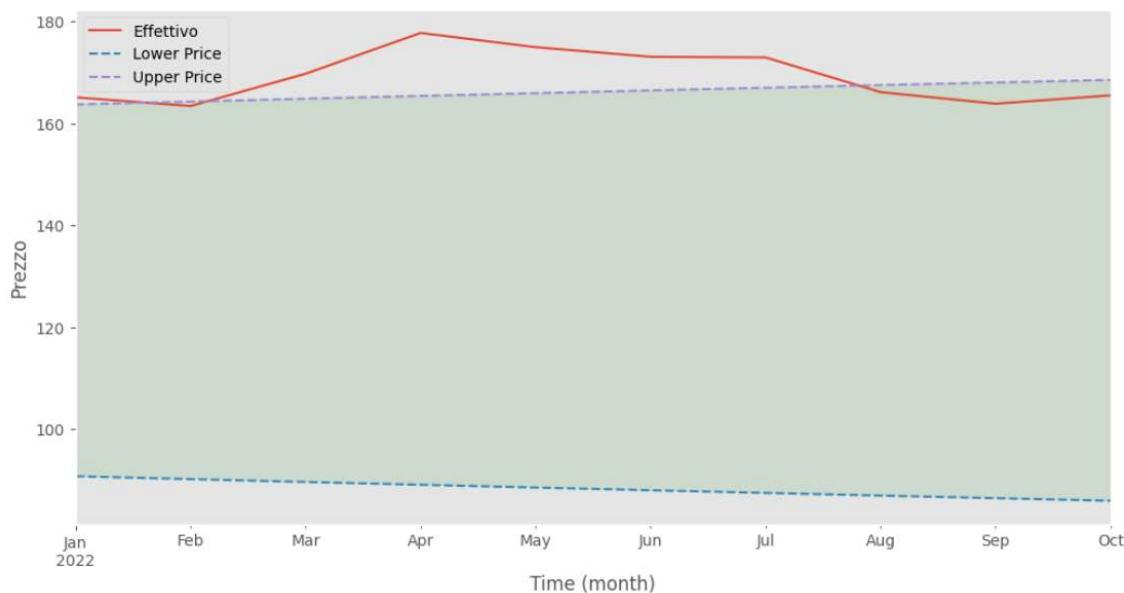
Il grafico mostra che i valori effettivi rientrano nell'area di prezzo prevista nel futuro dal modello, ed in particolare, che si muovono nella parte superiore del range.

Chiamando il metodo *evaluate_models()* con i valori dei prezzi di Microsoft, ottengo la combinazione di parametri **(0, 2, 1)**. Mostro ora il grafico del confronto:



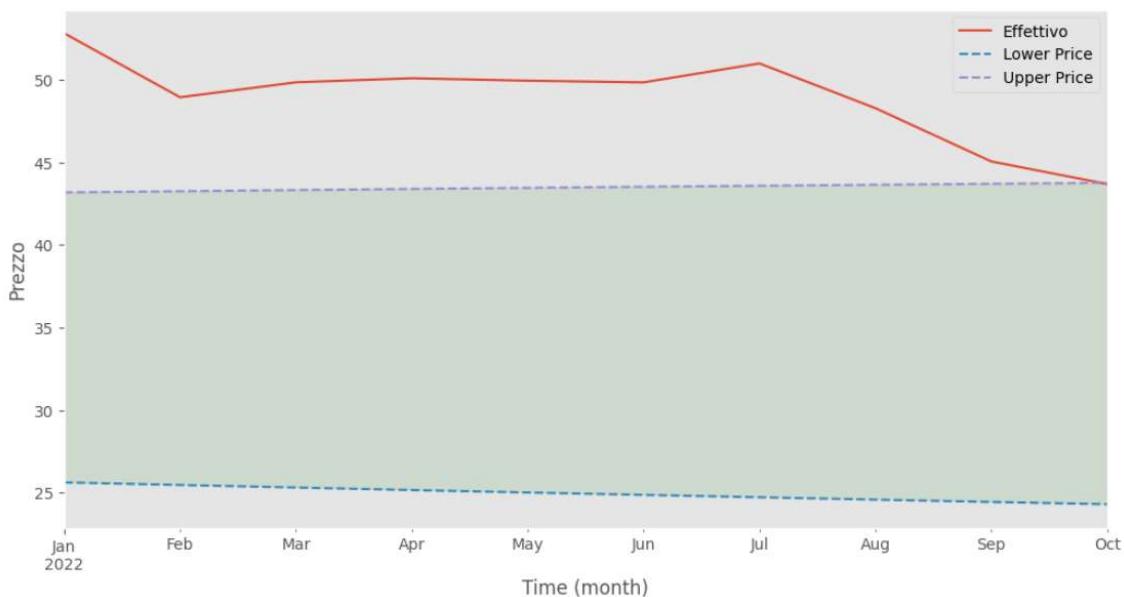
Il grafico mostra che, anche in questo caso, i valori effettivi rientrino nell'area di prezzo prevista dal modello.

