capm

November 19, 2023

1 CAPM CAPITAL ASSET PRICING MODEL

Import necessary libraries

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

Here we will be using the data's of Apple and Amazon, so two tech companies.Then we have General Electric, who was a company that's had poor performance over this time period.Plus, we have the VIX, which is a measure of volatility in the market.It actually uses a combination of put options and call options to try to mimic the sentiment for volatility in the future market.

2 Load stock data

At first we load the csv files containing the stock prices of apple, amazon, general electric and the CBOE Volatility Index since 1-1-2010

[2]:
```
aapl = pd.read_csv("apple_2010.csv",index_col='Date',parse_dates=True)
amzn = pd.read_csv("amazon_2010.csv",index_col='Date',parse_dates=True)
ge = pd.read_csv("GE_2010.csv",index_col='Date',parse_dates=True)
vix = pd.read_csv("VIX_2010.csv",index_col='Date',parse_dates=True)
sp500 = pd.read_csv("sp500_2010.csv",index_col='Date',parse_dates=True)
```

Lets take a look at one of them.You'll notice we have open, high, low, close, adjusted close and volume.What we're going to do is just check out the cumulative returns over time so we can get a visual representation of how well we're performing against the overall market.And technically, we can do this two ways we can do absolute cumulative returns,so the actual dollar gain.But it's probably better to do it in percent gain.

[3]:
```
aapl
```

[3]:

Date	Open	High	Low	Close	Adj Close
2009-12-31	7.611786	7.619643	7.520000	7.526071	6.462008
2010-01-04	7.622500	7.660714	7.585000	7.643214	6.562591
2010-01-05	7.664286	7.699643	7.616071	7.656429	6.573935
2010-01-06	7.656429	7.686786	7.526786	7.534643	6.469369
2010-01-07	7.562500	7.571429	7.466071	7.520714	6.457407

```
…                …          …          …          …          …
2021-08-30  149.000000  153.490005  148.610001  153.119995  153.119995
2021-08-31  152.660004  152.800003  151.289993  151.830002  151.830002
2021-09-01  152.830002  154.979996  152.339996  152.509995  152.509995
2021-09-02  153.869995  154.720001  152.399994  153.649994  153.649994
2021-09-03  153.759995  154.630005  153.089996  154.300003  154.300003

                Volume
Date
2009-12-31  352410800
2010-01-04  493729600
2010-01-05  601904800
2010-01-06  552160000
2010-01-07  477131200
…                 …
2021-08-30   90956700
2021-08-31   86453100
2021-09-01   80313700
2021-09-02   71115500
2021-09-03   57808700

[2940 rows x 6 columns]
```

3 Compute cumulative returns

The cumulative return is the overall total return of an investment over a given time period. You can compute it by subtracting the current price of a stock from the price it had when you bought it.To compute the percentage value, you divide it instead of subtracting

[6]:
```
aapl["Adj Close"]
```

[6]:
```
Date
2009-12-31      6.462008
2010-01-04      6.562591
2010-01-05      6.573935
2010-01-06      6.469369
2010-01-07      6.457407
                  …
2021-08-30    153.119995
2021-08-31    151.830002
2021-09-01    152.509995
2021-09-02    153.649994
2021-09-03    154.300003
Name: Adj Close, Length: 2940, dtype: float64
```

[29]:
```
amzn["Adj Close"]
```

[29]:
```
Date
2009-12-31     134.520004
2010-01-04     133.899994
2010-01-05     134.690002
2010-01-06     132.250000
2010-01-07     130.000000
                  …
2021-08-30    3421.570068
2021-08-31    3470.790039
2021-09-01    3479.000000
2021-09-02    3463.120117
2021-09-03    3478.050049
Name: Adj Close, Length: 2940, dtype: float64
```

[7]:
```
ge["Adj Close"]
```

[7]:
```
Date
2009-12-31     86.062485
2010-01-04     87.882713
2010-01-05     88.337753
2010-01-06     87.882713
2010-01-07     92.433258
                  …
2021-08-30    105.190002
2021-08-31    105.410004
2021-09-01    103.660004
2021-09-02    106.260002
2021-09-03    104.750000
Name: Adj Close, Length: 2940, dtype: float64
```

So let's go ahead and create a function that can take in one of these data frames and then compute the cumulative return.

[8]:
```
def compute_cumulative(data, get_absolute=True):
    initial_price = data["Adj Close"].iloc[0]
    last_price = data["Adj Close"].iloc[-1]
    if get_absolute:
        # Returns absolute dollar gain for 1 share
        return last_price - initial_price
    else:
        # Returns percentage change
        return 100*(last_price-initial_price) / initial_price
```

