Financial Engineering II

Project

Name - Abana Sidhu

Roll no - 200123001

Pairs Trade

Overview

Pair trading is the basic form of statistics arbitrage. It relies on the assumption that two cointegrated stocks would not drift too far away from each other. First step, we select two stocks and run Engle-Granger two step analysis. Once the criteria of cointegration is met, we standardize the residual and set one sigma away (two tailed) as the threshold. After that, we compute the current standardized residual of the selected stocks accordingly. When the standardized residual exceeds the threshold, it generates the trading signal. The simple rule is we always long the cheap stock and short the expensive stock.

Strategy

A pairs trade or pair trading is a market neutral trading strategy enabling traders to profit from virtually any market conditions: uptrend, downtrend, or sideways movement. This strategy is categorized as a statistical arbitrage and convergence trading strategy.

1. To explain what my strategy exactly does and how it is working, i have taken the example of the Indian Market, i have taken a few securities over a specified time interval. Then i have found all the cointegrated pairs of stocks from all the possible pairs of securities by considering all the stocks having p-value less than a certain cut-off. For further analysis, i have identified the security pair with minimum p-value.
2. Next i have calculated the spread of the two series. In order to actually calculate the spread, i have used a linear regression to get the coefficient for the linear combination to construct between our two securities. Since, the absolute spread isn't very useful in statistical terms. It is more helpful to normalize our signal by treating it as a z-score. A Z-score is nothing but a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean.
3. Next i define my strategy and generate the trading signals during backtesting on another time period which i have defined as -

    - Go "Long" the spread whenever the z-score is below -1.0
    - Go "Short" the spread when the z-score is above 1.0
    - Exit positions when the z-score approaches zero
    - Next i have created calculated the returns of our portfolio on the basis of strategy

Normal And Lognormal Distribution for Stock Prices

Stock Prices can be modeled using the Lognormal Distribution as long as we assume the growth factor to be be distributed normally.

This also alleviates the problem caused by normal distributions having a negative side as stock prices cannot have negative values. Returns are usually assumed to be originating from Normal distribution so our modelling provides us with unique advantages.

Random Walk Hypothesis

According to the Random Walk Hypothesis, stock prices evolve according to a geometric random walk. That is,

$$S_t = S_{t-1}(1+\alpha) \tag{1}$$

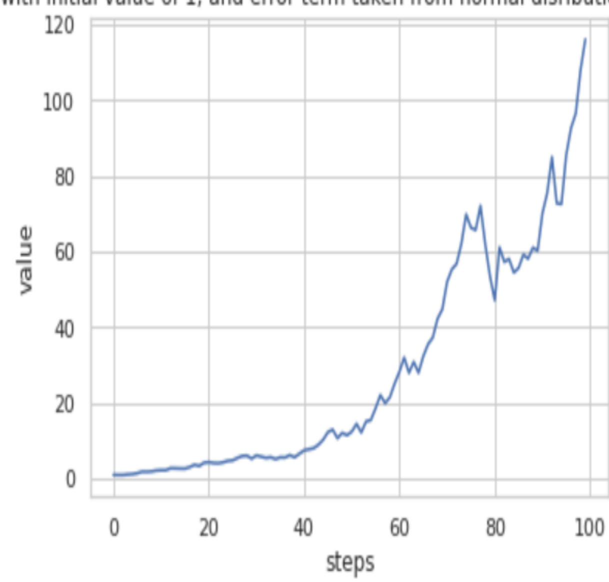
where,

S_t is the price of a stock at some time t with t being in a the form of some discrete time steps.
α is a random variable satisfying $\alpha \sim N(\mu, \sigma^2)$
This implication of this hypothesis is that stock prices cannot be predicted. To explain this random walk better, we have simulated a geometric random walk in the following section.

Here, error_term corresponds to α which satisfies $\alpha \sim N(0.05, 0.1)$.