Faccio lo stesso con i valori dei prezzi di **Johnson & Johnson** e ottengo la combinazione di parametri **(0, 1, 0)**. Mostro il grafico del confronto:



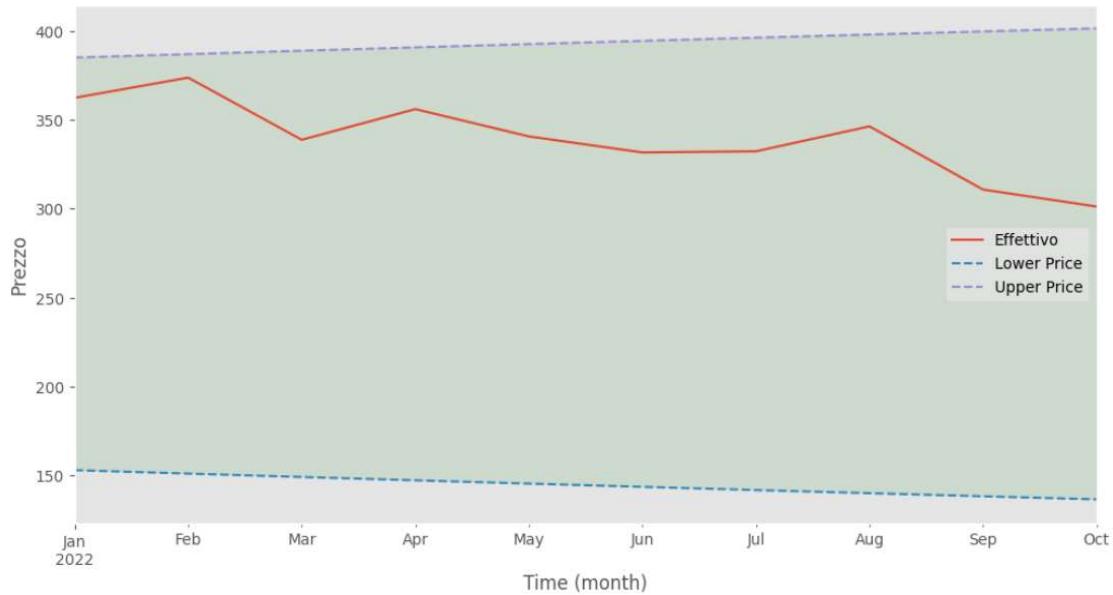
In questo caso invece, i prezzi effettivi sono solo in parte coerenti con il range di prezzi previsto, e in particolar modo siano vicini all' *Upper Price* previsto.

Chiamando il metodo *evaluate_models()* con i valori dei prezzi di **Pfizer**, ottengo la combinazione di parametri **(1, 0, 0)**. Mostro ora il grafico del confronto:



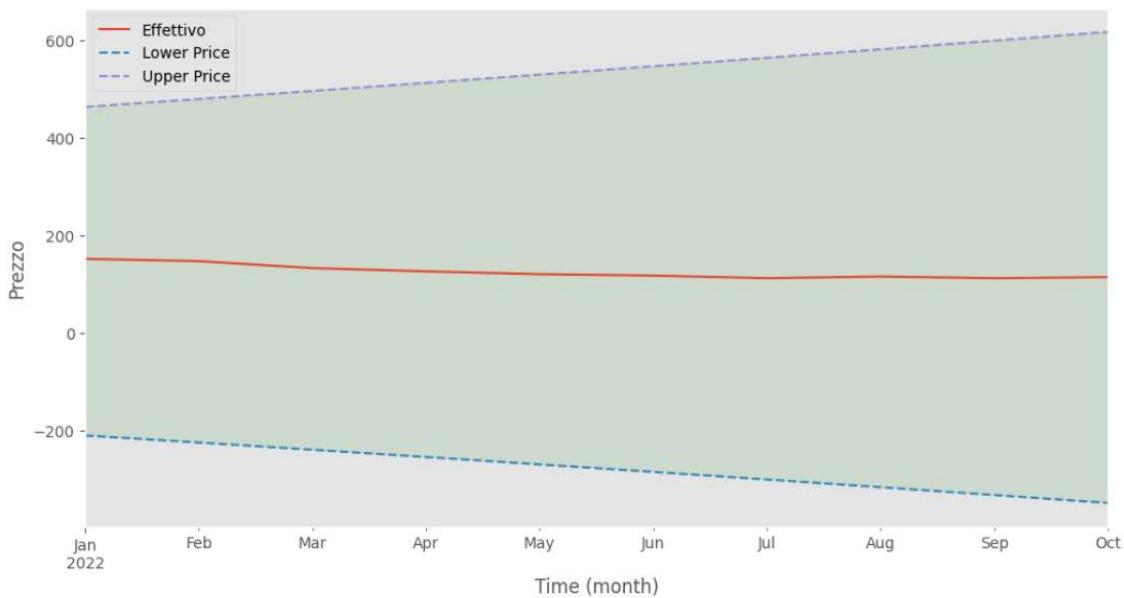
Il grafico mostra, come il modello sottostimi le prestazioni future offerte dall'azione di Pfizer, poiché i valori effettivi si muovono al di sopra dell'area di prezzo prevista dal modello. Da notare, però, che il range di prezzo previsto è limitato.

