BETA NEUTRAL PORTFOLIO OPTIMIZATION

A Python Analysis

FE520 Introduction to Python for Financial Applications

Names: Mary Nyone 10444257
Raj Ramchandani 10453427
Zeal Shah 10442898

Overview

Modern Portfolio Theory as proposed by Nobel Laureate Harry Markowitz remains hugely popular today due to its simplicity and performance. The idea developed by Markowitz provides insight to how risk-averse investors can construct portfolios to optimize or maximize returns based on a given level of market risk. This structure emphasizes that risk is an inherent part of a higher reward. Creating optimal portfolios has not been met without challenges. The Capital Asset Pricing Model (CAPM) describes the relationship between the expected return and risk of investing in a security. It shows that the expected return on a security is equal to the risk-free return plus a risk premium which is based on the beta of that security. Here, we are introduced to the term Beta.

Objective:

Working as an adviser to a high net worth individual, we are required to produce an optimal investment strategy to minimize market exposure in his/her portfolio. Here, we identify, a beta-neutral or zero-beta portfolio would be ideal in this case. Our investor, with an investment of $1m dollars has a low risk tolerance and is primarily concerned with capital preservation with a set level of returns. With this in mind, we tend to invest in blue chip corporations that are diversified across multiple sectors.

What is a Beta-Neutral Portfolio?

A beta neutral portfolio is a portfolio constructed to have zero systemic risk. According to the Capital Asset Pricing Model (CAPM) beta coefficients are a measure of the volatility, or systematic risk of an individual stock in comparison to the unsystematic risk of the entire market. Beta is used to help investors understand whether a stock moves in the same direction as the rest of the market, and how volatile or risky it is compared to the market. The market itself has a beta of 1. For example, a stock whose returns vary less than the market's returns has a beta with an absolute value less than 1.0. A stock with a beta of 2 has returns that change, on average, by twice the magnitude of the overall market. A stock with a beta of 1 moves exactly with the market. Beta can calculate with the below formula,

$$\beta_p = \frac{Cov\,(r_p,\ \ r_b)}{Var\,(r_b)}$$

Where $\beta_p = Beta\ of\ the\ portfolio$

$Cov\,(r_p,\ \ r_b)$ = Covariance between the return of the portfolio and benchmark

$Var\,(r_b)$ = Variance of the return of the benchmark

Our goal is to create a portfolio where the sum of betas of stocks in our portfolio becomes 1

$$\sum_{i=1}^{N} \beta_i \, \omega_i = 0$$

Where β_i is the beta of each stock,
and ω_i refers to the weight allocated to each stock

Our optimization problem therefore becomes;

Minimize $\omega^T \sum \omega$ *minimize the portfolio variance*

Subject to

$\omega^T \mu \geq \mu^*$ *The expected mean return is at least some target return*

$\omega^T 1=1$ *Weight sum up to 100% (note, we do allow long and short positions)*

$\sum_{i=1}^{N} \beta_i \, \omega_i = 0$*Beta of portfolio is 0*

Importing Libraries

In our portfolio analysis, the following functions are used:

1. **ffn - Financial Functions for Python -** ffn is a library that contains many useful functions for those who work in **quantitative finance**. It stands on the shoulders of giants (Pandas, Numpy, Scipy, etc.) and provides a vast array of utilities, from performance measurement and evaluation to graphing and common data transformations.

2. **empyrical:** Used for common financial risk and performance- We used it to Calculate Beta Value

3. **pypfopt:** PyPortfolioOpt is a library that implements portfolio optimization methods, including classical efficient frontier techniques as well as recent developments in the field like shrinkage and CVaR optimization, along with some novel experimental features.

Method:

Selecting Stocks:

We are fixed with a challenge to select stocks for our portfolio. First, we look at a basket comprising of stocks to pick from. In this case, we choose stocks in the Dow Jones 30. The Dow Jones Industrial Average has consisted of most commonly followed equity indices and includes 30 large companies included in the stock exchanges.

