Beta Weighting your Portfolio

In this notebook we begin by using yfinance to import financial stock data.
https://ranaroussi.github.io/yfinance/index.html

In [1]:
```
import numpy as np
import pandas as pd
import yfinance as yf
import datetime as dt
from scipy import stats
```

Step 1: Specify date range for analysis

Here we begin by creating start and end dates using pythons datetime module.

In [2]:
```
start = dt.datetime.now() - dt.timedelta(days=365)
end = dt.datetime.now()
start, end
```

Out[2]:
```
(datetime.datetime(2023, 11, 27, 16, 33, 40, 406711),
 datetime.datetime(2024, 11, 26, 16, 33, 40, 406781))
```

Step 2: Select the stocks/tickers you would like to analyse

For stock tickers, use the search bar in yahoo finance to work out other ticker structures. https://finance.yahoo.com/

In [3]:
```
stockList = ['NVDA', 'GOOGL', 'META', 'TSLA','JPM', 'XOM', 'MA', 'CRM', '
stocks = ['SPY'] + stockList
stocks
```

Out[3]:
```
['SPY',
 'NVDA',
 'GOOGL',
 'META',
 'TSLA',
 'JPM',
 'XOM',
 'MA',
 'CRM',
 'AMD',
 'NOW']
```

Step 3 call the yfinance module:

First, we'll retrieve the data for all the tickers using yfinance, and then calculate the log returns.

In [4]:
```
df = yf.download(stocks, start, end).Close
log_returns = np.log(df / df.shift(1)).dropna()
log_returns.head()
```

```
[*********************100%***********************]  11 of 11 completed
```

Out[4]:

Ticker Date	AMD	CRM	GOOGL	JPM	MA	META	NOW	I
2023-11-28	-0.005232	0.000578	0.005775	0.002282	0.000098	0.012736	-0.007950	-0.0
2023-11-29	0.014968	0.023855	-0.016239	0.005067	0.001978	-0.020233	0.020370	0.0
2023-11-30	-0.021959	0.089432	-0.018392	0.011340	0.009737	-0.015318	0.009981	-0.0
2023-12-01	0.001896	0.031649	-0.005068	0.004857	0.001280	-0.007148	0.007337	-0.0
2023-12-04	-0.023505	-0.036584	-0.019837	0.007306	-0.014904	-0.014888	-0.004599	-0.0

Step 4a: Directly calculate beta:

$$\beta = \frac{\mathrm{Cov(Stock,Market)}}{\mathrm{Var(Market)}}$$

In [5]:
```python
def calc_beta(df):
    """
    Calculate beta of each stock against the market index using the covar

    Parameters:
    - df (pd.DataFrame): A DataFrame containing returns data, where one c
      (representing the market index), and other columns represent indivi

    Returns:
    - pd.Series: A Series containing the beta values for each stock, with
    """
    # Extract market returns (from 'SPY' column)
    market = df['SPY'].values
    market_variance = np.var(market)  # Variance of the market index

    betas = []

    # Loop through all columns except 'SPY'
    for col in df.columns:
        if col != 'SPY':
            # Extract stock returns
            stock = df[col].values
            # Calculate covariance between stock and market
            covariance = np.cov(stock, market)[0, 1]
            # Calculate beta as covariance/variance
            beta = covariance / market_variance
            betas.append(beta)

    # Return beta values as a pandas Series
    return pd.Series(betas, index=df.columns.drop('SPY'), name='Beta')
```

In [6]:
```python
calc_beta(log_returns)
```

Out[6]:
```
Ticker
AMD      2.298860
CRM      1.418787
GOOGL    1.190805
JPM      0.757434
MA       0.650546
META     1.550913
NOW      1.364008
NVDA     2.748033
TSLA     2.283945
XOM      0.233868
Name: Beta, dtype: float64
```

Step 4b: Use linear regression to get coefficient of market and stocks returns

In [7]:
```python
def regression_beta(df):
    """
    Calculate beta of each stock against the market using linear regressi

    Parameters:
    - df (pd.DataFrame): DataFrame containing the market index (column na
      and individual stock returns.

    Returns:
    - pd.Series: A Series containing the beta values for each stock.
    """

    market = df['SPY'].values  # Market returns using 'SPY'
    betas = []

    # Loop through all columns except 'SPY'
    for stock in df.columns:
        if stock != 'SPY':
            stock_returns = df[stock].values
            slope, _, _, _, _ = stats.linregress(market, stock_returns)
            betas.append(slope)

    return pd.Series(betas, index=df.columns.drop('SPY'), name='Beta')
```