Faccio lo stesso con i valori dei prezzi di **Mastercard** e ottengo la combinazione di parametri **(2, 1, 1)**. Mostro ora il grafico del confronto:



Il grafico mostra che i valori effettivi rientrano nell'area di prezzo prevista nel futuro dal modello.

Chiamando il metodo *evaluate_models()* con i valori dei prezzi di **J.P Morgan**, ottengo la combinazione di parametri **(2, 2, 0)**. Mostro ora il grafico del confronto:



Il grafico mostra come, anche in questo caso, i valori effettivi rientrino nell'area di prezzo prevista dal modello. Da notare, però, come il range di prezzo previsto sia elevato in questo caso.

Capitolo 4: Strategie di trading e backtesting

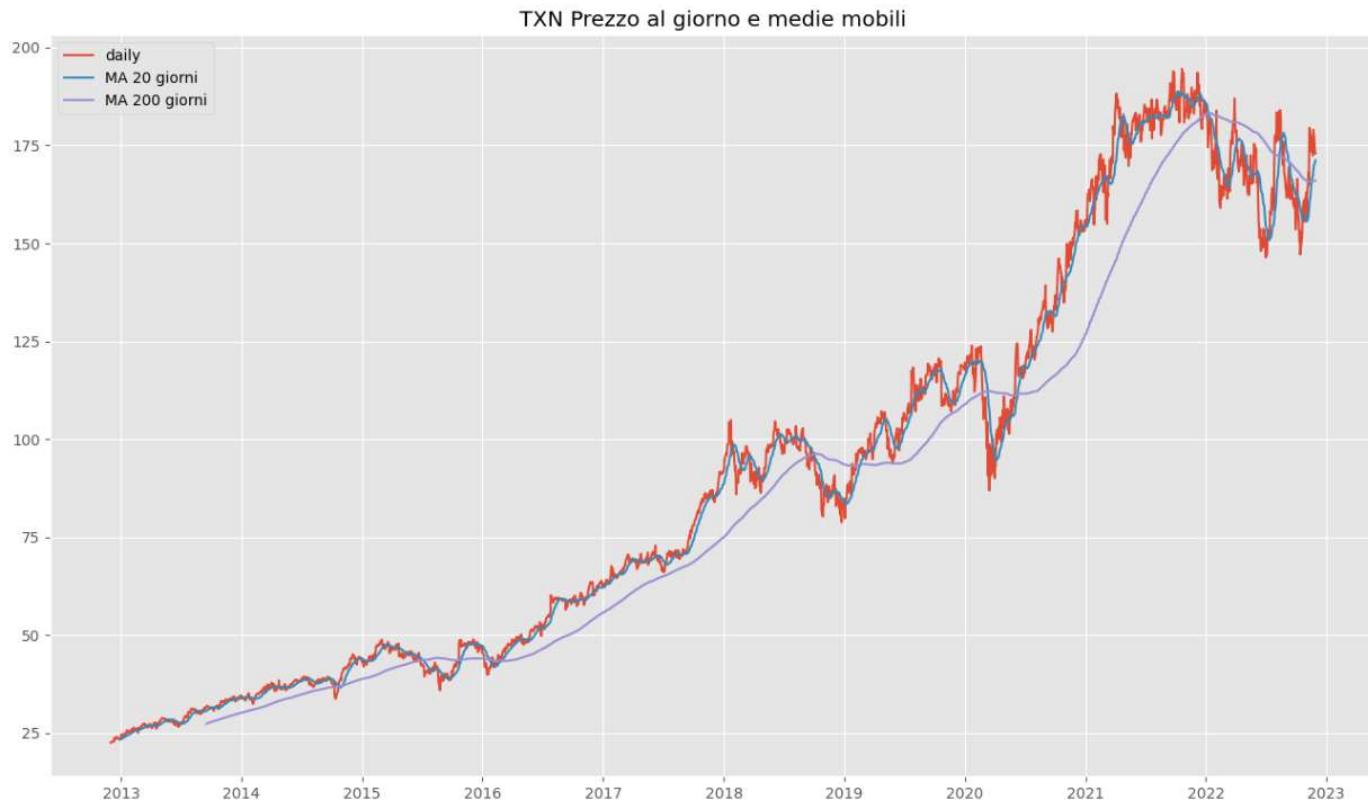
Ora analizzo una strategia di trading basata sul valore della **media mobile a 20 periodi** (veloce) rispetto alla **media mobile a 200 periodi** (lenta).

