

Progetto di Business Intelligence per i Servizi Finanziari

Banfi Michele 869294

January 3, 2023

Chapter 1

Sommario dei dati utilizzati

1.1 Titoli

I titoli scelti per questo progetto sono i seguenti:

Settore tecnologico

Netflix, Inc. (**NFLX**)

Meta Platforms, Inc. (**META**)

La scelta di Netflix é stata basata sulle notizie di questi ultimi anni della perdita di utenti e del raggiungimento di utenti massimi che Netflix ha riscontrato e il conseguente calo del titolo sul mercato. Mentre Meta é stata presa in considerazione per il recente cambio di brand e nome, da Facebook a Meta, successivo al lancio del Metaverso

Settore finanziario

The Goldman Sachs Group, Inc. (**GS**)

Citigroup Inc. (**C**)

Le scelte di queste due aziende sono dovuto al loro andamento dopo un calo, infatti Goldman Sachs, dopo il calo del covid si é subito ripresa e ha raddoppiato il suo valore in meno di un anno. Mentre Citigroup dopo la crisi finanziaria del 2008 non ha mai raggiunto di nuovo i valori che aveva precedentemente

Settore consumer defensive

The Coca-Cola Company (**KO**)

PepsiCo, Inc. (**PEP**)

Coca-Cola e PepsiCo sono stati scelti in quanto concorrenti allo stesso mercato e in quando non hanno subito un calo dopo il covid-19 e inoltre, il loro andamento nel mercato sembra essere costante in rialzo.

1.2 Scaricare i dati

Per prima cosa importiamo le librerie che ci serviranno in questo progetto:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import yfinance as yf
```

Procediamo al dowload dei dati tramite `yfinance.download()`

```
1 start = "2012-11-30"
2 end = "2022-11-30"
3
4 #Tech
5 NFLX = yf.download("NFLX", start, end)
6 META = yf.download("META", start, end)
7
8 #Financials
9 GS = yf.download("GS", start, end)
10 C = yf.download("C", start, end)
11
12 #Consumers
```

```

13     KO = yf.download("KO", start, end)
14     PEP = yf.download("PEP", start, end)

```

1.3 Fusione dei dati

Ora procediamo alla fusione dei dati in un unico DataFrame e di ogni titolo prendiamo solo l'Adj close".

```

1     market = pd.concat([NFLX['Adj Close'], META['Adj Close'], GS['Adj Close'], C['Adj
    Close'], KO['Adj Close'], PEP['Adj Close']], axis=1)
2     market.columns = ['NFLX', 'META', 'GS', 'C', 'KO', 'PEP']

```

1.4 Presentazione dei dati

Per mostrare la struttura del nostro DataFrame possiamo usare la seguente funzione che ci ritorna le righe iniziali del nostro DataFrame

```

1     market.head()

```

	NFLX	META	GS	C	KO	PEP
Date						
2012-11-30	11.672857	28.000000	99.521103	28.667482	27.682371	52.271511
2012-12-03	10.857143	27.040001	100.036469	28.377247	27.288162	52.018406
2012-12-04	12.378571	27.459999	98.498741	28.435295	27.120255	52.010952
2012-12-05	11.910000	27.709999	98.963432	30.234777	27.237057	52.301300
2012-12-06	12.310000	26.969999	99.022560	30.699169	27.288162	52.533886


E inoltre possiamo procedere anche alla stampa di un grafico di questi dati:

```

1     market.plot(grid=True, title="Markets Adjs Closes")
2     market.plot(grid=True, title="Markets Adjs Closes", subplots=True)

```

market.png



marketSubplots.png

Chapter 2

Statistiche descrittive

2.1 Rendimenti annui

Per calcolare il rendimento **composto** e **cumulato annuale** dobbiamo prima trasformare i dati da giornalieri ad annuali nel seguente modo:

```
1 NFLXy = NFLX.groupby(pd.Grouper(freq='Y')).last()
```

Li raggruppiamo e li salviamo in `marketY`. Dopodiché possiamo procedere con il calcolo dei **rendimenti cumulati annui** e quello dei **rendimenti composti annui**:

```
1 #Rendimento cumulato dei nostri titoli
2 marketY2 = marketY.pct_change(1)
3 marketY2 = marketY2.dropna()
4 marketY3 = np.cumprod(marketY2 + 1)
5
6 #Rendimento composto annuale NFLX
7 rcoNFLXy = ((marketY['NFLX'][-1]/marketY['NFLX'][0])** (1/marketY['NFLX'].count())
  - 1) * 100
```

Applichiamo questo procedimento per tutti i titoli.

Questo é il risultato dei nostri rendimenti cumulati:

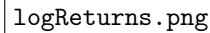
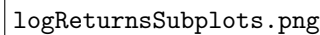
	NFLX	META	GS	C	KO	PEP
Date						
2013-12-31	3.976347	2.052968	1.407936	1.318358	1.172331	1.246481
2014-12-31	3.689491	2.930879	1.559786	1.370062	1.234067	1.461696
2015-12-31	8.647370	3.931630	1.469481	1.314142	1.297496	1.589572
2016-12-31	9.359542	4.321938	1.982669	1.523134	1.292863	1.713495
2017-12-31	14.512582	6.628851	2.136010	1.934785	1.478803	2.018807

2.2 Rendimenti semplici e logaritmici

Usiamo le seguenti formule per calcolarci i **rendimenti semplici** e **logaritmici**

```
1 #Rendimento semplice
2 rsNFLX = NFLX['Adj Close'] / NFLX['Adj Close'].shift(1)
3 #Rendimento logaritmico
4 rlnNFLX = np.log(NFLX['Adj Close'] / NFLX['Adj Close'].shift(1))
```

Applichiamo le stesse formule per tutti i titoli e li uniamo in un singolo DataFrame ottenendo i seguenti grafici

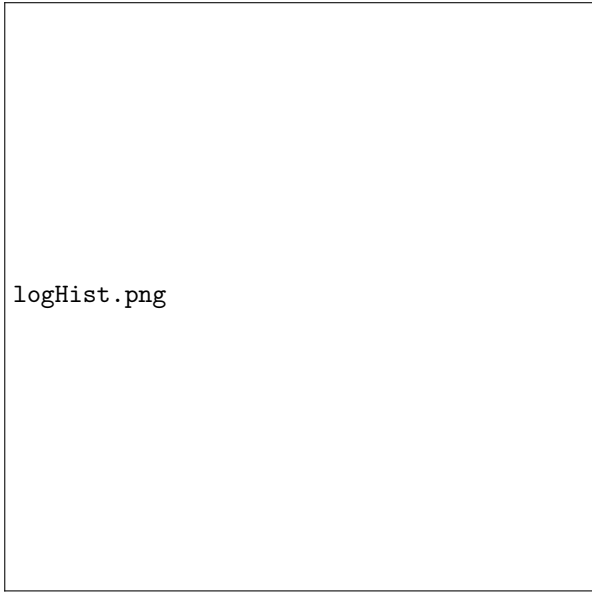
A placeholder for a plot titled 'logReturns.png'.A placeholder for a plot titled 'logReturnsSubplots.png'.

Possiamo subito notare come le serie dei rendimenti possano darci una diversa prospettiva sull'andamento dei titoli rispetto alla sola serie dei prezzi, infatti ci permettono di confrontare diversi titoli aventi prezzi molto differenti e quindi di concentrarci sull'andamento e non sul valore. Nel primo grafico Netflix (blu) e Meta(arancione) sono i due titoli con più outliers rispetto agli altri titoli e che di conseguenza hanno avuto dei movimenti di prezzo più ampi e forti (sia in positivo che in negativo) rispetto agli altri titoli. Inoltre nel secondo grafico si nota l'importanza che ha avuto la crisi causata dall'epidemia di covid-19 sui mercati nel 2020, tranne che nei casi su Netflix e Meta. Che però hanno avuto dei cali dopo il 2020, Netflix ad esempio aveva divulgato la notizia della perdita di abbonati alla piattaforma.

2.3 Rendimenti con istogrammi

Plottiamo ora i rendimenti tramite gli istogrammi con il seguente metodo:

```
1 plt.hist(r1NFLX, density=True, bins=50)
2 plt.title('NFLX Logarithmic returns histogram')
```



logHist.png

Tramite gli istogrammi possiamo valutare la quantità di volte che si sono presentati i rendimenti nella serie, e si nota subito che per un titolo come Golden Sachs si ha un istogramma più largo rispetto a Netflix ad esempio, questo indica una maggiore stazionarietà del titolo e di conseguenza dei cambi di prezzo più lievi.

2.4 Grafici diagnostici

Per studiare meglio i rendimenti aggiungiamo altri 3 grafici all'istogramma, la kernel density, boxplot e qq-plot.


- Box Plot -> In un solo grafico ci mostra: mediana, media primo quartile, terzo quartile e la distanza tra essi, e infine ci mostra anche gli outliers
- QQ-Plot -> Ci mostra quanto una serie è prossima alla normalità. Una serie normale ha la linea del qq plot esattamente in diagonale
- Kernel density -> È una curva addolcita dell'istogramma

Il codice per produrre i grafici è il seguente:

```
1      #NFLX
2      fig = plt.figure(figsize=(15, 15))
3
4      ax=fig.add_subplot(221)
5      plt.hist(r1NFLX, density=True, bins=50)
6      plt.title('NFLX Log returns histogram')
7
8      ax=fig.add_subplot(222)
9      plt.boxplot(r1NFLX)
10     plt.title('NFLX Log returns Boxplot')
11     r1NFLX.to_frame().boxplot()
12
13     ax=fig.add_subplot(223)
14     sm.graphics.qqplot(r1NFLX, line='s', ax=ax)
15     plt.title('NFLX Log returns Q-Q plot')
16
17     ax=fig.add_subplot(224)
18     r1NFLX.plot.density()
```

Il codice soprastante è il codice per produrre i grafici di Netflix, in modo equivalente possiamo produrre i grafici per gli altri titoli.

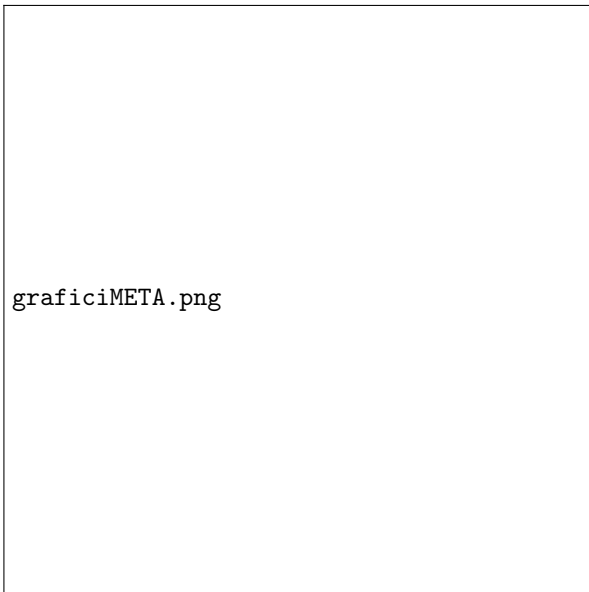
2.4.1 NFLX



graficiNFLX.png

Netflix presenta una distribuzione molto compatta e ricca di outliers, come si può vedere nel BoxPlot.

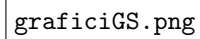
2.4.2 META



graficiMETA.png

Anche META resta molto distante dalla normalità e resta comunque molto ricca di outliers, più vicini però alla media rispetto a NFLX.

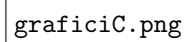
2.4.3 GS

A large rectangular box with a thin black border, intended for a plot or figure. The text 'graficiGS.png' is centered within the box.

graficiGS.png

Goldach Sech é il titolo tra i sei che si avvicina leggermente un pó di piú alla normalit  ma ne resta comunque molto lontano.

2.4.4 C

A large rectangular box with a thin black border, intended for a plot or figure. The text 'graficiC.png' is centered within the box.

graficiC.png

Citigroup, come Netflix ha molti outliers distanti dalla media

2.4.5 KO



Anche Coca-Cola si avvicina a Goldman Sachs per vicinanza alla normalità ma sempre rimanendo ricchissimo di outliers. E inoltre é la serie più spostata rispetto al centro, quindi con una skewness maggiore

2.4.6 PEP



Pepsi, rispetto al suo concorrente presenta la maggior parte dei suoi outliers vicini alla media e pochi lontani

2.5 Statistiche descrittive univariate

Ora procediamo con il calcolo dei seguenti dati:

- Media
- Varianza
- Deviazione standard

- Asimmetria
- Curtosi Per calcolare i dati usiamo le seguenti funzioni sulle serie:

```

1      #NFLX
2      meanNFLX = round(rlNFLX.mean(), 4)
3      varNFLX = round(rlNFLX.var(), 4)
4      stdNFLX = round(rlNFLX.std(), 4)
5      skewNFLX = round(rlNFLX.skew(), 4)
6      kurtNFLX = round(rlNFLX.kurtosis(), 4)

```

Calcolati tutti i valori li uniamo in un unico DataFrame così che possiamo confrontarli

	NFLX	META	GS	C	KO	PEP
Mean	0.0013	0.0005	0.0005	0.0002	0.0003	0.0005
Variance	0.0009	0.0006	0.0003	0.0004	0.0001	0.0001
Standard Deviation	0.0300	0.0241	0.0176	0.0203	0.0115	0.0115
Skewness	-0.4118	-1.1173	-0.1999	-0.4763	-0.8931	-0.5869
Kurtosis	31.0142	27.8449	9.8418	14.9718	10.6779	23.7351

Il titolo con il rendimento più alto è Netflix che ha una media di 0.0013 mentre quello con il rendimento più basso è Citigroup con 0.0002. Mentre per quanto riguarda la deviazione standard, il titolo più alto è sempre Netflix con 0.03 mentre quelli più bassi sono Coca-Cola e Pepsi con 0.0115 [...]

2.6 Varianza e covarianza

Calcoliamo ora la matrice di covarianza e la covarianza tra i nostri titoli con le seguenti funzioni:

```

1      #Covariance Matrix
2      covariance = rlMarket.cov().iloc[0,1]
3      covarianceMatrix = rlMarket.cov()

```

Che ci restituiscono per la covarianza: 0.0003, mentre per la matrice:

	NFLX	META	GS	C	KO	PEP
NFLX	0.000902	0.000298	0.000147	0.000157	0.000047	0.000068
META	0.000298	0.000583	0.000157	0.000164	0.000062	0.000082
GS	0.000147	0.000157	0.000311	0.000296	0.000085	0.000084
C	0.000157	0.000164	0.000296	0.000414	0.000102	0.000092
KO	0.000047	0.000062	0.000085	0.000102	0.000131	0.000096
PEP	0.000068	0.000082	0.000084	0.000092	0.000096	0.000132

Procediamo con il calcolo della matrice di correlazione con il seguente codice:

```

1      correlation = rlMarket.corr()
2      plt.figure(figsize=(15,15))
3      plt.title('Matrice di correlazione')
4      sns.heatmap(correlation, vmax=1, square=True, annot=True, cmap='YlGnBu')

```

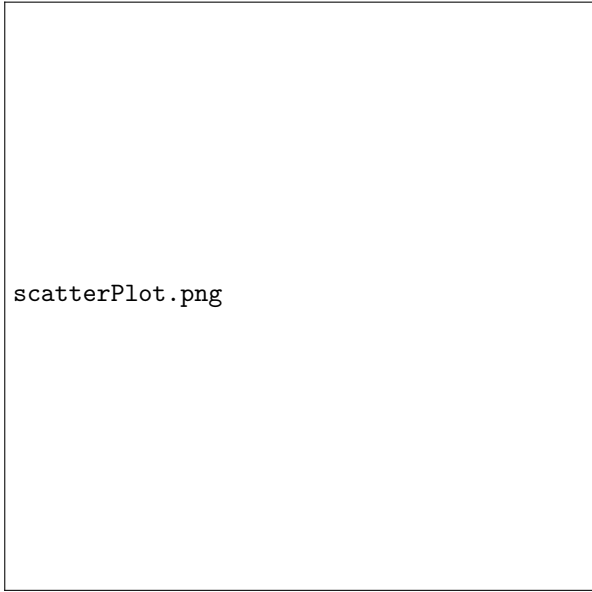


correlationMatrix.png

Proseguiamo quindi con la costruzione degli scatter plot

```
1 plt.figure(figsize=(15,15))
2 pd.plotting.scatter_matrix(rlMarket, figsize=(12, 12))
3 plt.show()
```

Con la seguente immagine:



scatterPlot.png

[...]

Chapter 3

Analisi di previsione

Per fare l'analisi di previsione ho deciso di usare una Support Vector Machine (SVM). Per prima cosa prendiamo i dati e li raggruppiamo mensilmente con il seguente codice:

```
1 NFLXm = NFLX.groupby(pd.Grouper(freq='M')).last()
2 METAm = META.groupby(pd.Grouper(freq='M')).last()
3 GSmm = GS.groupby(pd.Grouper(freq='M')).last()
4 Cm = C.groupby(pd.Grouper(freq='M')).last()
5 KOm = KO.groupby(pd.Grouper(freq='M')).last()
6 PEPm = PEP.groupby(pd.Grouper(freq='M')).last()
```

Adesso possiamo procedere alla creazione delle "prediction". Quello che vogliamo ottenere è la predizione del prossimo mese di conseguenza la riga "prediction" è lo shift dei prezzi di una unità.

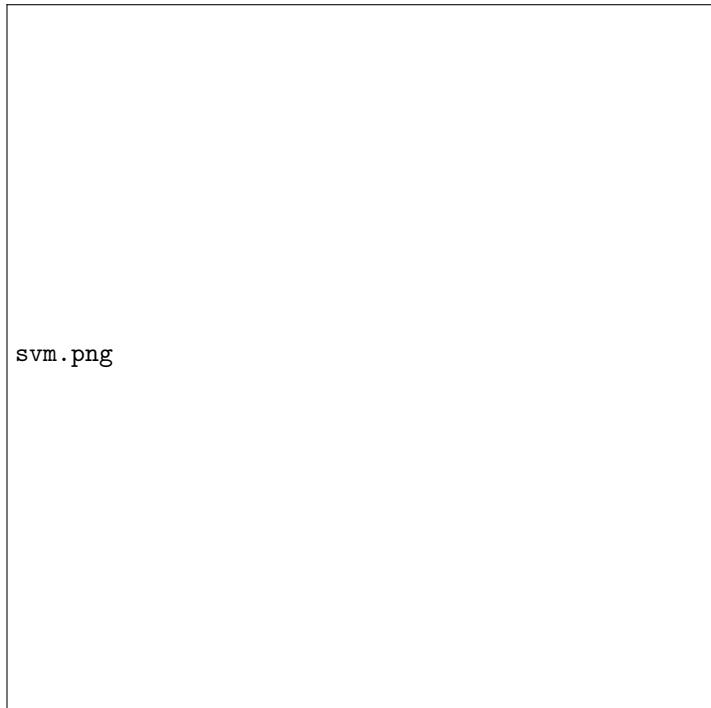
```
1 df = GSmm[['Adj Close']]
2 forecast_out = 1
3 monthPrevision = 10
4 df['prediction'] = df[['Adj Close']].shift(-forecast_out)
5 x = np.array(df.drop(['prediction'], 1))
6 x = x[:-forecast_out]
7 y = np.array(df['prediction'])
8 y = y[:-forecast_out]
```

Ora andiamo avanti dividendo il nostro dataset in due, uno di allenamento e uno di test. In particolare le dimensioni dei due dataset sono 80 e 30 rispettivamente. Poi alleniamo la nostra SVM con il metodo .fit() e otteniamo un primo valore che è quello di confidenza che è pari a 0.5844

```
1 #usiamo 30 mesi per il test e 80 per il train
2 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=30, train_size
=80)
3 svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
4 svr_rbf.fit(x_train, y_train)
5 svm_confidence = svr_rbf.score(x_test, y_test)
6 print("svm confidence: ", round(svm_confidence, 4))
```

Prendiamo gli ultimi 10 mesi dei nostri prezzi per confrontare le previsioni rispetto ai valori effettivi e mostriamoli a grafico

```
1 #prendiamo gli ultimi 10 mesi per fare la previsione
2 x_mesi = np.array(df.drop(['prediction'], 1))[-monthPrevision:]
3 y_mesi = np.array(df[['Adj Close'], 1))[-monthPrevision:]
4
5 #effettuiamo la previsione
6 svm_prediction = svr_rbf.predict(x_mesi)
7 plt.figure(figsize=(40,15))
8 plt.plot(y_mesi, color='red', label='Real GS Stock Price')
9 plt.plot(svm_prediction, color='blue', label='Predicted GS Stock Price')
10 plt.title('GS Stock Price Prediction')
11 plt.xlabel('Time')
12 plt.ylabel('GS Stock Price')
13 plt.legend()
14 plt.show()
```



svm.png

Chapter 4

Strategia di trading e backtesting

Come strategia di trading ho deciso di adottare l'indicatore **MACD** il cui acronimo sta per **Moving Average Convergence Divergence** che é un trend-following momentum indicator. Per calcolare questo indicatore sono necessarie tre Medie Mobili Esponenziali (**EMA**). La prima media mobile, quella più veloce, viene calcolata a 12 periodi; quella più lenta invece è a 26 periodi. Queste due medie mobili vengono sottratte fra loro per calcolarne la differenza che sarà quindi rappresentata graficamente da una sola linea (MACD Line). Per generare segnali è stata introdotta la terza linea, ovvero una media mobile esponenziale, solitamente a 9 periodi, della precedente differenza (Signal Line). Per generare un segnale di acquisto lungo, la MACD Line deve incrociare al rialzo la Signal Line.

Confronteremo la seguente strategia con una strategia Buy-and-Hold e con un strategia di incrocio di due Medie Mobili Semplici (**SMA**) sul prezzo.

4.1 Costruzione della strategia

Quindi per prima cosa calcoliamo gli indicatori