[9]:
```
compute_cumulative(aapl)
```

[9]:
```
147.8379945755005
```

To get the absolute value

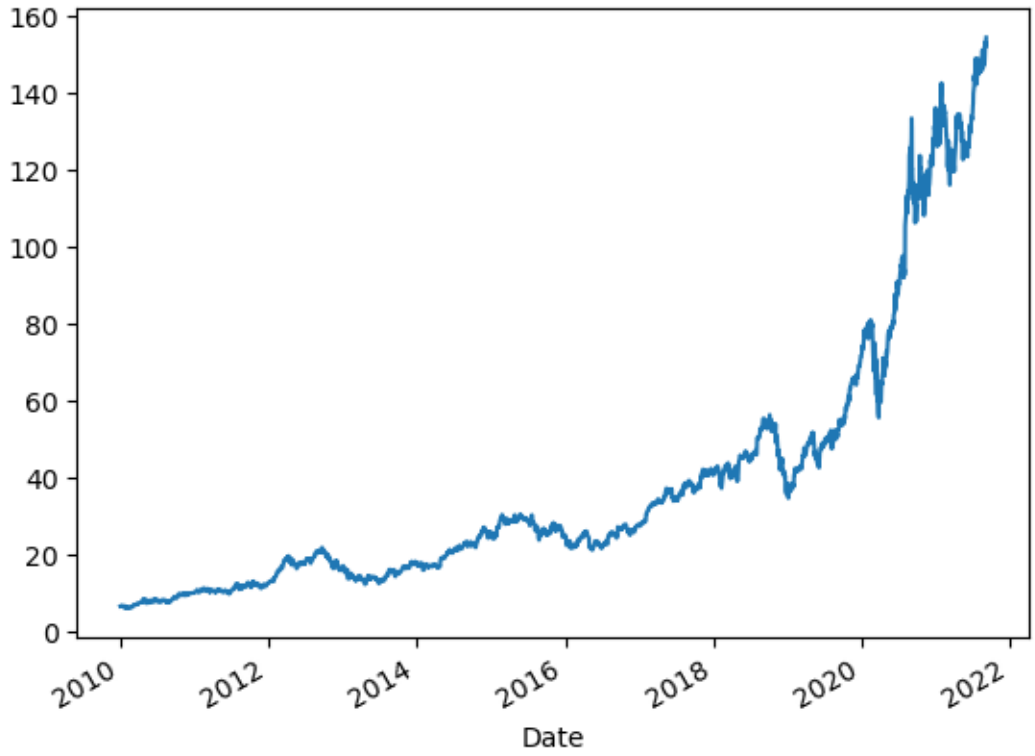
[10]:
```
compute_cumulative(aapl,get_absolute=False)
```

[10]:
```
2287.802548057404
```

So let's just confirm that Apple really did perform that well

[12]:
```
aapl["Adj Close"].plot()
```

[12]:
```
<Axes: xlabel='Date'>
```



So, it looks like it started at a very low dollar amount, see below.

[13]:
```
aapl
```

[13]:

Date	Open	High	Low	Close	Adj Close
2009-12-31	7.611786	7.619643	7.520000	7.526071	6.462008
2010-01-04	7.622500	7.660714	7.585000	7.643214	6.562591
2010-01-05	7.664286	7.699643	7.616071	7.656429	6.573935
2010-01-06	7.656429	7.686786	7.526786	7.534643	6.469369
2010-01-07	7.562500	7.571429	7.466071	7.520714	6.457407
…	…	…	…	…	…
2021-08-30	149.000000	153.490005	148.610001	153.119995	153.119995

```
2021-08-31  152.660004  152.800003  151.289993  151.830002  151.830002
2021-09-01  152.830002  154.979996  152.339996  152.509995  152.509995
2021-09-02  153.869995  154.720001  152.399994  153.649994  153.649994
2021-09-03  153.759995  154.630005  153.089996  154.300003  154.300003

               Volume
Date
2009-12-31  352410800
2010-01-04  493729600
2010-01-05  601904800
2010-01-06  552160000
2010-01-07  477131200
…                 …
2021-08-30   90956700
2021-08-31   86453100
2021-09-01   80313700
2021-09-02   71115500
2021-09-03   57808700

[2940 rows x 6 columns]
```

Lets see how it ended up

[14]:
```
aapl.tail()
```

[14]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2021-08-30  149.000000  153.490005  148.610001  153.119995  153.119995
2021-08-31  152.660004  152.800003  151.289993  151.830002  151.830002
2021-09-01  152.830002  154.979996  152.339996  152.509995  152.509995
2021-09-02  153.869995  154.720001  152.399994  153.649994  153.649994
2021-09-03  153.759995  154.630005  153.089996  154.300003  154.300003

              Volume
Date
2021-08-30  90956700
2021-08-31  86453100
2021-09-01  80313700
2021-09-02  71115500
2021-09-03  57808700
```

Started with 6,46USD ended with 154,30USD Pretty cool.So clearly, there's a huge jump here, and 2287% increase calculated above is actually in line with what we see here in the adjusted close price.

Let's go ahead and now calculate the cumulative return in percent terms for all the actual stocks.

[15]:
```
def create_cumulative_abs(df):
    df['Cumulative Absolute'] = df["Adj Close"] - df['Adj Close'].iloc[0]