Questa tipologia di strategia è la più utilizzata sia dagli investitori di lungo termine sia da chi opera su *timeframe* più ridotti. Questo perché si tratta dello strumento più semplice ed efficace per determinare se siamo in una fase di trend rialzista o ribassista:

- se il prezzo della media mobile veloce taglia al rialzo la media mobile lenta ho un trend rialzista;
- se il prezzo della media mobile veloce taglia al ribasso la media mobile lenta ho un trend ribassista;
- se il prezzo della media mobile veloce taglia al rialzo e al ribasso la media mobile lenta più volte consecutivamente in un periodo di tempo relativamente breve ho un trend orizzontale (lateralizzazione del prezzo).

Per l'analisi utilizzo l'azione di **Texas Instruments**. Innanzitutto, calcolo le due medie mobili, a 20 e 200 periodi, e le mostro in un grafico, al quale aggiungo l'andamento giornaliero del prezzo.

```
TXN['MA20'] = TXN['Adj Close'].rolling(20).mean()  
TXN['MA200'] = TXN['Adj Close'].rolling(200).mean()
```



Analizzando questo grafico posso notare che, da metà del 2015 e fino a inizio del 2016, la media a 20 giorni ha toccato più volte alternativamente, dall'alto verso il basso e viceversa, la media a 200 giorni; infatti, il prezzo a inizio di questo periodo e a fine periodo è rimasto invariato.

Il grafico mostra anche che, da inizio 2016 fino al terzo trimestre del 2018 (circa tre anni) e da metà dell'anno 2020 fino a fine 2022 (due anni), si è verificato un forte trend rialzista che ha portato, nel primo caso il prezzo dai 40 dollari fino ai 90 dollari e nel secondo dai 110 dollari fino a 180 dollari di valore per azione. Entrambi questi periodi sono racchiusi dalla rottura a rialzo della media lenta da parte di quella più veloce e dalla rottura a ribasso della stessa.

Per riuscire a valutare in termini numerici la strategia utilizzata, ho bisogno di fare alcuni calcoli:

```
TXN['Price_yesterday'] = TXN['Adj Close'].shift(1)
TXN['Change'] = TXN['Adj Close'] / TXN['Price_yesterday']
TXN['Invested_SMA'] = [1 if TXN.loc[i, 'MA20'] > TXN.loc[i, 'MA200']
else 0 for i in TXN.index]
```

Tramite questo algoritmo sono in grado di ottenere una serie che indica tutti i punti in cui la media mobile a 20 giorni è superiore di quella a 200 giorni, con il risultato conseguito posso calcolare i ritorni ottenuti applicando questa strategia.

```
sma = TXN[TXN['Invested_SMA'] == 1]      #simple moving average
sma['Return'] = np.cumprod(sma['Change'])
sma['rtn'] = sma['Return'].pct_change()
```