```
1 data = NFLX.drop(columns=['Open', 'High', 'Low',])
2 data['SMA20'] = data['Adj Close'].rolling(20).mean()
3 data['SMA120'] = data['Adj Close'].rolling(120).mean()
4 exp1 = data['Close'].ewm(span=12, adjust = False).mean()
5 exp2 = data['Close'].ewm(span=26, adjust = False).mean()
6 data['MACD'] = exp1 - exp2
7 data['Signal line'] = data['MACD'].ewm(span=9, adjust = False).mean()
8 data.dropna()
```

Poi procediamo al calcolo dei segnali di acquisto e dei rendimenti delle strategia SMA e MACD

```
1 data['Price_yesterday'] = data['Adj Close'].shift(1)
2 data['Change'] = data['Adj Close'] / data['Price_yesterday']
3
4 #segnale acquisto strategia SMA
5 data['Invested_SMA'] = [1 if data.loc[i, 'SMA20'] > data.loc[i, 'SMA120']
6                        else 0 for i in data.index]
7
8 #segnale acquisto strategia MACD
9 data['Invested_MACD'] = [1 if data.loc[i, 'MACD'] > data.loc[i, 'Signal line']
10                        else 0 for i in data.index]
11 data = data.dropna()
12
13 #calcolo strategia SMA
14 sma = data[data['Invested_SMA'] == 1]
15 sma['Return'] = np.cumprod(sma['Change'])
16 sma['rtn'] = sma['Return'].pct_change()
17 sma['rtn'].std()*np.sqrt(252)
18 sma['rtn'].mean()*252 / (sma['rtn'].std()*np.sqrt(252))
19
20 #calcolo strategia MACD
21 macd = data[data['Invested_MACD'] == 1]
22 macd['Return'] = np.cumprod(macd['Change'])
23 macd['rtn'] = macd['Return'].pct_change()
```

```

24     macd['rtn'].std()*np.sqrt(252)
25     macd['rtn'].mean()*252 / (macd['rtn'].std()*np.sqrt(252))

```

4.2 Misurazione efficienza della strategia

Ora calcoliamo il rendimento di una strategia Buy-and-Hold con il seguente codice

```

1     data['Buy_and_hold'] = np.cumprod(data['Change'])
2     data['rtn'] = data['Buy_and_hold'].pct_change()
3     data['rtn'].std()
4     data['rtn'].mean()*252 / (data['rtn'].std()*np.sqrt(252))

```

Mostriamo graficamente ora i dati con il seguente codice in 4 diversi grafici:

```

1     plt.figure(figsize=(20,24))
2     ax1 = plt.subplot2grid((12,1), (0,0), rowspan = 4, colspan = 1, title = 'NFLX')
3     ax2 = plt.subplot2grid((12,1), (4,0), rowspan = 2, colspan = 1, sharex = ax1)
4     ax3 = plt.subplot2grid((12,1), (6,0), rowspan = 2, colspan = 1, sharex = ax1)
5     ax4 = plt.subplot2grid((12,1), (8,0), rowspan = 2, colspan = 1, sharex = ax1)
6
7     ax1.plot(data['Adj Close'], label = 'Price')
8     ax1.plot(data['SMA20'], label = 'SMA20')
9     ax1.plot(data['SMA120'], label = 'SMA120')
10
11     ax2.bar(data.index, data['Volume'], label = 'Volume')
12
13     ax3.plot(data['MACD'], label = 'MACD')
14     ax3.plot(data['Signal line'], label = 'Signal line')
15
16     ax4.plot(data['Buy_and_hold'], label = 'Buy and hold')
17     ax4.plot(sma['Return'], label = 'SMA')
18     ax4.plot(macd['Return'], label = 'MACD')
19     plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
20     ax4.set_xlabel('Date (Year - month)')
21
22     ax1.legend()
23     ax2.legend()
24     ax4.legend()
25     ax3.legend()
26     plt.show()

```


MACD .png

Chapter 5

Capital Asset Pricing Model

Per prima cosa scarichiamo i dati dell'indice SP 500 insieme ai titoli scelti in precedenza e sistemiamo il DataFrame

```
1 marketBenchmark = '^GSPC'
2 df = yf.download(['NFLX', 'META', 'GS', 'C', 'KO', 'PEP', marketBenchmark], start,
3 end)
4 marketCloses = df['Adj Close']
5 x = marketCloses.pct_change().dropna()
```

Ora possiamo procedere al calcolo del Beta di ciascun titolo rispetto al mercato. Il seguente codice mostra il metodo utilizzato per Netflix, analogamente lo si può usare per gli altri titoli:

```
1 #NFLX and S&P500 Beta
2 NFLXMarket = pd.concat([x['NFLX'], x['^GSPC']], axis=1)
3 covNFLX = NFLXMarket.cov().iloc[0,1]
4 marketVariance = x['^GSPC'].var()
5 betaNFLX = covNFLX/marketVariance
```

Ottenendo i seguenti valori:

	NFLX	META	GS	C	KO	PEP
Beta	1.154346	1.234385	1.204651	1.348596	0.633521	0.674546

5.1 Fattori di rischio Fama French

I tre fattori del modello di Fama e French (*) sono:

- il fattore mercato, cioè la dipendenza dall'andamento del mercato azionario (**MKT**)
- il fattore dimensione (**SMB**) costruito come rendimento in eccesso delle azioni a piccola capitalizzazione rispetto alle grandi
- il fattore valore (**HML**) costruito come rendimento in eccesso delle azioni con un rapporto fra patrimonio e prezzo alto (Value stocks) e quelle con un rapporto fra patrimonio e prezzo basso (Growth stocks)

Come prima cosa, per poter scaricare i dati del fattore Fama French importiamo la libreria `pandas_datareader` nel seguente modo

```
1 import pandas_datareader as pdr
```

Poi procediamo con il download dei dati con la seguente funzione

```
1 factor_df = pdr.famafrench.FamaFrenchReader('F-F_Research_Data_Factors', start,
2 end).read()[0]
```

I valori `start` e `end` sono i valori di inizio e fine del nostro periodo di analisi. Il nostro DataFrame `factor_df` si presenta nel seguente modo:

	mkt	smb	hml	rf
Date				
2012-11	0.0078	0.0064	-0.0084	0.0001
2012-12	0.0118	0.0150	0.0351	0.0001
2013-01	0.0557	0.0033	0.0096	0.0000
2013-02	0.0129	-0.0028	0.0011	0.0000
2013-03	0.0403	0.0081	-0.0019	0.0000

5.1.1 NFLX Fama-French

Procediamo poi al calcolo della regressione per i valori di Netflix con il seguente codice:

```

1  #NFLX Fama French risk exposure
2  y = NFLXm['Adj Close'].pct_change().dropna()
3
4  #sistemiamo i mesi, per farli combaciare
5  factor_df = factor_df.tail(-1)
6  y = y.head(-1)
7
8  #sistemiamo le date allo stesso formato
9  y.index = y.index.strftime('%Y-%m')
10 factor_df.index = factor_df.index.strftime('%Y-%m')
11
12 #rinominiamo la colonna
13 y.name = 'rtn'
14
15 #uniamo i due dataframe
16 ff_data = factor_df.join(y)
17
18 #calcoliamo l'Excess Return
19 ff_data['excess_rtn'] = ff_data.rtn - ff_data.rf
20
21 #effettuiamo la regressione
22 ff_model = smf.ols(formula = 'excess_rtn ~ mkt + smb + hml', data = ff_data).fit()

```

Il soprastante codice produce il seguente output:

Dep. Variable:	excess_rtn	R-squared:	0.216			
Model:	OLS	Adj. R-squared:	0.195			
Method:	Least Squares	F-statistic:	10.54			
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	3.53e-06			
Time:	14:53:46	Log-Likelihood:	77.670			
No. Observations:	119	AIC:	-147.3			
Df Residuals:	115	BIC:	-136.2			
Df Model:	3					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.0221	0.012	1.824	0.071	-0.002	0.046
mkt	1.3657	0.281	4.858	0.000	0.809	1.923
smb	-0.0198	0.483	-0.041	0.967	-0.977	0.937
hml	-0.8983	0.334	-2.693	0.008	-1.559	-0.238
Omnibus:	59.945	Durbin-Watson:	1.717			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	296.596			
Skew:	1.655	Prob(JB):	3.94e-65			
Kurtosis:	9.991	Cond. No.	42.0			

Analogamente calcoliamo l'esposizione di ogni titolo ai fattori di rischio

5.1.2 META Fama-French

Dep. Variable:	excess_rtn	R-squared:	0.243			
Model:	OLS	Adj. R-squared:	0.223			
Method:	Least Squares	F-statistic:	12.30			
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	4.90e-07			
Time:	15:04:42	Log-Likelihood:	120.19			
No. Observations:	119	AIC:	-232.4			
Df Residuals:	115	BIC:	-221.3			
Df Model:	3					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	0.0029	0.008	0.348	0.729	-0.014	0.020
mkt	1.1127	0.197	5.658	0.000	0.723	1.502
smb	-0.3803	0.338	-1.125	0.263	-1.050	0.289
hml	-0.5585	0.233	-2.394	0.018	-1.021	-0.096
Omnibus:	24.295	Durbin-Watson:	1.707			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	166.493			
Skew:	0.216	Prob(JB):	7.02e-37			
Kurtosis:	8.779	Cond. No.	42.0			

5.1.3 GS Fama-French

Dep. Variable:	excess_rtn	R-squared:	0.676			
Model:	OLS	Adj. R-squared:	0.667			
Method:	Least Squares	F-statistic:	79.93			
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	5.23e-28			
Time:	15:17:45	Log-Likelihood:	199.72			
No. Observations:	119	AIC:	-391.4			
Df Residuals:	115	BIC:	-380.3			
Df Model:	3					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.0006	0.004	-0.127	0.899	-0.009	0.008
mkt	1.3185	0.101	13.081	0.000	1.119	1.518
smb	0.2460	0.173	1.419	0.158	-0.097	0.589
hml	0.6648	0.120	5.559	0.000	0.428	0.902
Omnibus:	3.869	Durbin-Watson:	2.068			
Prob(Omnibus):	0.144	Jarque-Bera (JB):	4.502			
Skew:	-0.091	Prob(JB):	0.105			
Kurtosis:	3.935	Cond. No.	42.0			

5.1.4 C Fama-French

Dep. Variable:	excess_rtn	R-squared:	0.724			
Model:	OLS	Adj. R-squared:	0.717			
Method:	Least Squares	F-statistic:	100.5			
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	5.32e-32			
Time:	15:19:03	Log-Likelihood:	194.52			
No. Observations:	119	AIC:	-381.0			
Df Residuals:	115	BIC:	-369.9			
Df Model:	3					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.0077	0.005	-1.692	0.093	-0.017	0.001
mkt	1.4752	0.105	14.010	0.000	1.267	1.684
smb	0.1708	0.181	0.944	0.347	-0.188	0.529
hml	1.0114	0.125	8.096	0.000	0.764	1.259

Omnibus:	0.264	Durbin-Watson:	1.837
Prob(Omnibus):	0.876	Jarque-Bera (JB):	0.121
Skew:	-0.077	Prob(JB):	0.942
Kurtosis:	3.029	Cond. No.	42.0

5.1.5 KO Fama-French

Dep. Variable:	excess_rtn	R-squared:	0.434
Model:	OLS	Adj. R-squared:	0.419
Method:	Least Squares	F-statistic:	29.37
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	3.57e-14
Time:	15:20:01	Log-Likelihood:	231.66
No. Observations:	119	AIC:	-455.3
Df Residuals:	115	BIC:	-444.2
Df Model:	3		

	coef	std err	t	P> t 	[0.025	0.975]
Intercept	0.0002	0.003	0.057	0.955	-0.006	0.007
mkt	0.6516	0.077	8.455	0.000	0.499	0.804
smb	-0.7616	0.132	-5.749	0.000	-1.024	-0.499
hml	0.1742	0.091	1.905	0.059	-0.007	0.355

Omnibus:	0.634	Durbin-Watson:	2.422
Prob(Omnibus):	0.728	Jarque-Bera (JB):	0.289
Skew:	-0.078	Prob(JB):	0.866
Kurtosis:	3.185	Cond. No.	42.0

5.1.6 PEP Fama-French

Dep. Variable:	excess_rtn	R-squared:	0.441
Model:	OLS	Adj. R-squared:	0.427
Method:	Least Squares	F-statistic:	30.27
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	1.69e-14
Time:	15:20:51	Log-Likelihood:	243.18
No. Observations:	119	AIC:	-478.4
Df Residuals:	115	BIC:	-467.2
Df Model:	3		

	coef	std err	t	P> t 	[0.025	0.975]
Intercept	0.0041	0.003	1.365	0.175	-0.002	0.010
mkt	0.6386	0.070	9.129	0.000	0.500	0.777
smb	-0.6303	0.120	-5.240	0.000	-0.869	-0.392
hml	-0.0088	0.083	-0.106	0.916	-0.173	0.156

Omnibus:	0.613	Durbin-Watson:	2.344
Prob(Omnibus):	0.736	Jarque-Bera (JB):	0.549
Skew:	0.164	Prob(JB):	0.760
Kurtosis:	2.944	Cond. No.	42.0

Possiamo notare che in tutti e 6 i titoli il fattore di rischio maggiore é **mkt** che corrisponde al mercato

Chapter 6

Creazione di un portafoglio

Chapter 7

Conclusioni

Contents