Code:

```
price = list()
benchmark = ffn.get('spy', start='2016-11-01',end='2018-11-01')
names = ['mmm','axp','aapl','ba','cat','cvx','csco','ko', 'xom','gs','hd','ibm','intc','jnj','jpm',
         'mcd','mrk','msft','nke','pfe','pg','trv','unh','utx','vz','v','wmt','wba','dis']
```

Next, we select beta of each stock

```
for i in names:

    temp = ffn.get(i, start='2016-11-01',end='2018-11-01')
    price.append(temp)
    alpha, beta = alpha_beta(temp, benchmark)
    print('name: ',i,'\tbeta value: ',beta)
```

Result:

```
name:  mmm      beta value:  0.6166690496672075
name:  axp      beta value:  0.49324124956658455
name:  aapl     beta value:  1.2814752120237054
name:  ba       beta value:  3.338177801341278
name:  cat      beta value:  0.9865648747460943
name:  cvx      beta value:  0.30497004172699455
name:  csco     beta value:  0.2583288771674521
name:  ko       beta value:  0.08051303787606492
name:  xom      beta value:  0.031170028583297853
name:  gs       beta value:  0.31360330199716613
name:  hd       beta value:  1.0575558680664585
name:  ibm      beta value:  -0.11748307706015737
name:  intc     beta value:  0.2664495498066668
name:  jnj      beta value:  0.3017684323440961
name:  jpm      beta value:  0.5479576051769822
name:  mcd      beta value:  0.6851956588427172
name:  mrk      beta value:  0.06132061138922086
name:  msft     beta value:  0.7293870878877612
name:  nke      beta value:  0.4103539452328578
name:  pfe      beta value:  0.14542335785963392
name:  pg       beta value:  -0.035490322207502485
name:  trv      beta value:  0.2993939372109521
name:  unh      beta value:  1.625797995832064
name:  utx      beta value:  0.40906332169405885
name:  vz       beta value:  0.10638289245696372
name:  v        beta value:  0.9158042487147368
name:  wmt      beta value:  0.4300338521121386
name:  wba      beta value:  -0.20816347449337422
name:  dis      beta value:  0.10552614916960182
```

From our calculations, and ranking stocks quantitatively, on characteristics such as value, momentum, liquidity and opinion. We make our decision on the 5 stocks for our portfolio.

- Industrials & HealthCare tend to outperform the market in downtimes. Top component in Dow Industrials & HealthCare is “CAT” and “PFE” respectively
- Diversify in Technology stocks as well. Technology has performed well in recent years. Choose “MSFT” (positive beta value), and “IBM” (negative beta value).
- Finally, pick a stock in retail. “WMT” being a giant retail stock

Therefore, we complete our five choices in the following stocks:

1. “PFE” - Pfizer Inc
2. “MSFT” - Microsoft Corp
3. “CAT” - Caterpillar Inc
4. “IBM” - International Business Machines Corp
5. “WMT” - Walmart Inc

Our Analysis Begins:

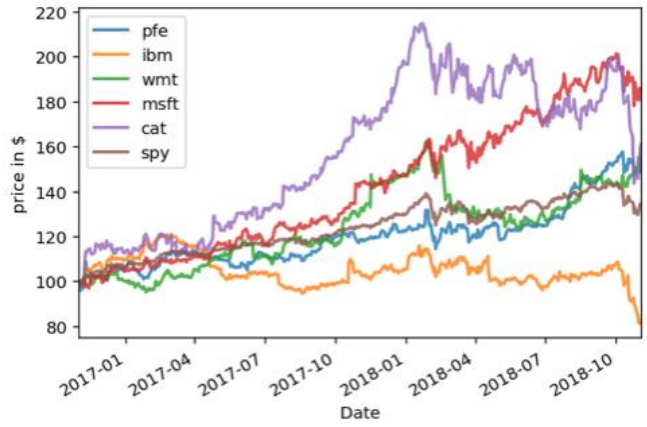
We get prices of each stock for the past two years “2016-11-01” to “2018-11-01” and plot prices against the SPY (our benchmark, representing the market).