The walk is initialized with $S_0 = 1$

Geometric Random Walk Simulation with initial value of 1, and error term taken from normal disribution with mean 0.05 and standard deviation 0.1



Data importing and processing

```
In [ ]: symbols =  ['AXISBANK.NS' , 'BAJAJFINSV.NS' ,'BAJAJHLDNG.NS','BAJFINANCE.NS' , 'BANKBARODA.NS','HDFC.NS','HDFCBANK.NS','ICICIBANK.NS','INDUSINDBK.N
asset_data =  yf.download(symbols, start="2012-03-20" ,end= "2017-03-20")['Adj Close']

asset_data.keys()
```

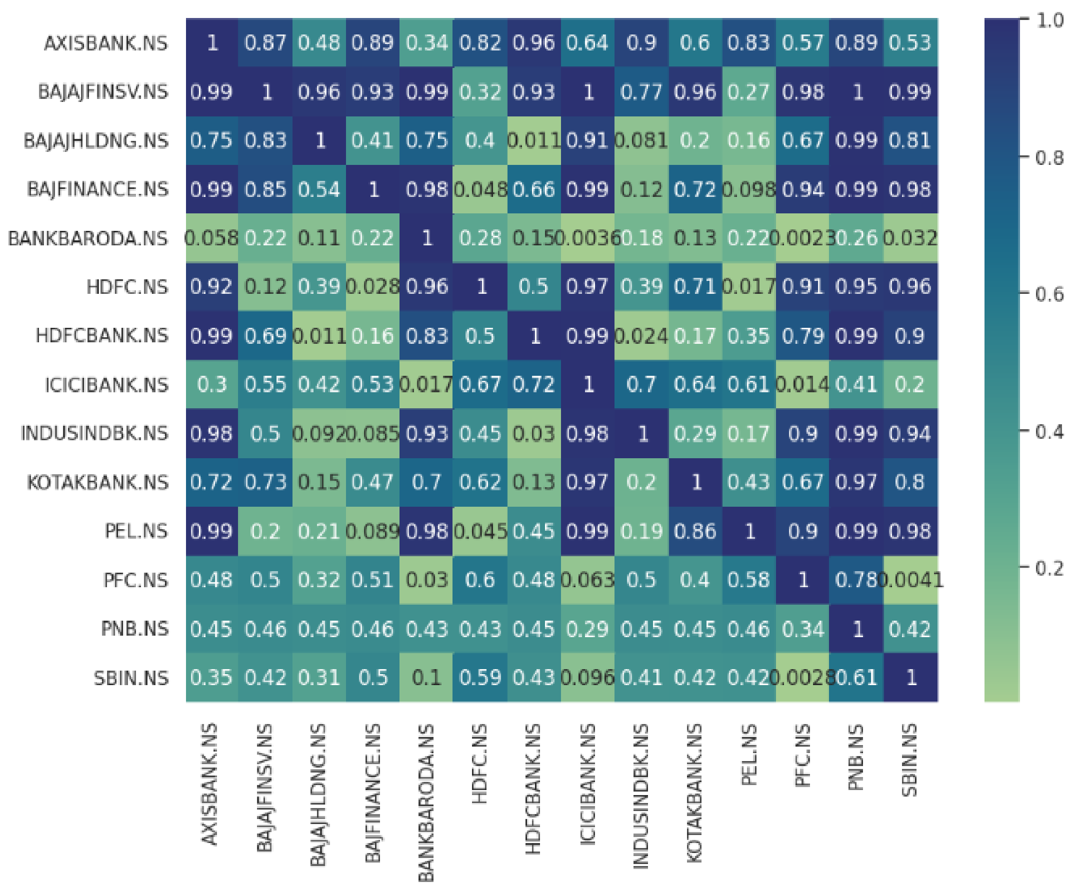
Date	AXISBANK.NS	BAJAJFINSV.NS	BAJAJHLDNG.NS	BAJFINANCE.NS	BANKBARODA.NS	HDFC.NS	HDFCBANK.NS	ICICIBANK.NS	INDUSINDBK.NS
2012-03-20	221.233215	59.313595	644.318298	72.966103	136.869644	558.175354	233.842804	143.148270	296.223663
2012-03-21	229.140091	59.503159	640.184753	74.985901	140.911987	566.664612	238.585327	147.157867	301.369995
2012-03-22	219.396408	58.443542	639.985962	75.170341	135.729492	559.618591	233.310684	141.738235	293.013000
2012-03-23	219.927856	59.230961	647.378723	75.105797	138.743973	560.764587	237.798767	143.455505	294.287781
2012-03-26	209.820526	58.215096	640.145020	74.709206	136.385956	552.869446	236.734604	137.460785	289.755249

Cointegration

Cointegration is a statistical method used to test the correlation between two or more non-stationary time series in the long run or for a specified period.