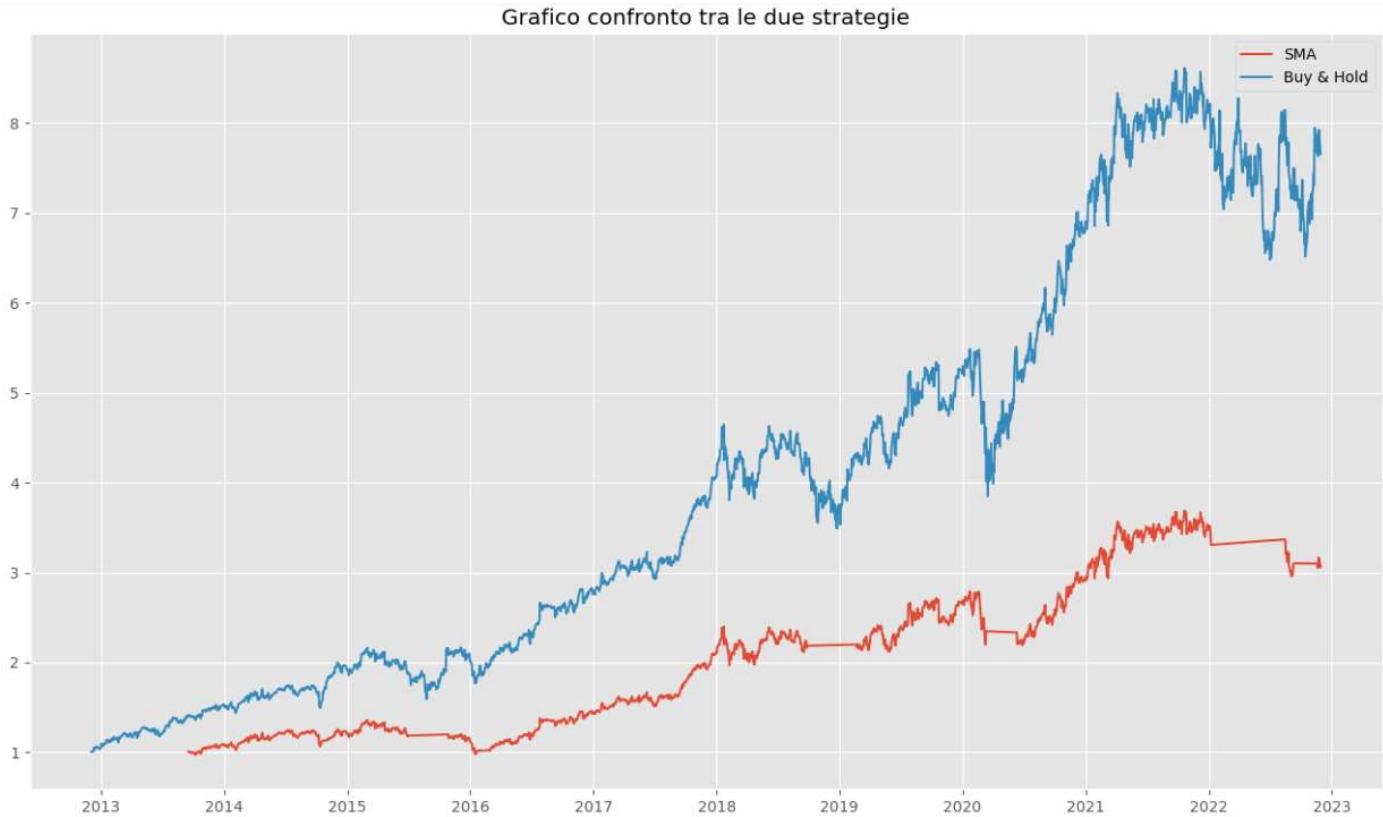
Da cui:

Ritorno strategia medie mobili: 3.06

Confrontiamo ora questo valore con il ritorno che avremmo ottenuto utilizzando la strategia **Buy and Hold**:

Ritorno strategia Buy_and_hold: 7.659

Posso concludere che usando questa seconda strategia avrei ottenuto un notevole aumento del ritorno finale; questo lo posso osservare anche nel grafico di confronto sotto:



Il vantaggio, che avrei potuto avere utilizzando la strategia delle medie mobili, sarebbe stato una volatilità annua inferiore:

Volatilità annua strategia medie mobili: 0.23

Volatilità annua Buy_and_hold: 0.27

Capitolo 5: CAPM

In questo capitolo analizzo il valore del **beta** (β) di ciascun titolo rispetto al mercato (indice S&P 500).

Ricordo che:

- con $\beta < 0$, l'asset si muove nella direzione opposta al mercato;
- con $\beta = 0$, l'asset e il mercato sono scorrelati;
- con $0 < \beta \leq 1$, l'asset si muove nella stessa direzione del mercato, non si sa nulla della sua volatilità;
- con $\beta > 1$, l'asset si muove nella stessa direzione del mercato con maggiore volatilità.

Il calcolo del beta è dato dalla covarianza del prezzo dell'asset rispetto al mercato diviso la varianza del mercato:

```
covariance = b1.cov().iloc[0,1]
benchmark_variance = b1.Market.var()
betaTXN = covariance / benchmark_variance
```

Da queste operazioni ottengo che:

```
Beta TXN: 1.057
Beta MSFT: 0.93
Beta JNJ: 0.623
Beta PFE: 0.758
Beta MA: 1.1314
Beta JPM: 1.164
```

Dati questi valori del beta, so che tutte le sei azioni si muovono nella stessa direzione dell'indice di mercato e in particolare che TXN, MA e JPM presentano anche una volatilità maggiore rispetto ad esso.

Ora calcolo l'esposizione di ciascun titolo ai fattori di rischio **Fama-French**, in modo da avere un ulteriore parametro per valutare un'azienda.