    return df
```

[16]:
```
aapl = create_cumulative_abs(aapl)
```

[17]:
```
aapl
```

[17]:

Date	Open	High	Low	Close	Adj Close	Volume	Cumulative Absolute
2009-12-31	7.611786	7.619643	7.520000	7.526071	6.462008	352410800	0.000000
2010-01-04	7.622500	7.660714	7.585000	7.643214	6.562591	493729600	0.100582
2010-01-05	7.664286	7.699643	7.616071	7.656429	6.573935	601904800	0.111927
2010-01-06	7.656429	7.686786	7.526786	7.534643	6.469369	552160000	0.007360
2010-01-07	7.562500	7.571429	7.466071	7.520714	6.457407	477131200	-0.004601
…	…	…	…	…	…	…	…
2021-08-30	149.000000	153.490005	148.610001	153.119995	153.119995	90956700	146.657987
2021-08-31	152.660004	152.800003	151.289993	151.830002	151.830002	86453100	145.367993
2021-09-01	152.830002	154.979996	152.339996	152.509995	152.509995	80313700	146.047986
2021-09-02	153.869995	154.720001	152.399994	153.649994	153.649994	71115500	147.187985
2021-09-03	153.759995	154.630005	153.089996	154.300003	154.300003	57808700	147.837995

[2940 rows x 7 columns]

Second day you already gained 10 cents, followind day 11 cents, etc. Clearly, they are going to be some days where your gain maybe negative.So the column on the right show you your cumulative gain in absolute terms.Remember, this is all in dollar terms.

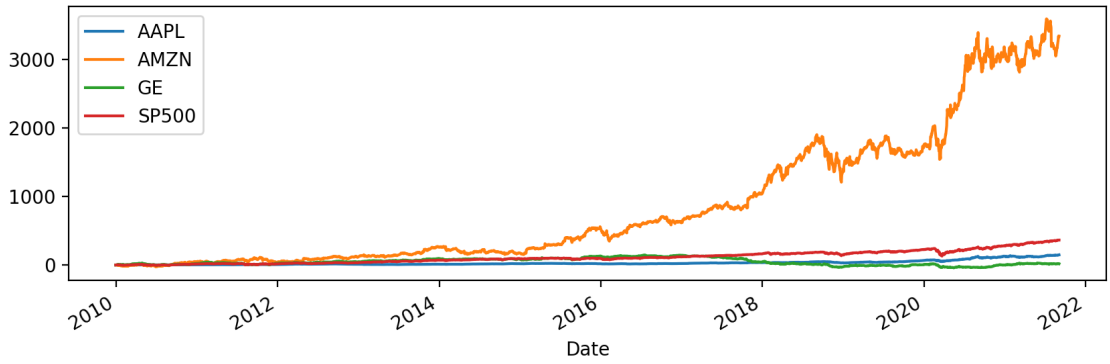
So let's go ahead and do this for all the data frame.

[19]:
```
amzn = create_cumulative_abs(amzn)
ge = create_cumulative_abs(ge)
sp500 = create_cumulative_abs(sp500)
```

Now I have Amazon, General Electric and SPF 500 absolute dollar gains.So let's just for fun,lets compare these.

[20]:
```
plt.figure(figsize=(10,3),dpi=200)
aapl['Cumulative Absolute'].plot(label='AAPL')
amzn['Cumulative Absolute'].plot(label='AMZN')
ge['Cumulative Absolute'].plot(label='GE')
sp500['Cumulative Absolute'].plot(label='SP500')
plt.legend()
```

[20]:
```
<matplotlib.legend.Legend at 0x1529d36ecd0>
```

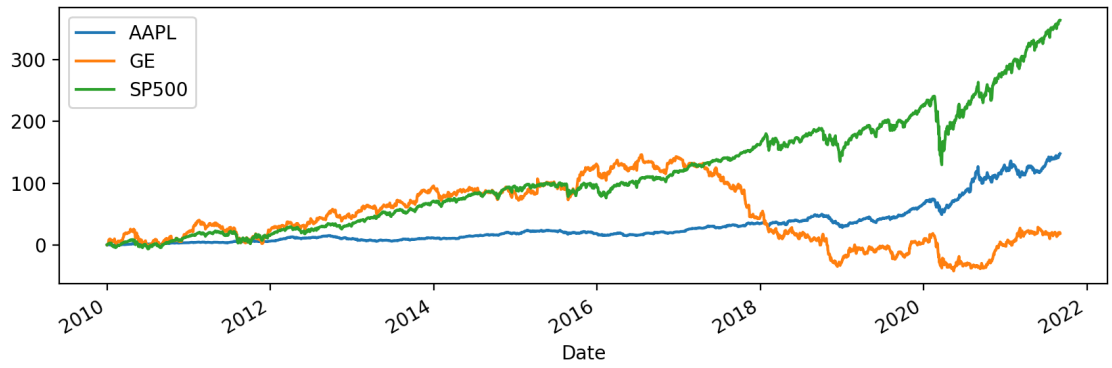


4 COMMENT

Looks great but we see the absolute dollar gains, and keep in mind, it's not fair to do this comparison from one stock to another because we should really be considering the initial capital spent. However, it is interesting to see the pure dollar gains for some stocks being so much greater than other ones. You'll notice here that Amazon gained a lot impure dollar terms versus Apple, GE and S&P 500.So even with their amazing percent gains, if we just look at pure dollar terms, Amazon makes these others almost look flat.However, we can always just take out Amazon, run it and then we get to see the performance again in pure dollar terms to avoid the unfair competetion.:)Now, as we've mentioned, you probably shouldn't be using pure dollar terms for this sort of comparison.We are able to see here how many dollars earned per share if we held since the start of the time series, but it's probably way better to use a cumulative percent gain.Let's go ahead and create a function that does that for us.But first, lets take out Amazon and see how our graph looks like.

[21]:
```
plt.figure(figsize=(10,3),dpi=200)
aapl['Cumulative Absolute'].plot(label='AAPL')
ge['Cumulative Absolute'].plot(label='GE')
sp500['Cumulative Absolute'].plot(label='SP500')
plt.legend()
```

[21]:
```
<matplotlib.legend.Legend at 0x1529de47990>
```



Now we can create the function mentioned above.It's going to be much better to actually compare and plot everything based off the percent change.

[23]:
```
def calc_cum_perc(df):
    df['Percent Change'] = 100*(df['Adj Close'] - df['Adj Close'].iloc[0]) /␣
 ↪df['Adj Close'].iloc[0] #initial adjusted close price