Function for finding cointegration of two time series

```python
def find_cointegrated_pairs(data):
    n = data.shape[1]
    score_matrix = np.zeros((n,n))
    pvalue_matrix = np.ones((n,n))
    keys = data.keys()
    pairs = []
    for i in range(n):
        for j in range(n):
            if i == j:
               continue
            s1 = data[keys[i]]
            s2 = data[keys[j]]
            result = coint(s1,s2)
            score = result[0]
            pvalue = result[1]
            score_matrix[i,j] = score
            pvalue_matrix[i,j] = pvalue
            if (pvalue < 0.05 and pvalue!=0 and keys[i]!=keys[j]):                #cutoff
                pairs.append((pvalue, (keys[i], keys[j])))
    return score_matrix, pvalue_matrix, pairs
```



Next Identify the pairs of stocks having a co-integrating relationship, i.e, they have p-values less than the defined cutoff of 0.05. p-value is a measure of how strong the relationship between two series is. The lower the pvalue the better generally.

```
[(0.010966848895415472, ('BAJAJHLDNG.NS', 'HDFCBANK.NS')),
 (0.0480459132152O773, ('BAJFINANCE.NS', 'HDFC.NS')),
 (0.0035997410082O799, ('BANKBARODA.NS', 'ICICIBANK.NS')),
 (0.0023112934783160884, ('BANKBARODA.NS', 'PFC.NS')),
 (0.03230422759992858, ('BANKBARODA.NS', 'SBIN.NS')),
 (0.028424199218378486, ('HDFC.NS', 'BAJFINANCE.NS')),
 (0.017351659963185134, ('HDFC.NS', 'PEL.NS')),
 (0.0107723282871 6199, ('HDFCBANK.NS', 'BAJAJHLDNG.NS')),
 (0.024201566525215334, ('HDFCBANK.NS', 'INDUSINDBK.NS')),
 (0.017181766888814034, ('ICICIBANK.NS', 'BANKBARODA.NS')),
 (0.014297710599519523, ('ICICIBANK.NS', 'PFC.NS')),
 (0.029744183415609127, ('INDUSINDBK.NS', 'HDFCBANK.NS')),
 (0.045397413661 8473, ('PEL.NS', 'HDFC.NS')),
 (0.029614678614128833, ('PFC.NS', 'BANKBARODA.NS')),
 (0.004093980840363093, ('PFC.NS', 'SBIN.NS')),
 (0.0028155414619155422, ('SBIN.NS', 'PFC.NS'))]
```

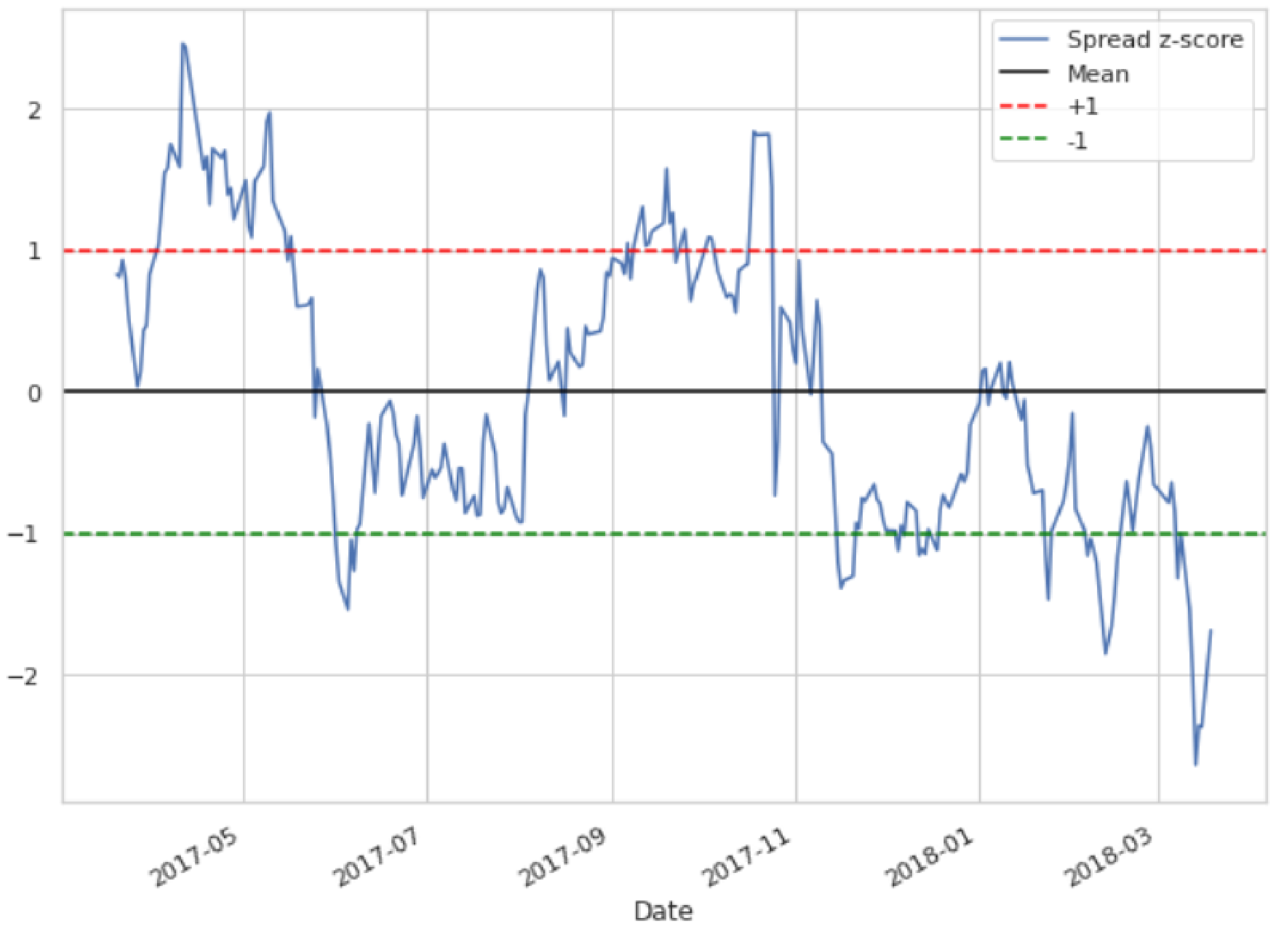
A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. Z-scores also make it possible to adapt scores from data sets having a very different range of values to make scores that can be compared to one another more accurately.