Code:

```
#Stock selection is made! Top five stocks for our portfolio
#Plot of each stock to show price evolution within the given time frame including the benchmark
prices1 = ffn.get('pfe,ibm,wmt,msft,cat,spy', start='2016-11-01', end='2018-11-01')
prices = ffn.get('pfe,ibm,wmt,msft,cat', start='2016-11-01', end='2018-11-01')
benchmark = ffn.get('spy', start='2016-11-01')
ax = prices1.rebase().plot(lw=2, alpha=0.8)
plt.ylabel('price in $')
```

Result:

```
Text(0,0.5,'price in $')
```

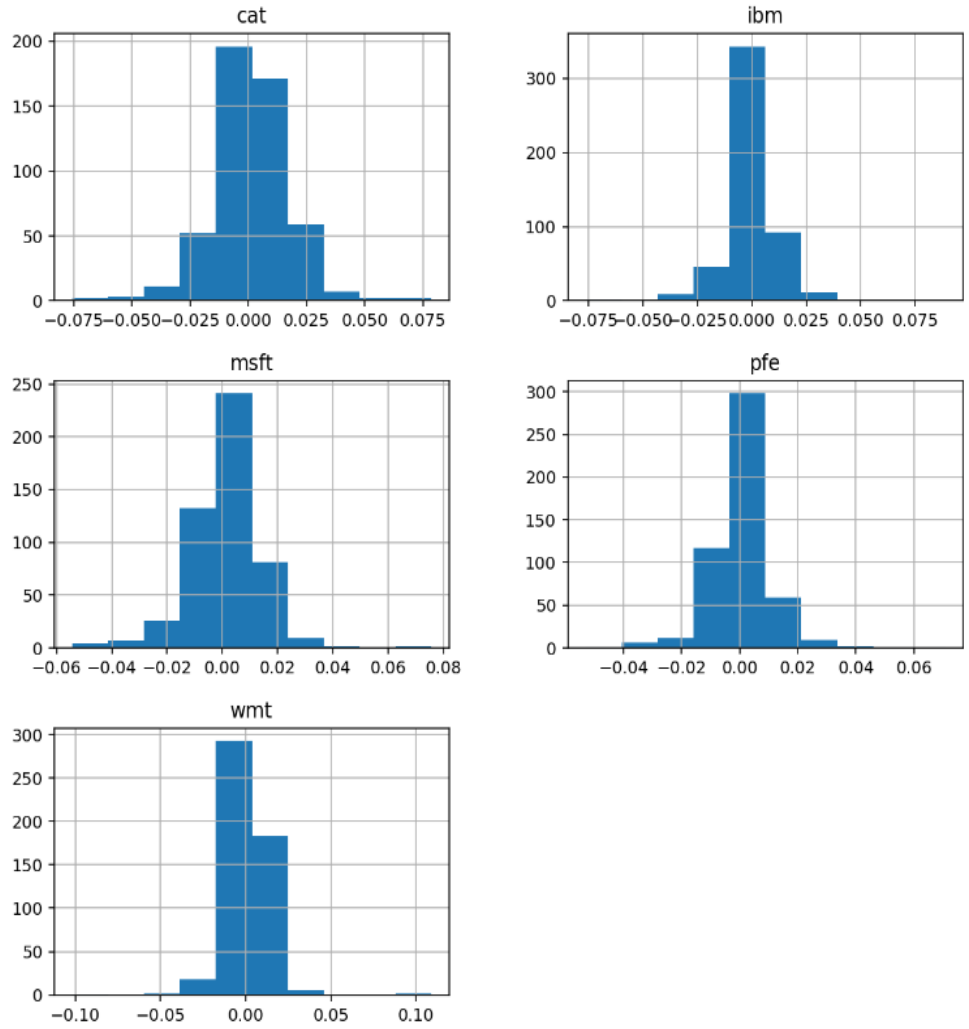


After studying past returns and performance of these assets, we gain insight into the distribution of prices over established timeframe. The expected return will be the center of the distribution. Here, we plot a histogram for the distribution of expected returns and report some important performance matrix.