    return df
```

[24]:
```
aapl = calc_cum_perc(aapl)
```

[26]:
```
aapl
```

[26]:

Date	Open	High	Low	Close	Adj Close	Volume	Cumulative Absolute	Percent Change
2009-12-31	7.611786	7.619643	7.520000	7.526071	6.462008	352410800	0.000000	0.000000
2010-01-04	7.622500	7.660714	7.585000	7.643214	6.562591	493729600	0.100582	1.556515
2010-01-05	7.664286	7.699643	7.616071	7.656429	6.573935	601904800	0.111927	1.732071
2010-01-06	7.656429	7.686786	7.526786	7.534643	6.469369	552160000	0.007360	0.113904
2010-01-07	7.562500	7.571429	7.466071	7.520714	6.457407	477131200	-0.004601	-0.071201
…	…	…	…	…	…			
2021-08-30	149.000000	153.490005	148.610001	153.119995	153.119995			
2021-08-31	152.660004	152.800003	151.289993	151.830002	151.830002			
2021-09-01	152.830002	154.979996	152.339996	152.509995	152.509995			
2021-09-02	153.869995	154.720001	152.399994	153.649994	153.649994			
2021-09-03	153.759995	154.630005	153.089996	154.300003	154.300003			

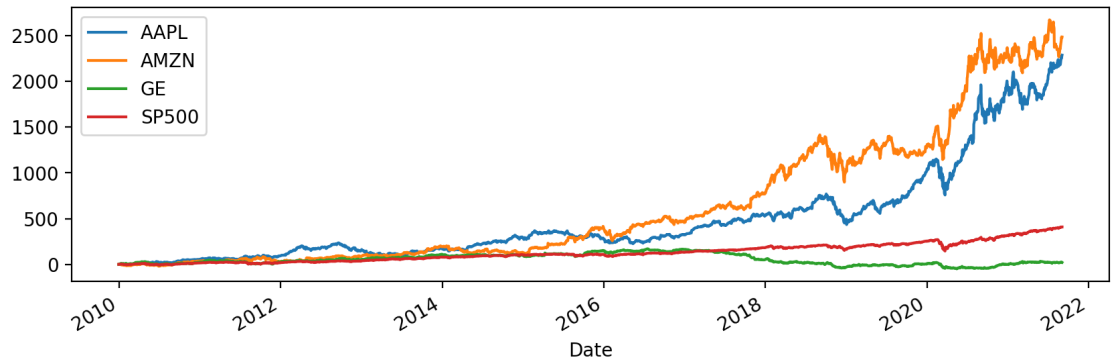
```
...                ...                 ...              ...
2021-08-30    90956700            146.657987      2269.541849
2021-08-31    86453100            145.367993      2249.579119
2021-09-01    80313700            146.047986      2260.102050
2021-09-02    71115500            147.187985      2277.743614
2021-09-03    57808700            147.837995      2287.802548

[2940 rows x 8 columns]
```

Lets do the same for other stocks and plot them.

[27]:
```
amzn = calc_cum_perc(amzn)
ge = calc_cum_perc(ge)
sp500 = calc_cum_perc(sp500)
```

[28]:
```
plt.figure(figsize=(10,3),dpi=200)
aapl['Percent Change'].plot(label='AAPL')
amzn['Percent Change'].plot(label='AMZN')
ge['Percent Change'].plot(label='GE')
sp500['Percent Change'].plot(label='SP500')
plt.legend()
```

[28]:
```
<matplotlib.legend.Legend at 0x1529e05ce10>
```



5 COMMENT

This is a much fairer comparison because everything's normalized in terms of percent.this is taking into account that initial capital spent because the Apple share that it costs the same as an Amazon share at the start of 2010, right?So what does this actually tell us?Well, it tells us that you probably should have invested in Apple and Amazon, and depending on how much capital you had available to locate, you would have actually performed pretty similar results,whether you had invested in Apple or Amazon.Check out the above graph,(output20).It looked like Amazon as way better performing than Apple.But, recall, in reality, they had very different starting prices.If I take a

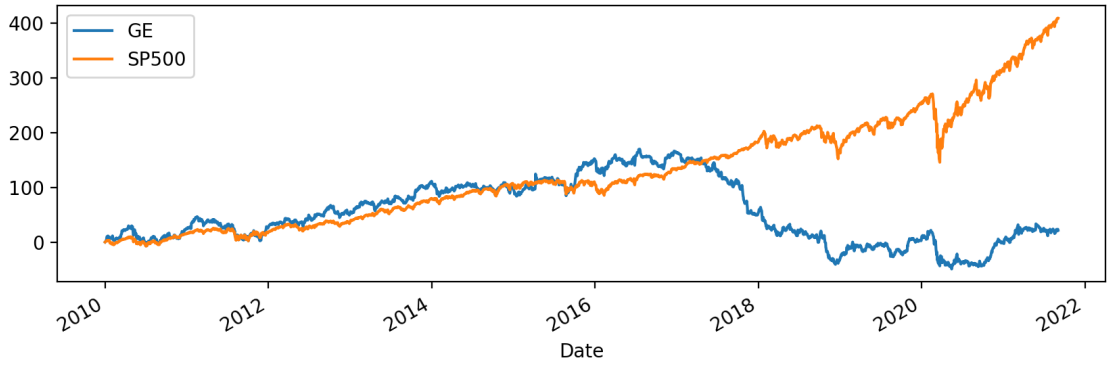
look at Amazon, know how it starts at a price of 134USD and ends at 3470USD versus Apple started a lot cheaper.So obviously, when Apple is such a lower starting price, it's not going to be as reflective of absolute dollar gains.But when we think about it in terms of percent now we can fairly compare.In reality, you would have performed very well in either of them, actually almost the same at the very end of the time period.But notice that your market, the S&P 500, while it performed quite well, it's not nearly this percent gain of these technology stocks of Apple and Amazon.And notice that General Electric actually performed very poorly.In fact, let's remove Apple and Amazon from this plot just to fully understand what's happening.

[30]:
```
amzn
```

[30]:
```
                  Open         High          Low        Close    Adj Close  \
Date
2009-12-31   137.089996   137.279999   134.520004   134.520004   134.520004
2010-01-04   136.250000   136.610001   133.139999   133.899994   133.899994
2010-01-05   133.429993   135.479996   131.809998   134.690002   134.690002
2010-01-06   134.600006   134.729996   131.649994   132.250000   132.250000
2010-01-07   132.009995   132.320007   128.800003   130.000000   130.000000
…                   …            …            …            …            …
2021-08-30  3357.429932  3445.000000  3355.219971  3421.570068  3421.570068
2021-08-31  3424.800049  3472.580078  3395.590088  3470.790039  3470.790039
2021-09-01  3496.399902  3527.000000  3475.239990  3479.000000  3479.000000
2021-09-02  3494.760010  3511.959961  3455.000000  3463.120117  3463.120117
2021-09-03  3452.000000  3482.669922  3436.439941  3478.050049  3478.050049