In [8]:
```python
regression_beta(log_returns)
```

Out[8]:
```
Ticker
AMD      2.289701
CRM      1.413134
GOOGL    1.186061
JPM      0.754416
MA       0.647954
META     1.544734
NOW      1.358573
NVDA     2.737085
TSLA     2.274845
XOM      0.232936
Name: Beta, dtype: float64
```

Step 4c: Use Matrix Algebra to complete linear regression in one line

For linear regression on a model of the form y=Xβ, where X is a matrix with full column rank, the least squares solution,

$$\hat{\beta} = arg\min ||X\beta - y||_2$$

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

https://stats.stackexchange.com/questions/23128/solving-for-regression-parameters-in-closed-form-vs-gradient-descent/23132#23132

In [9]:
```
def matrix_beta(df):
    """
    Calculate beta of each stock against the market using matrix algebra.

    Parameters:
    - df (pd.DataFrame): DataFrame containing the market index (column na
      and individual stock returns.

    Returns:
    - pd.Series: A Series containing the beta values for each stock.
    """

    # Extract the market returns and stock returns
    market = df['SPY'].values[:, np.newaxis]  # Ensure 2D shape for matri
    stocks = df.drop(columns='SPY').values

    # Add an intercept column of ones to the market data
    intercept = np.ones_like(market)
    X = np.hstack((intercept, market))  # Combine intercept and market da

    # Calculate beta using the least squares formula
    betas = np.linalg.pinv(X.T @ X) @ X.T @ stocks

    # Extract the slope coefficients (ignoring the intercept)
    return pd.Series(betas[1], index=df.columns.drop('SPY'), name='Beta')
```

In [10]:
```
beta = matrix_beta(log_returns)
beta
```

Out[10]:
```
Ticker
AMD      2.289701
CRM      1.413134
GOOGL    1.186061
JPM      0.754416
MA       0.647954
META     1.544734
NOW      1.358573
NVDA     2.737085
TSLA     2.274845
XOM      0.232936
Name: Beta, dtype: float64
```

Step 5: Define your Portfolio and make DataFrame

Calculate Beta Weighted Portfolio

In [11]:
```python
# Number of shares held for each stock (excluding SPY)
units = np.array([100, 250, 300, 400, 200, 150, 180, 220, 170, 140])

# Extract the latest prices for all stocks (excluding SPY)
SPYprices = df['SPY'].iloc[-1]
stock_prices = df.iloc[-1].drop('SPY').values
price = np.round(stock_prices, 2)  # Round prices to 2 decimal places

# Calculate portfolio value for each stock
value = units * price

# Calculate portfolio weights
total_value = np.sum(value)
weights = np.round(value / total_value, 2)  # Rounded to 2 decimal places

# Round beta to 2 decimal places (assuming `beta` is already calculated)
beta = np.round(beta, 2)
```

In [12]:
```python
Portfolio = pd.DataFrame({
    'Direction': 'Long',
    'Type': 'S',
    'Stock Price': price,
    'Price': price,
    'Units': units,
    'Value': units*price,
    'Weight': weights,
    'Beta': beta,
    'Weighted Beta': weights*beta
}).reset_index()
Portfolio
```

Out[12]:

	Ticker	Direction	Type	Stock Price	Price	Units	Value	Weight	Beta	Weighte Bet
0	AMD	Long	S	141.13	141.13	100	14113.0	0.02	2.29	0.045
1	CRM	Long	S	339.11	339.11	250	84777.5	0.12	1.41	0.169
2	GOOGL	Long	S	167.65	167.65	300	50295.0	0.07	1.19	0.083
3	JPM	Long	S	250.29	250.29	400	100116.0	0.14	0.75	0.105
4	MA	Long	S	526.60	526.60	200	105320.0	0.14	0.65	0.091
5	META	Long	S	565.11	565.11	150	84766.5	0.12	1.54	0.184
6	NOW	Long	S	1052.71	1052.71	180	189487.8	0.26	1.36	0.353
7	NVDA	Long	S	136.02	136.02	220	29924.4	0.04	2.74	0.109
8	TSLA	Long	S	338.59	338.59	170	57560.3	0.08	2.27	0.181
9	XOM	Long	S	119.97	119.97	140	16795.8	0.02	0.23	0.004