Here, i have used Z-score to perform normalization on the spread of the securities

$$z = \frac{x - \mu}{\sigma}$$

where,
x is a series
μ is the series mean
σ is the series standard deviation
z is the z-score

```
def zscore(series):
    return (series - series.mean()) / np.std(series)
```



Backtesting

Trading signal generation

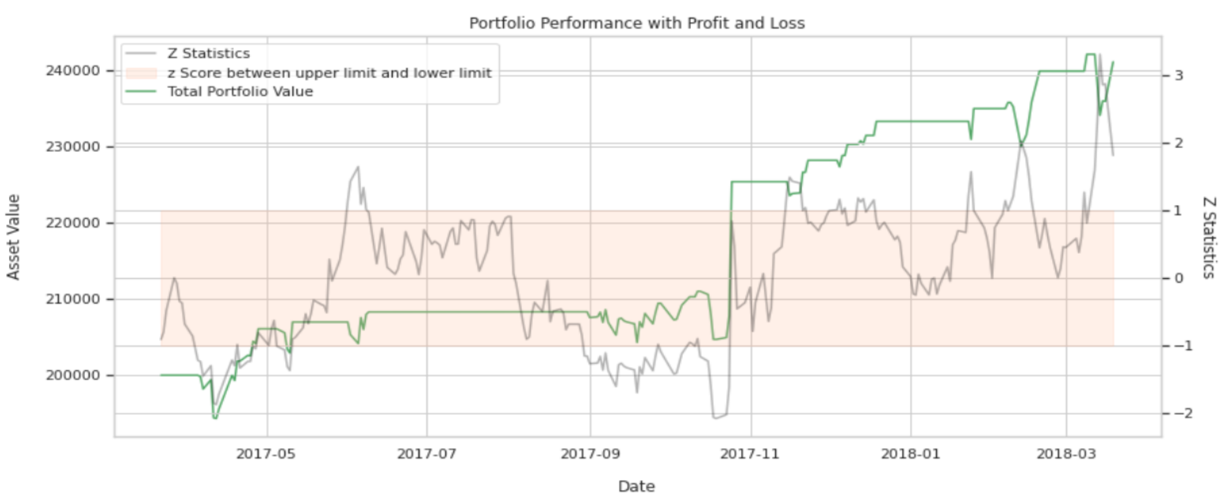
```
#Dataframe for trading signals
def signal_pairs(x,y):
  signals = pd.DataFrame()
  signals['asset1'] = x
  signals['asset2'] = y

  #Calculating Z score of spread and defining upper and lower threshhold
  ratio = signals['asset1']/signals['asset2']
  signals['z'] = zscore(ratio)
  signals['z upper limit'] = np.mean(signals['z']) + np.std(signals['z'])
  signals['z lower limit'] = np.mean(signals['z']) - np.std(signals['z'])
  signals['signals1'] = 0
  signals['signals1'] = np.select([signals['z'] >signals['z upper limit'], signals['z'] < signals['z lower limit']], [-1, 1], default=0)

  signals['positions1'] = signals['signals1'].diff()
  signals['signals2'] = -signals['signals1']
  signals['positions2'] = signals['signals2'].diff()

  return signals
```