Code:

```
returns = prices.to_returns().dropna()
ax = returns.hist(figsize=(10,10))
```

Result:



Next, we made the function which calculate beta value from prior two year from the data then it checks the profit in a month and then it calculates the beta value for each and every month Complete process is repeated for 12 times:

Code:

```
def calc(stockStartTime,stockEndTime):
    #calculating 2 years prior of the start date
    tempdate = dt.datetime.fromisoformat(stockStartTime)
    tempdate = tempdate - dt.timedelta(weeks=104)
    prices = ffn.get('pfe,ibm,wmt,msft,cat', start= tempdate.strftime('%Y-%m-%d'), end=stockStartTime)
    # Expected returns and sample covariance
    mu = expected_returns.mean_historical_return(prices)
    S = risk_models.sample_cov(prices)
    #Minimum volatility. May be useful if you're trying to get an idea of how low the volatility could be,
    #but in practice it makes a lot more sense to me to use the portfolio that maximises the Sharpe ratio.
    # Optimise portfolio for maximum Sharpe Ratio to serve as benchmark
    ef = EfficientFrontier(mu, S)
    raw_weights = ef.min_volatility() #ef.max_sharpe()
    cleaned_weights = ef.clean_weights()
    print("Cleaned weights:\n",cleaned_weights)
    print("Portfolio Performance:\n",ef.portfolio_performance(verbose=True))
    #To achieve beta neutrality
    ef = EfficientFrontier(mu, S, weight_bounds=(-1, 1))
    print(" Weights: ",ef.efficient_return(target_return=0.15, market_neutral=True))
    weights = ef.efficient_return(target_return=0.2, market_neutral=True)
    weight_sum = sum(w for w in weights.values() if w > 0)
    normalised_weights = {k:v/weight_sum for k,v in weights.items()}
    #print("Normalized weights: ",normalised_weights)
    #We then need to convert these weights into an actual allocation, telling you how many-
    #shares of each asset you should purchase.
    latest_prices = get_latest_prices(prices)
    da = DiscreteAllocation(weights, latest_prices, total_portfolio_value=1000000)
    allocation, leftover = da.lp_portfolio()
```

```
    #print(allocation)
    print("")
    for key,val in allocation.items():
        print("Number of positions in ",key," stock: ",val)
    print("")
    print("Funds remaining: ${:.2f}".format(leftover))
    print("")
    prices2 = ffn.get('pfe,ibm,wmt,msft,cat', start=stockStartTime, end=stockEndTime)
    latest_prices2 = get_latest_prices(prices2)
    sum1 = 0
    for key,val in allocation.items():
        sum1 = sum1 - (latest_prices[key]*val)
    print("Value of Portfolio after short sales :\t",abs(sum1))
    new = 0
    for key,val in allocation.items():
        new = new + (latest_prices2[key]*val)
        sum1 = sum1 + (latest_prices2[key]*val)
    print("Value at end of period :\t\t",new)
    print("Profit at end of time period :\t\t",sum1)
    return sum1
```

```
calc(stockStartTime = "2018-11-01",stockEndTime = "2018-12-01")
```

Result:

```
Cleaned weights:
 {'pfe': 0.36653, 'ibm': 0.21878, 'wmt': 0.22718, 'msft': 0.16404, 'cat': 0.02347}
Expected annual return: 18.3%
Annual volatility: 12.7%
Sharpe Ratio: 1.28
Portfolio Performance:
 (0.18256989382586908, 0.12675995933515896, 1.2825019405065212)
 Weights:  {'pfe': 0.07780753417839327, 'ibm': -0.40095262697007006, 'wmt': 0.057495506014782254, 'msft': 0.22245907360138628,
'cat': 0.04319051317550822}
0 out of 5 tickers were removed

Allocating long sub-portfolio:
0 out of 4 tickers were removed

Allocating short sub-portfolio:
0 out of 1 tickers were removed

Number of positions in  pfe  stock:  4933
Number of positions in  wmt  stock:  1239
Number of positions in  msft  stock:  5424
Number of positions in  cat  stock:  899
Number of positions in  ibm  stock:  -2752

Funds remaining: $87.22

Value of Portfolio after short sales :   700043.0845222473
Value at end of period :                 722801.6747055054
Profit at end of time period :           22758.590183258057
```

Rebalancing:

We calculate new beta values every month and get new allocations to maintain our beta neutral portfolio. The 2 years data is shifted on a rolling basis. Our returns show the profit/loss calculated each month as shown below.