               Volume  Cumulative Absolute  Percent Change
Date
2009-12-31   4523000             0.000000        0.000000
2010-01-04   7599900            -0.620010       -0.460906
2010-01-05   8851900             0.169998        0.126374
2010-01-06   7178800            -2.270004       -1.687485
2010-01-07  11030200            -4.520004       -3.360098
…                …                    …               …
2021-08-30   3192200          3287.050064     2443.539964
2021-08-31   4356400          3336.270035     2480.129296
2021-09-01   3629900          3344.479996     2486.232448
2021-09-02   2923700          3328.600113     2474.427600
2021-09-03   2575700          3343.530045     2485.526270

[2940 rows x 8 columns]
```

[31]:
```
plt.figure(figsize=(10,3),dpi=200)
ge['Percent Change'].plot(label='GE')
sp500['Percent Change'].plot(label='SP500')
plt.legend()
```

[31]:
```
<matplotlib.legend.Legend at 0x1529e27c310>
```



6 COMMENT

I can see that S&P 500, looks like performed much better than General Electric.

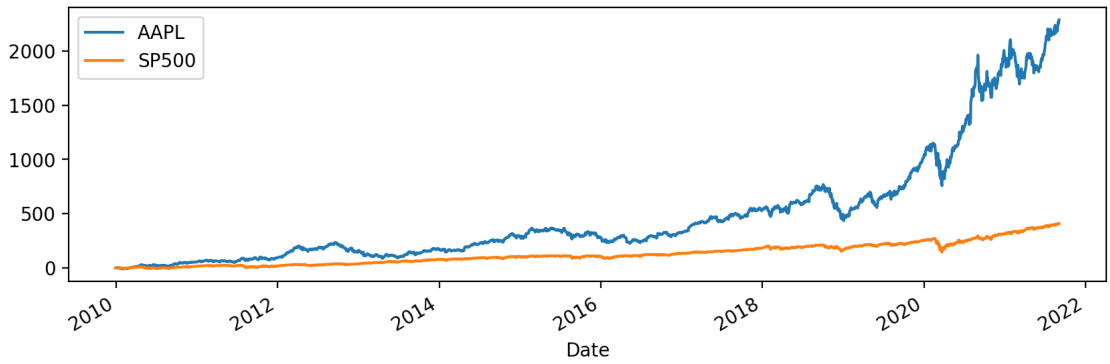
So what does this actually have to do with the capital asset pricing model?The capital asset pricing model takes into account the relationship between the returns and the actual market So if we treat the S&P 500 as our overall market, we're going to be interested in how the daily returns are related to thr market.

So, for example, let's go ahead and take a closer look at the S&P 500 again versus GE.You'll notice that there is some sort of relationship here.As GE goes up, it looks like the S&P 500 also went up and when the S&P 500 did crash so that GE, and we actually see this behavior even in Apple.

So let's compare Apple and the S&P 500.

[32]:
```
plt.figure(figsize=(10,3),dpi=200)
aapl['Percent Change'].plot(label='AAPL')
sp500['Percent Change'].plot(label='SP500')
plt.legend()
```

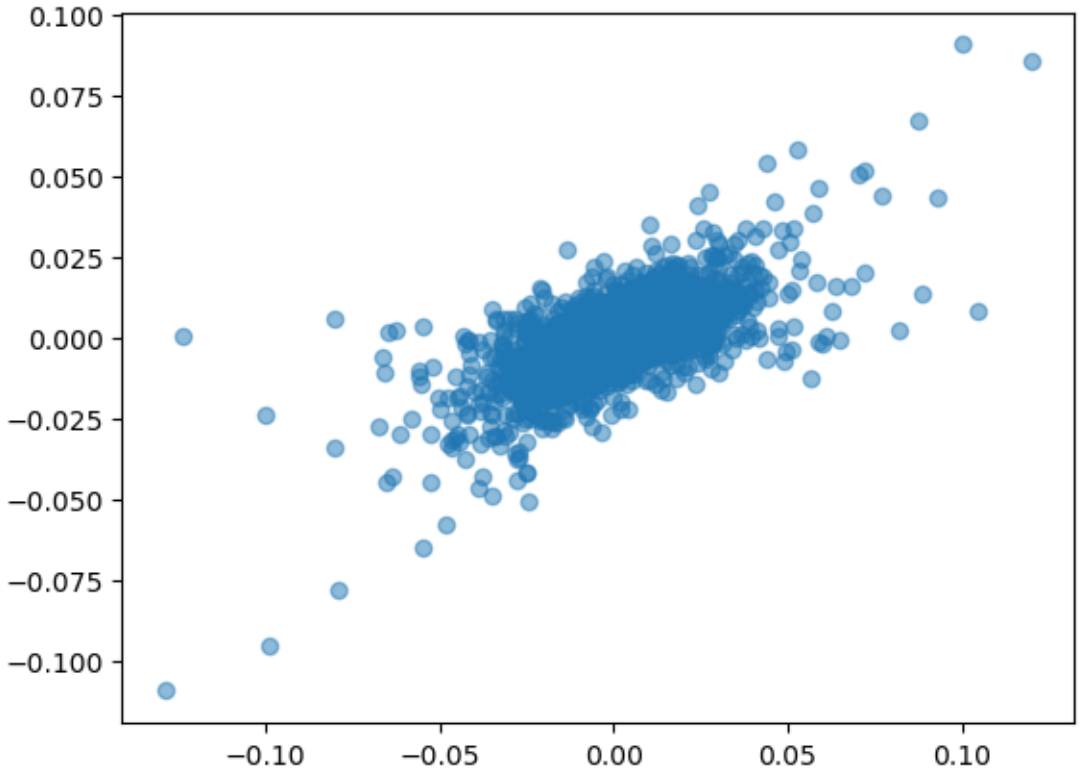
[32]:
```
<matplotlib.legend.Legend at 0x1529e031310>
```



So while it looks like they're really unrelated, you do see some behavior when Apple crashes.Notice there's a dip right after 2020 for both stocks.Lets compare the daily returns ofS&P 500 versus Apple through scatter plots so that I can get an idea if Apple is going up, is the market also going up?

[33]:
```
aapl['Daily Returns'] = aapl['Adj Close'].pct_change()
sp500['Daily Returns'] = sp500['Adj Close'].pct_change()