I tre fattori del modello di Fama e French sono:

- il fattore mercato, cioè il ritorno in eccesso del mercato (*MKT*).
- il fattore dimensione (*SMB*, *small minus big*) costruito come rendimento in eccesso delle azioni a piccola capitalizzazione rispetto alle grandi.
- il fattore valore (*HML*, *high minus low*) costruito come rendimento in eccesso delle azioni con un rapporto fra patrimonio e prezzo alto, e quelle con un rapporto fra patrimonio e prezzo basso

Per ogni azione ottengo il valore dei tre fattori (*MKT*, *SMB*, *HML*):

TXN:		MSFT:		JNJ:	
	coef		coef		coef
Intercept	0.0116	Intercept	0.0158	Intercept	0.0064
mkt	0.5261	mkt	0.4695	mkt	0.3169
smb	0.2661	smb	-0.0503	smb	-0.0942
hml	0.0471	hml	-0.1277	hml	0.0880

PFE:		MA:		JPM:	
	coef		coef		coef
Intercept	0.0037	Intercept	0.0126	Intercept	0.0073
mkt	0.4920	mkt	0.3798	mkt	0.4585
smb	-0.2136	smb	0.0849	smb	0.4567
hml	0.0097	hml	0.0397	hml	0.5088

Da questi dati posso concludere che il titolo più esposto, sia al rischio *SMB* che *HML*, è JPM con 0.4567 e 0.5088.

Utilizzando il beta calcolato in precedenza, è possibile calcolare il suo **rendimento atteso** utilizzando la formula:

$$R_i = R_f + \beta_i(R_{mkt} - R_f)$$

Dove:

- R_i è il rendimento atteso dell'investimento i
- R_f è il rendimento del *risk free* (che in questo caso è pari a zero)
- β_i è il beta dell'investimento i rispetto al mercato
- R_{mkt} è il rendimento del mercato

Ora procedo a calcolare il rendimento ad un anno del mercato; la previsione del prezzo la realizzo utilizzando un modello ARIMA.

Ottengo così due valori che sono il prezzo massimo previsto e il prezzo minimo previsto per il mercato:

Prezzo minimo previsto a un anno: 3013.46
 Prezzo massimo previsto a un anno: 4900.0

Con questi valori calcolo il ritorno semplice netto a un anno del prezzo del mercato:

Ritorno semplice netto considerando il prezzo minimo previsto: -0.2385
Ritorno semplice netto considerando il prezzo massimo previsto: 0.2381

Ed infine, utilizzando la formula, trovo il ritorno semplice netto previsto per l'azione di Johnson & Johnson:

Ritorno semplice netto considerando il prezzo minimo previsto dell'azione JNJ: -0.1488
Ritorno semplice netto considerando il prezzo massimo previsto dell'azione JNJ: 0.1485

Capitolo 6: Costruzione di portafoglio

In questo capitolo confronto vari portafogli; questi sono formati dalle sei azioni prese in considerazione con pesi differenti, in modo da trovare la combinazione di pesi che rappresenti il **portafoglio ottimale**. I diversi portafogli sono rappresentati su un grafico, che ha sull'asse delle x la varianza (il rischio) e sull'asse delle y la media (il ritorno atteso).

Nel grafico mostro anche una linea blu tratteggiata che raffigura la frontiera efficiente sotto forma di una parabola; il vertice di questa parabola rappresenta il punto al di sotto del quale i portafogli non sono più considerati ottimali, mentre al di sopra troviamo quelli che, dato un certo ritorno richiesto, offrono il minor valore in termini di rischio.

Prima di tutto considero i ritorni mensili dei primi 108 mesi di ogni azione:

```
df1 = df1.resample('M').mean()
df1 = df1.pct_change().dropna()
returns_df1 = df1.head(108)
```