Date	asset1	asset2	z	z upper limit	z lower limit	signals1	positions1	signals2	positions2
2017-03-20	159.604233	97.562065	-0.773233	1.0	-1.0	0	NaN	0	NaN
2017-03-21	156.953796	96.122887	-0.794990	1.0	-1.0	0	0.0	0	0.0
2017-03-22	155.749069	96.357178	-0.911361	1.0	-1.0	0	0.0	0	0.0
2017-03-23	156.327316	95.754723	-0.796859	1.0	-1.0	0	0.0	0	0.0
2017-03-24	162.929306	97.160431	-0.483701	1.0	-1.0	0	0.0	0	0.0

Plotting signals

Profit and Loss Calculations

```
#returns portfolio dataframe which containst the total assets at every time step

def pnl_calculation(signals):

  initial_capital = 100000

  # shares to buy for each position
  positions1 = initial_capital// max(signals['asset1'])
  positions2 = initial_capital// max(signals['asset2'])

  #pnl for the 1st asset
  portfolio = pd.DataFrame()
  portfolio['asset1'] = signals['asset1']
  portfolio['holdings1'] = signals['positions1'].cumsum() * signals['asset1'] * positions1
  portfolio['cash1'] = initial_capital - (signals['positions1'] * signals['asset1'] * positions1).cumsum()
  portfolio['total asset1'] = portfolio['holdings1'] + portfolio['cash1']
  portfolio['return1'] = portfolio['total asset1'].pct_change()
  portfolio['positions1'] = signals['positions1']

  # pnl for the 2nd asset
  portfolio['asset2'] = signals['asset2']
  portfolio['holdings2'] = signals['positions2'].cumsum() * signals['asset2'] * positions2
  portfolio['cash2'] = initial_capital - (signals['positions2'] * signals['asset2'] * positions2).cumsum()
  portfolio['total asset2'] = portfolio['holdings2'] + portfolio['cash2']
  portfolio['return2'] = portfolio['total asset2'].pct_change()
  portfolio['positions2'] = signals['positions2']

  # total pnl and z-score
  portfolio['z'] = signals['z']
  portfolio['total asset'] = portfolio['total asset1'] + portfolio['total asset2']
  portfolio['z upper limit'] = signals['z upper limit']
  portfolio['z lower limit'] = signals['z lower limit']
  portfolio = portfolio.dropna()

  return portfolio
```



```
def calculate_cagr(portfolio):

    # calculate CAGR
    final_portfolio = portfolio['total asset'].iloc[-1]
    initial_portfolio = portfolio['total asset'].iloc[0]
    delta = (portfolio.index[-1] - portfolio.index[0]).days
    print('Number of days = ', delta)
    YEAR_DAYS = 365
    returns = (final_portfolio/initial_portfolio) ** (YEAR_DAYS/delta) - 1
    return returns
```

```
print('CAGR = {:.3f}%' .format(calculate_cagr(portfolio) * 100))
```

```
Number of days =  362
CAGR = 20.750%
```