plt.scatter(aapl['Daily Returns'],sp500['Daily Returns'],alpha=0.5)
```

[33]: <matplotlib.collections.PathCollection at 0x152a0423c10>



7 COMMENT

If there was a perfect one to one correlation between daily returns of Apple and S&P 500, we would have a straight line.Our X and Y axes would have points equal to each other. So the more you are linearly aligned, the greater your beta is going to be.

8 CAPM,BETA,ALPHA AND LINEAR REGRESSION

Lets try to understand notion of CAPM and Beta.The actual use case of capital asset pricing model when it first came out was to try to define some sort of expected return of your investment.But in fact,this is a pretty outdated methodology in order to figure out some sort of expected return

because it relies so heavily on many assumptions. However, the actual concept of a beta and alpha are extremely useful to try to understand the overall performance of any particular security or even a portfolio or algorithm.So, let's go ahead and check out how we can derive beta and alpha terms.

Now recall we were able to calculate some daily returns, as well as check out a visual relationship between those returns Now, let's solidify this further by actually performing a linear regression.And what this simple linear regression is going to do is it's going to return a slope intercept R value and P value. And for our use case, we really just care about the slope and the intercept where the slope is going to be our beta term and the intercept is going to be our alpha term.

So let's make sure that we calculate the daily returns for all the securities that are remaining, and then we also need to make sure that we drop any missing values.Otherwise, they will mess up our linear regression calculation.Recall we've already done this daily returns calculation for Apple and SP500

[34]:
```
amzn['Daily Returns'] = amzn['Adj Close'].pct_change(1)
ge['Daily Returns'] =ge['Adj Close'].pct_change(1)
vix['Daily Returns'] = vix['Adj Close'].pct_change(1)
```

[35]:
```
aapl = aapl.dropna()
ge = ge.dropna()
vix = vix.dropna()
sp500 = sp500.dropna()
amzn = amzn.dropna()
```

It's time to actually calculate the linear regression between the S&P 500 daily returns and any particular stocks daily returns.

[36]:
```
from scipy.stats import linregress
```

[37]:
```
def beta_and_alpha(df):
    beta,alpha,_,_,_ = linregress(sp500['Daily Returns'],df['Daily Returns'])