Complete process is repeated for 12 times:

For the 12 months:

Code:

```
def totalret(startDate):
    tempdate = dt.datetime.fromisoformat(startDate)
    monthend = tempdate + dt.timedelta(days=30)
    print("monthend:",monthend)
    total = 0
    monthlyRet = list()
    percentage = list()
    for i in range(12):
        print("\n\n********************",tempdate.strftime('%d %b %Y'),"**********************\n\n")
        ret = calc( stockStartTime = tempdate.strftime('%Y-%m-%d') , stockEndTime = monthend.strftime('%Y-%m-%d'))
        monthlyRet.append(ret)
        total = total + ret
        tempdate = tempdate + dt.timedelta(days=30)
        print("temp:",tempdate)
        monthend = monthend + dt.timedelta(days=30)
        print("monthend:",monthend)
        i = i + 1
    print("\n")
    for i in monthlyRet:
        print("Returns in percentage:", i/1000000)
    print("\n\n\n**********The total return for 1 year : ", total,"******************")
```

Result:

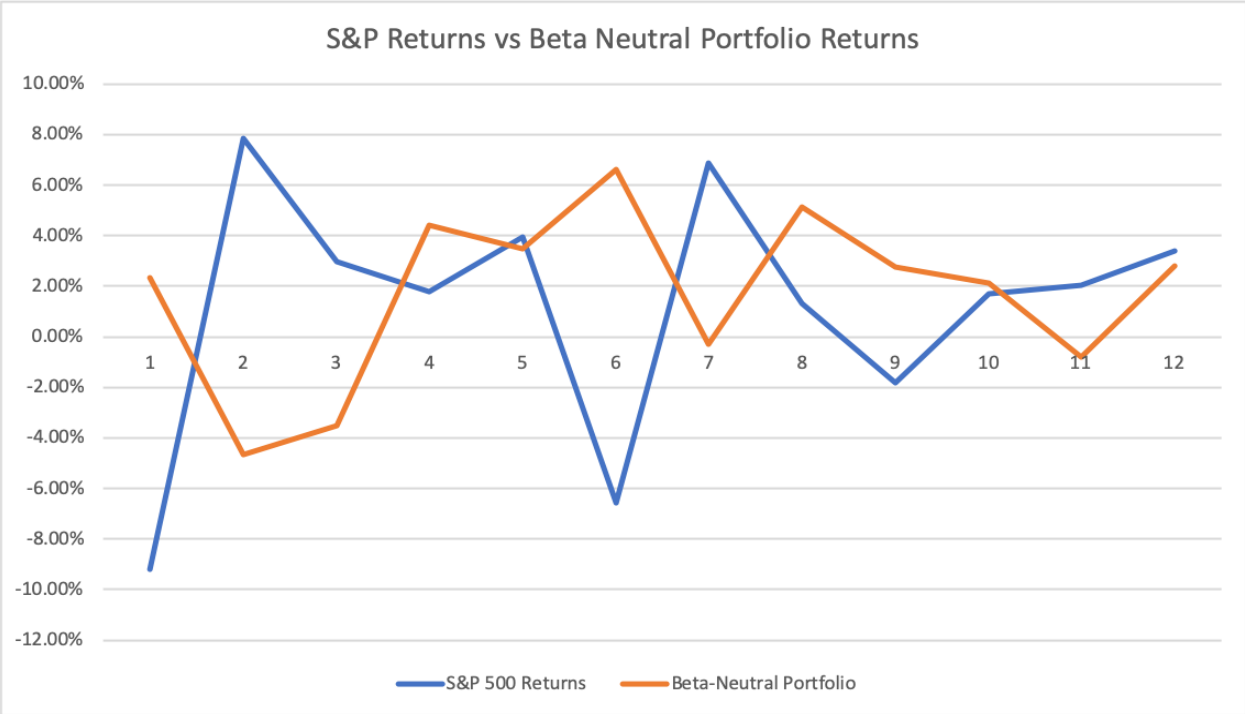
```
Returns per time period: $22758.59      Returns in percentage: 0.022758590183258057
Returns per time period: $-46756.18     Returns in percentage: -0.046756176795959475
Returns per time period: $-34914.72     Returns in percentage: -0.03491472280883789
Returns per time period: $44175.05      Returns in percentage: 0.044175052219390866
Returns per time period: $34958.26      Returns in percentage: 0.03495826466751099
Returns per time period: $66078.62      Returns in percentage: 0.06607862036895752
Returns per time period: $-3512.68      Returns in percentage: -0.003512680061340332
Returns per time period: $51446.52      Returns in percentage: 0.05144651662826538
Returns per time period: $27355.20      Returns in percentage: 0.027355202072143554
Returns per time period: $21227.38      Returns in percentage: 0.021227376876831055
Returns per time period: $-8066.15      Returns in percentage: -0.008066154880523682
Returns per time period: $28035.33      Returns in percentage: 0.02803533045196533


**********The total return for 1 year :  202785.21892166138 *****************
```