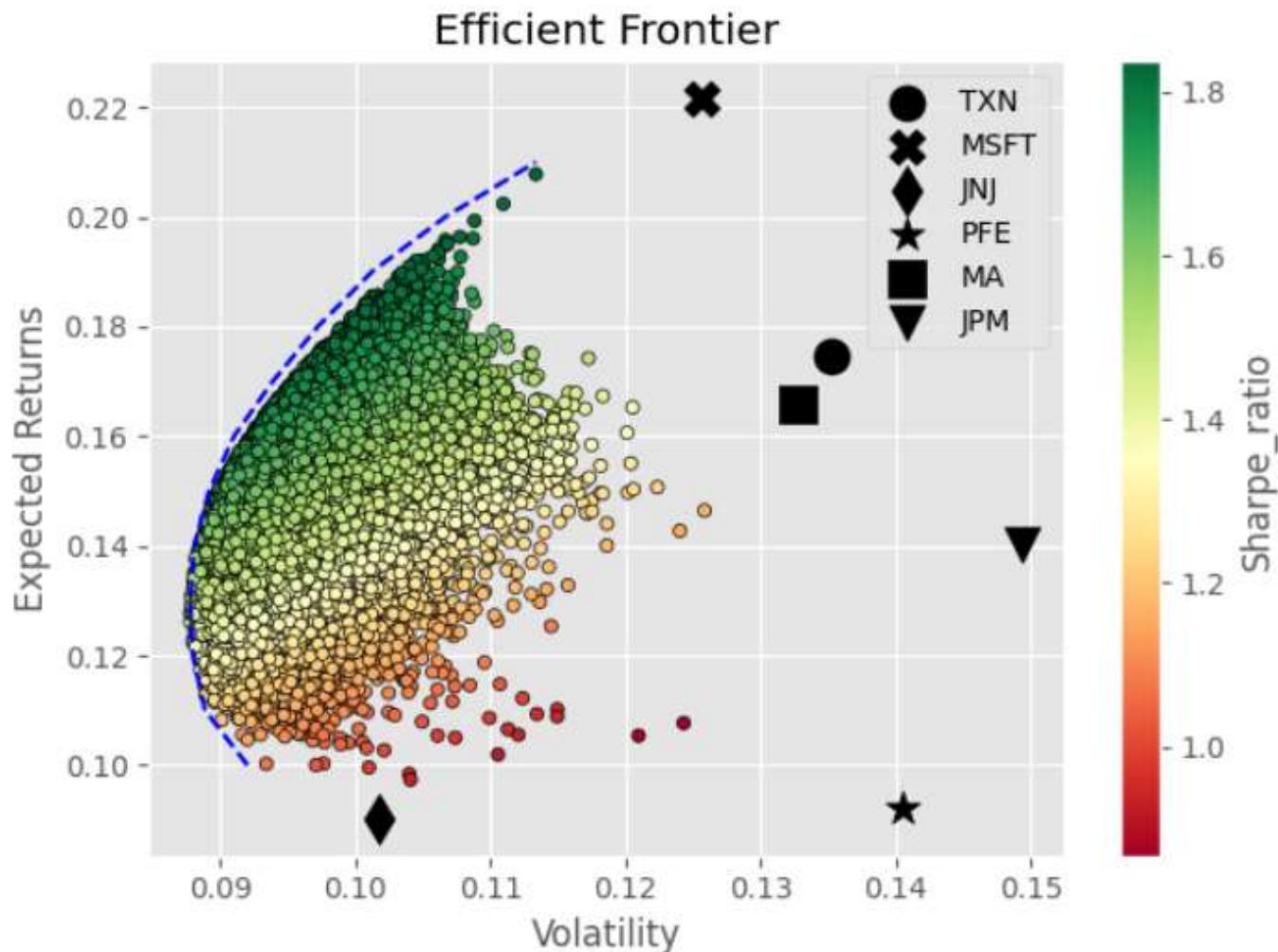
Successivamente creo tanti vettori dei pesi; ogni vettore deve avere come somma dei diversi elementi il valore uno.

```
weights = np.random.random(size=(n_port, n_assets))
weights /= np.sum(weights, axis=1)[:, np.newaxis]
```

Successivamente, tramite la funzione *dot()* creo i portafogli con le diverse combinazioni; calcolo per ognuno di questi il valore dei ritorni, della volatilità e lo *Sharpe ratio*, e unisco tutti i risultati in un unico *DataFrame*:

	Returns	Volatility	Sharpe_ratio
0	0.149993	0.091256	1.643651
1	0.143102	0.095127	1.504317
2	0.156247	0.102791	1.520051
3	0.144327	0.092225	1.564947
4	0.163417	0.094935	1.721346
...
99995	0.142212	0.092397	1.539148
99996	0.146500	0.099648	1.470173
99997	0.161801	0.096610	1.674788
99998	0.157513	0.100066	1.574096
99999	0.151810	0.094419	1.607840

Genero il grafico media-varianza:



Da notare che ad ogni portafoglio, rappresentato da un pallino, associo un colore in base al suo valore di *Sharpe*; mentre le figure nel grafico, descritte dalla legenda, mostrano il posizionamento dei portafogli formati dalle singole azioni.

Dopo aver ottenuto il grafico posso mostrare quale sia il portafoglio ottimale con il **miglior Sharpe ratio** e quello con la **minore volatilità**.

Portafoglio con il massimo Sharpe ratio:

Performance

Returns: 19.20% Volatility: 10.45% Sharpe_ratio: 183.72%

Weights

TXN: 15.08% MSFT: 59.54% JNJ: 5.24% PFE: 3.36% MA: 8.99% JPM: 7.78%

Portafoglio con la minore volatilità:

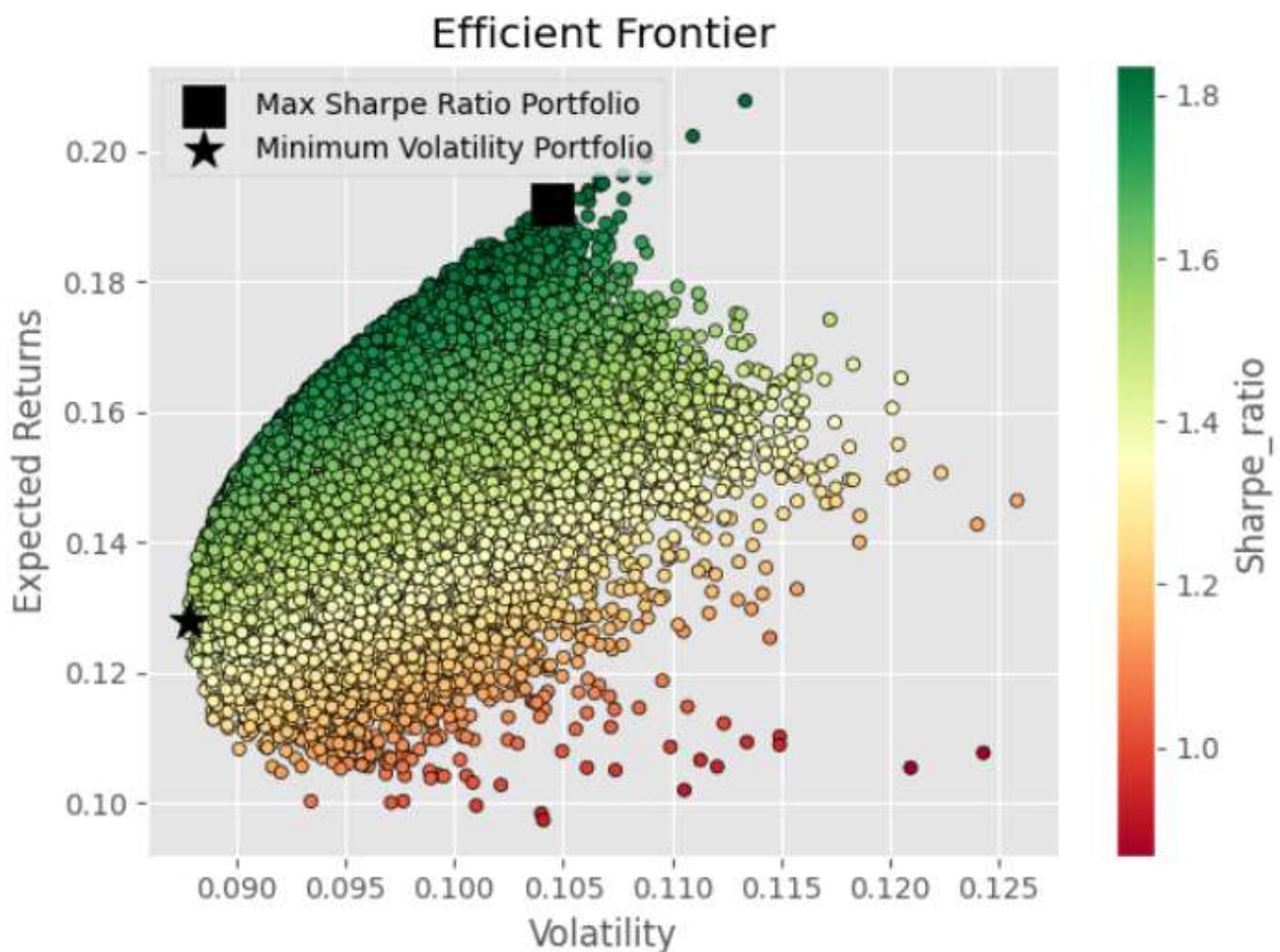
Performance

Returns: 12.78% Volatility: 8.78% Sharpe_ratio: 145.59%

Weights

TXN: 4.65% MSFT: 11.53% JNJ: 44.79% PFE: 11.47% MA: 17.79% JPM: 9.76%

Mostro i due portafogli nel grafico:



Per calcolare il **beta** del portafoglio ottimale (quello con il miglior *Sharpe ratio*) rispetto al mercato (S&P500), eseguo lo stesso calcolo effettuato nel capitolo 5:

```
SP500 = yf.download('^GSPC', start_date, end_date)
SP500_return = SP500['Adj Close'].resample('M').mean().pct_change().dropna()
benchmark_variance = SP500_return.var()
DF = SP500_return.to_frame().join(portfolio_opt_return.to_frame(), lsuffix = "SP500", rsuffix = "PORTFOLIO")
covariance = DF.cov().iloc[0,1]
beta = covariance / benchmark_variance
```

E ottengo un valore del **β** pari a:

$$\text{Beta: } 0.866$$

Questo valore indica che il portafoglio si muove nella stessa direzione del mercato, senza però avere informazioni rispetto alla sua volatilità.

Ora confronto il valore dei ritorni del portafoglio **ottimale** con quello **effettivo**.

Creo il portafoglio effettivo con la combinazione delle diverse azioni prese con lo stesso peso:

```
weight_portfolio = [0.1666, 0.1666, 0.1666, 0.1666, 0.1666, 0.1666]
portfolio_return = returns_df1.dot(weight_portfolio)
```

E ottengo che:

Ritorni medi annui portafoglio effettivo: 0.1474

Ritorni medi annui portafoglio ottimale: 0.1920