    return beta,alpha
```

[38]:
```
beta_and_alpha(aapl)
```

[38]:
```
(1.0917032604626986, 0.00056985495810669)
```

9 COMMENT

Notice Apple has a very high beta, so almost moves really closely in line with actual S&P 500the market.The thing to note here is that because it's a little higher than one, it's actually going to move a little higher or a little lower than the gains or drops in the S&P 500.So that is to say, if the S&P 500 one day moved 1% , Apple tends to move by 1.09%, just a little higher.If the S&P 500 drops by 1%, Apple tends to drop by at 1.09%.And we can see there is a little bit of alpha here.But in general, Apple is a high beta stock, essentially telling you that it moves very much in line wih SP500.It makes sense because Apple actually comprises a very large portion of the S&P

500 at certain times, it could be up to five percent of the entire index because Apple is such a huge company.

let's check out for Amazon.

[39]:
```
beta_and_alpha(amzn)
```

[39]:
```
(1.026474799622411, 0.0006735393441121329)
```

10 COMMENT

Notice that Amazon very similar behavior to Apple.It moves in a little bit of a greater one beta ratio.And it also has a little bit of small alpha.

What about GE?

[40]:
```
beta_and_alpha(ge)
```

[40]:
```
(1.1488219188566973, -0.00043588058002061946)
```

11 COMMENT

Well, here we can see that GE actually has a higher beta than both Apple and Amazon.Is this good or bad? it really depends on what the market is over that time period.But this is essentially telling you when the market moves up by 1%, GE tends to move up even higher than by 1.4% percent or when the market moves down GE tends to move down even more than by 1.4%.However, the big concern is a negative alpha.What is a negative alpha mean? We know alpha is going to be the performance that is not correlated with the general market performance of daily returns. Having a negative alpha means that even as the market moves up, GE is able to move down so it's able to actually not have performance or have poor performance, regardless of the actual market conditions.You really don't want to have investments with negative alpha.You would prefer high alpha because that means you gain performance regardless of market conditions.But having a negative alpha tends to mean that even if the market's moving up, there's still going to be performance that is drawing GE down.

Now let's check it out for VIX.

So if we run this for VIX, we get a negative beta and this is really interesting.This means that we essentially have a negative correlation to the actual market.And that's actually what VIX is more or less designed to do.

[41]:
```
beta_and_alpha(vix)
```

[41]:
```
(-5.89940794981144, 0.006742981581293554)
```

12 COMMENT

That means as your S&P 500 moves up as that general market moves up, the VIX tends to move in the exact opposite direction by around -6%.So essentially, if we had an SP500 gain of 1%, VIX

would move down by about around -6%.

13 WHAT DOES LEVERAGED ETFs MEAN?

It simply means lborrow money to try to increase returns.

[42]:
```
sp_lev_ETF = pd.read_csv('sp_leveraged_2010.csv',
                         parse_dates=True,
                         index_col='Date')
```

[43]:
```
sp_lev_ETF
```

[43]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2009-12-31    9.787500    9.787500    9.552500    9.560000    9.009787
2010-01-04    9.715000    9.887500    9.715000    9.867500    9.299584
2010-01-05    9.867500    9.937500    9.780000    9.932500    9.360847
2010-01-06    9.910000    9.990000    9.892500    9.940000    9.367911
2010-01-07    9.907500   10.047500    9.850000   10.022500    9.445668
…                  …           …           …           …           …
2021-08-30  131.880005  133.119995  131.750000  132.619995  132.619995
2021-08-31  132.550003  132.779999  131.869995  132.309998  132.309998
2021-09-01  132.809998  133.139999  132.229996  132.380005  132.380005
2021-09-02  133.240005  133.669998  132.429993  133.169998  133.169998
2021-09-03  132.500000  133.419998  132.210007  133.119995  133.119995

              Volume
Date
2009-12-31  33780400
2010-01-04  43924400
2010-01-05  38780400
2010-01-06  42499600
2010-01-07  45704800
…                …
2021-08-30   1258900
2021-08-31   1763600
2021-09-01   1402900
2021-09-02    929800
2021-09-03   1260500

[2940 rows x 6 columns]
```

[44]:
```
sp_lev_ETF = calc_cum_perc(sp_lev_ETF)
```

[45]:
```
sp_lev_ETF
```

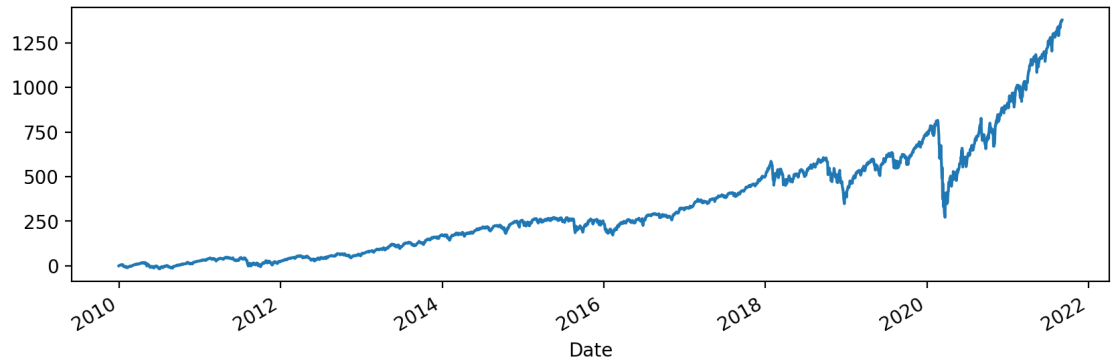
[45]:

Date	Open	High	Low	Close	Adj Close	Volume	Percent Change
2009-12-31	9.787500	9.787500	9.552500	9.560000	9.009787	33780400	0.000000
2010-01-04	9.715000	9.887500	9.715000	9.867500	9.299584	43924400	3.216478
2010-01-05	9.867500	9.937500	9.780000	9.932500	9.360847	38780400	3.896429
2010-01-06	9.910000	9.990000	9.892500	9.940000	9.367911	42499600	3.974841
2010-01-07	9.907500	10.047500	9.850000	10.022500	9.445668	45704800	4.837868
…	…	…	…	…	…	…	…
2021-08-30	131.880005	133.119995	131.750000	132.619995	132.619995	1258900	1371.954897
2021-08-31	132.550003	132.779999	131.869995	132.309998	132.309998	1763600	1368.514221
2021-09-01	132.809998	133.139999	132.229996	132.380005	132.380005	1402900	1369.291235
2021-09-02	133.240005	133.669998	132.429993	133.169998	133.169998	929800	1378.059404
2021-09-03	132.500000	133.419998	132.210007	133.119995	133.119995	1260500	1377.504418

[2940 rows x 7 columns]

[46]:
```
plt.figure(figsize=(10,3),dpi=200)
sp_lev_ETF['Percent Change'].plot(label='Lev Sp500')
```

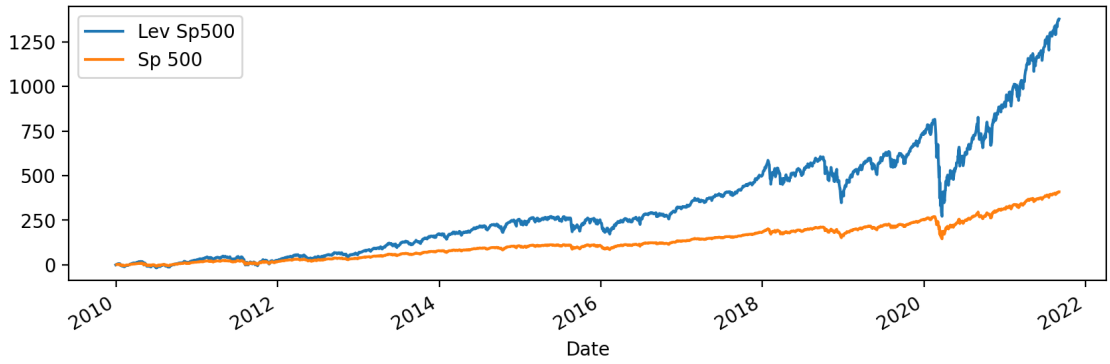
[46]: <Axes: xlabel='Date'>



So just running this figure, we can see we have really large gains. Let's go ahead and plot it against SP500.

[47]:
```
plt.figure(figsize=(10,3),dpi=200)
sp_lev_ETF['Percent Change'].plot(label='Lev Sp500')
sp500['Percent Change'].plot(label='Sp 500')
plt.legend()
```

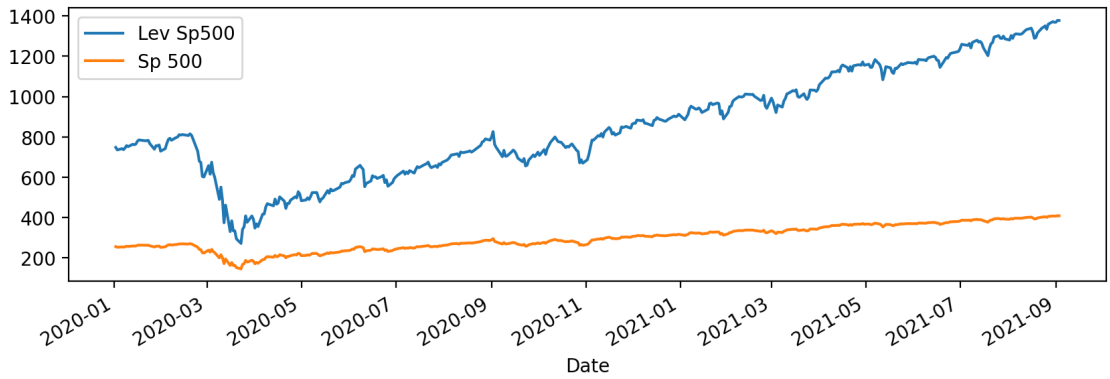
[47]:
```
<matplotlib.legend.Legend at 0x152a19d7190>
```



So we can see over that time period, it looks like the Lev ETF (Lev SP500) was moving a lot more dramatically than SP500. To make it statically more meaningful, lets code it like below.

[48]:
```
plt.figure(figsize=(10,3),dpi=200)
sp_lev_ETF['Percent Change']['2020':'2021'].plot(label='Lev Sp500')
sp500['Percent Change']['2020':'2021'].plot(label='Sp 500')
plt.legend()
```

[48]:
```
<matplotlib.legend.Legend at 0x152a37d2410>
```



[49]:
```
sp_lev_ETF['Daily Returns'] = sp_lev_ETF['Adj Close'].pct_change()
```

[50]:
```
sp_lev_ETF = sp_lev_ETF.dropna()
```

And now it's time to get beta and alpha beta and alpha of S&P Leveraged ETF.

[51]:
```
beta_and_alpha(sp_lev_ETF)
```

[51]:
```
(2.010135618299088, -7.749523800499192e-05)
```

14 COMMENT

So this ETF is doing its job.It's giving you 2x exposure to the market's beta.So when the market goes up by, let's say, 1%, this leveraged ETF is going to try to go up by 2% and when it goes down by, let's say, one percent, it's going to end up going down by double 2%.And that scales with it.