Back testing and Comparing our Results:

Here, we compare our portfolio performance for the 12 months of investment with that of the S&P 500 (our benchmark). We get the following returns for each.

DATE	S&P 500 RETURNS	BETA-NEUTRAL PORTFOLIO
DEC 31 2018	-9.18%	2.32%
JAN 31 2019	7.87%	-4.68%
FEB 28 2019	2.97%	-3.5%
MAR 31 2019	1.79%	4.4%
APR 30 2019	3.93%	3.5%
MAY 31 2019	-6.58%	6.6%
JUN 30 2019	6.89%	-0.3%
JULY 31 2019	1.31%	5.14%
AUG 31 2019	-1.81%	2.74%
SEPT 30 2019	1.72%	2.12%
OCT 31 2019	2.04%	-0.8%
NOV 30 2019	3.40%	2.8%

Our return is shown in the graph below:



Limitations of our Model:

We found the following limitations/challenges

- In real life, a market-neutral portfolio is not self-funded as brokers who lend shares for short selling require collateral. If the spread keeps widening after the long/short portfolio was set up, an investor may receive a margin call from the broker. In our model, we do not assume this risk as we assumed unlimited patience and resources for living through periods of negative returns.
- Reality is much more complicated than theory. Hedge fund managers use other tools e.g verticals and put strategies to hedge trade portfolios.
- Voiding/deleting any of the stock in not possible in our model, e.g if a company goes bankrupt.
- In our model, we ignore transaction costs.

Conclusion:

Our beta-neutral portfolio provided a return of $203,313.03 which is about 20% of our initial investment. Not only was this profitable, but in comparing with the market, there was almost a negative correlation with the returns of the S&P. This is great in downtimes but could pose questions during a bullish trend when prices of an industry's stocks or the overall rise in broad market indices go higher. Would our portfolio behave similarly or differently? Only time would tell.

References:

[1]Charlotte Werger - Modern portfolio theory INTRODUCTION TO PORTFOLIO ANALYSIS IN PYTHON https://s3.amazonaws.com/assets.datacamp.com/production/course_18408/slides/chapter4.pdf

[2]**ffn - Financial Functions for Python** https://pmorissette.github.io/ffn/

[3]Markowitz, H. (1952). Portfolio Selection. The Journal of Finance, 7(1), 77 91. https://doi.org/10.1111/j.1540-6261.1952.tb01525.x

[4]Mcgreco. Three approaches to calculate monthly returns...with Pandas! (2018, January 15). Retrieved from https://quantdare.com/calculate-monthly-returns-with-pandas/.

[5]robertmartin8/**PyPortfolioOpt** github

[6]https://mybinder.org/v2/gh/mmngreco/quantdare_posts/master?filepath=calcular_retornos/aggregate_returns.ipynb

S&P 500 Monthly Return: (n.d.). Retrieved from https://ycharts.com/indicators/sp_500_monthly_return.

William F. Sharpe, 1964. "**Capital Asset Prices: A Theory Of Market Equilibrium Under Conditions Of Risk**," Journal of Finance, American Finance Association, vol. 19(3), pages 425-442, September.