Step 6: What if we have options, let's consider things in terms of Delta

In [13]:
```python
Portfolio = Portfolio.drop(['Weight', 'Weighted Beta'], axis=1)
Portfolio['Delta'] = Portfolio['Units']
Portfolio
```

Out[13]:

	Ticker	Direction	Type	Stock Price	Price	Units	Value	Beta	Delta
0	AMD	Long	S	141.13	141.13	100	14113.0	2.29	100
1	CRM	Long	S	339.11	339.11	250	84777.5	1.41	250
2	GOOGL	Long	S	167.65	167.65	300	50295.0	1.19	300
3	JPM	Long	S	250.29	250.29	400	100116.0	0.75	400
4	MA	Long	S	526.60	526.60	200	105320.0	0.65	200
5	META	Long	S	565.11	565.11	150	84766.5	1.54	150
6	NOW	Long	S	1052.71	1052.71	180	189487.8	1.36	180
7	NVDA	Long	S	136.02	136.02	220	29924.4	2.74	220
8	TSLA	Long	S	338.59	338.59	170	57560.3	2.27	170
9	XOM	Long	S	119.97	119.97	140	16795.8	0.23	140

Step 7: Weight the Delta's using Beta

In [14]:

```
Portfolio['SPY Weighted Delta (point)'] = round(Portfolio['Beta'] * (Port
Portfolio['SPY Weighted Delta (1%)'] = round(Portfolio['Beta'] * (Portfol
Portfolio
```

Out[14]:

	Ticker	Direction	Type	Stock Price	Price	Units	Value	Beta	Delta	SPY Weighted Delta (point)
0	AMD	Long	S	141.13	141.13	100	14113.0	2.29	100	54.09
1	CRM	Long	S	339.11	339.11	250	84777.5	1.41	250	200.05
2	GOOGL	Long	S	167.65	167.65	300	50295.0	1.19	300	100.16
3	JPM	Long	S	250.29	250.29	400	100116.0	0.75	400	125.66
4	MA	Long	S	526.60	526.60	200	105320.0	0.65	200	114.57
5	META	Long	S	565.11	565.11	150	84766.5	1.54	150	218.47
6	NOW	Long	S	1052.71	1052.71	180	189487.8	1.36	180	431.28
7	NVDA	Long	S	136.02	136.02	220	29924.4	2.74	220	137.22
8	TSLA	Long	S	338.59	338.59	170	57560.3	2.27	170	218.67
9	XOM	Long	S	119.97	119.97	140	16795.8	0.23	140	6.47

Step 8: Total the Delta's to get Portfolio Overview

In [15]:

```
Portfolio.loc['Total', ['Value', 'SPY Weighted Delta (point)', 'SPY Weigh
= Portfolio[['Value','SPY Weighted Delta (point)', 'SPY Weighted Delta (1
Portfolio
```

Out[15]:

	Ticker	Direction	Type	Stock Price	Price	Units	Value	Beta	Delta	Weigh D (pc
0	AMD	Long	S	141.13	141.13	100.0	14113.0	2.29	100.0	5
1	CRM	Long	S	339.11	339.11	250.0	84777.5	1.41	250.0	20
2	GOOGL	Long	S	167.65	167.65	300.0	50295.0	1.19	300.0	10
3	JPM	Long	S	250.29	250.29	400.0	100116.0	0.75	400.0	12
4	MA	Long	S	526.60	526.60	200.0	105320.0	0.65	200.0	11
5	META	Long	S	565.11	565.11	150.0	84766.5	1.54	150.0	21
6	NOW	Long	S	1052.71	1052.71	180.0	189487.8	1.36	180.0	43
7	NVDA	Long	S	136.02	136.02	220.0	29924.4	2.74	220.0	13
8	TSLA	Long	S	338.59	338.59	170.0	57560.3	2.27	170.0	21
9	XOM	Long	S	119.97	119.97	140.0	16795.8	0.23	140.0	
Total	NaN	NaN	NaN	NaN	NaN	NaN	733156.3	NaN	NaN	160