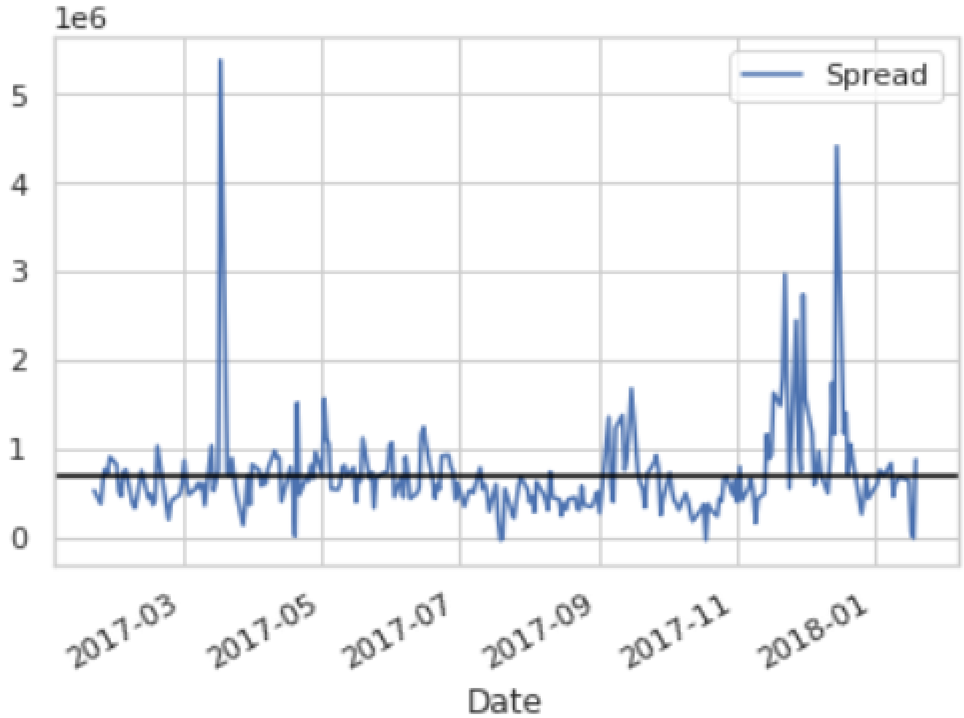
Alternative Strategy

A strategy based on Volumes of stocks whose prices are from a Cointegrated Series. Based on very Rudimentary concepts of volume and price action where we assume volumes mean reverting.

As volume increases, we assume that prices must go up to drive down volume and as volume decreases prices go down to drive up volume.

Using a better metric for the correlation of price and volume is surely bound to increase the CAGR through this strategy and also reduce Drawdown.

```
symbols_pairs_us =  ['ASML', 'AJG']
asset_pairs_us =  yf.download(symbols_us, start="2017-01-20" ,end= "2018-01-20")['Volume']
price_1 =  yf.download(symbols_pairs_us[0], start="2017-01-20" ,end= "2018-01-20")['Adj Close']
price_2 =  yf.download(symbols_pairs_us[1], start="2017-01-20" ,end= "2018-01-20")['Adj Close']
S1_us=asset_pairs_us['ASML']
S2_us=asset_pairs_us['AJG']

S1_us = sm.add_constant(S1_us)
results = sm.OLS(S2_us, S1_us).fit()
S1_us = S1_us['ASML']
b = results.params['ASML']

spread = S2_us - b * S1_us
spread.plot()
plt.axhline(spread.mean(), color='black')
plt.legend(['Spread']);
```



	asset1	asset2	z	z upper limit	z lower limit	signals1	positions1	signals2	positions2
Date									
2017-01-20	114.727303	47.270737	-0.021075	1.0	-1.0	0	NaN	0	NaN
2017-01-23	114.840096	47.181278	1.324656	1.0	-1.0	1	1.0	-1	-1.0
2017-01-24	115.197311	47.619629	-0.028375	1.0	-1.0	0	-1.0	0	1.0
2017-01-25	115.413498	48.120605	-0.539837	1.0	-1.0	0	0.0	0	0.0
2017-01-26	113.862495	48.120605	-0.535208	1.0	-1.0	0	0.0	0	0.0
...	...	...	...	...	...	...	...	...	...
2018-01-12	171.706223	58.674282	-0.363351	1.0	-1.0	0	0.0	0	0.0
2018-01-16	176.937073	58.251766	0.546453	1.0	-1.0	0	0.0	0	0.0
2018-01-17	189.088562	58.830425	3.852743	1.0	-1.0	1	1.0	-1	-1.0
2018-01-18	193.056808	58.812061	4.440648	1.0	-1.0	1	0.0	-1	0.0
2018-01-19	195.107376	59.050858	-0.171911	1.0	-1.0	0	-1.0	0	1.0

```
print('CAGR = {:.3f}%' .format(calculate_cagr(portfolio_us_vol) * 100))
```

```
Number of days =  360
CAGR = 3.907%
```

Plotting signals

Portfolio Performance with Profit and loss