

MIE479 Capstone Final Report

Bowen Chen (999657223)

Jiaen Jessica Leung (1000661740)

Yinger Li (999537899)

Instructor: Prof. Roy Kwon,
Minho Lee

Table of Content

Table of Contents

Table of Content	2
1. Introduction	4
2. Background	4
2.1. Motivation	4
2.2 Availability of Existing Decision Support Systems	5
2.3 Asset Allocation Models	5
2.3.1 Mean Variance Optimization (MVO)	6
2.3.2 Black-Litterman model	8
2.3.3 Relative Strength Index	9
2.3.4 Stochastic Oscillator	9
2.3.5 Earning Per Share.....	10
3. Backend	10
3.1 Get S&P 500 Data	10
3.2 Get Prices data	11
3.2.1 Return data	11
3.2.2 Find Momentum Indicators	12
3.3 Get Market Portfolio Weights	14
3.4.1 Use Momentum Indicators to generate views for the Quick Create Model	15
3.4.2 Use request.form.getlist() to get user input views for Professional Model	15
3.5 Update Views	15
3.6 Input parameters into Black-Litterman	16
3.7 Package used	18
4. Front End	18
4.1 Technology choices	18
4.2 Landing Page and Navigation User Interface	20
4.3 Non-Professional User Front-End Interface	21
4.3 Professional User Front-End Interface	22
4.4 Professional User Front-End Interface	24
4.5 Sample results.....	25
4.5.1 Quick create	25
4.5.2 Professional	27
5. Product Design Flow	28
6. Evaluation	29

6.1 Strength	29
6.2 Weakness and Improvements	29
Appendix A: Proposal Feedback from Minho	31
Appendix B: Works Cited	32
Appendix C: File Structure	33
Appendix D: Back end	34
<i>app.py</i>	34
Appendix E: Front end.....	40
<i>base.html</i>	40
<i>index.html</i>	46
<i>portfolio.html</i>	52
<i>login.html</i>	53
<i>customportfolio.html</i>	55
<i>result.html</i>	56
<i>customresult.html</i>	60
Appendix F: Business logic	61
<i>Black_Litterman.py</i>	61
<i>Return_Data_Collector.py</i>	65

1. Introduction

Our team aimed to build an asset allocation support system that could customize equity portfolios for investors with various knowledge of financial modeling and optimization techniques, ranging from nonprofessional to institutional users. Our original design objective has been refined from “Create a software system that allows a user to create multiple portfolios given a set of assets and associated data” to “Create a decision support system in the form of a web application to recommend optimally weighted equity portfolios targeted towards both professional and non-professional users.”

In this report, we will start by discussing the motivation, as well as the literature review and research into existing products on the market that we conducted in the initial stages of the design process. Then we will show development progress during the intermediary phases and how we reached our current design objective and framing. In addition, we will explain the final design by discussing the business logic, back end, and front end components. Finally, strengths, weaknesses, and future improvements will be discussed.

2. Background

In this section, we will discuss the motivation of our project. We will also summarize the references we used to determine the asset allocation model to use in our application and about some of the existing decision support systems in the market that have helped us come up with the initial ideas for our product.

2.1. Motivation

The stock market is a dynamic system that requires the use of computer technology for to model and make predictions. Market volatility and innovation from competitors in trading have made investment decision making increasingly complicated and risky. In this environment, it is evident that the decision support systems (DSSs) are playing more important roles and improving the quality of decisions made by both institutional and retail investors.

Professionals in finance have relied heavily on decision support technology to generate asset allocation and relative optimization models, as they provide a quick and easy way to generate results. However, non-professional individuals with no background in a finance field may have difficulty understanding these models and the financial jargon used to describe them. It is more difficult for them to make investment decisions that fit their financial goals. In developing our product we recognize the demand for decision support systems, not just from professional users, but also from non-professional, possibly non-experienced users.

2.2 Availability of Existing Decision Support Systems

There are a wide range of decision support systems that are already available in the market. SmartFolio is an online portfolio manager that designed by Bank of Montreal. It is an “Robo-advisor” that can interact with customers online and customize their portfolio based on their income, investment horizon, and tolerance for risk. They have also utilized model parameter estimation as an alternative for users to estimate expected returns, implementing techniques like maximum likelihood estimations of means and covariance, and the Black-Litterman model. SmartFolio’s distinguishing features for the Black-Litterman Models include:

- The market equilibrium portfolio is calculated from market capitalizations of portfolio components and market risk aversion as inputs.
- The posterior covariance matrix is set equal to the prior covariance matrix. This significantly simplifies the computations, while keeping the final results essentially intact.
- Uncertainty in investor’s subjective views can be expressed in two alternative ways:
 - As a global confidence level varying from 0% to 100
 - As separate confidence levels (varying from 0% to 100%) for each view. (SmartFolio, 2015)

Other software like “401K Plus” developed by Retirement Management System Inc., use Mean Variance Optimization as one of their tool to make asset allocation decisions. They use a combination of historical data and current market information to calculate statistical estimates for expected return, expected standard deviation and standard correlation coefficients. They have also considered portfolio spacing to ensure that the risk spread between each portfolio is relatively equal. In addition, the service has also enable the users to select from seven risk-tolerance ranges, from conservative to aggressive, in order to satisfy different needs. (401K Plus, 2015)

There are other softwares incorporate portfolio risk measurement such as Delta-Normal Method, and empirical distribution. They use VaR and CVaR measures and quantify the level of financial risk within a firm or investment portfolio over a specific time frame. The construction of the two values depend on users’ investing horizons and significance level.

2.3 Asset Allocation Models

We started building our asset allocation support system by focusing on the Mean Variance Optimization (MVO) and the Black-Litterman as they are two major portfolio allocations in the finance field. However, we have fully switched to the Black-Litterman since in our testing, MVO always created biased portfolios when there are more stocks selected. MVO resulted in portfolios with unrealistically heavy weights in a small number of assets while Black-Litterman performed well.

Hence, we aimed to use the Black-Litterman as the main asset allocation model, and focus on developing multiple ways to customize Black-Litterman geared towards different user groups.

For the professional users:

- Stocks can be selected from S&P500
- Views can be input by the users
- The market equilibrium portfolio is calculated from market capitalizations of portfolio components and market risk aversion as inputs; covariance matrix will be calculated in python every time based on the different combination of the assets

For the non-professional users:

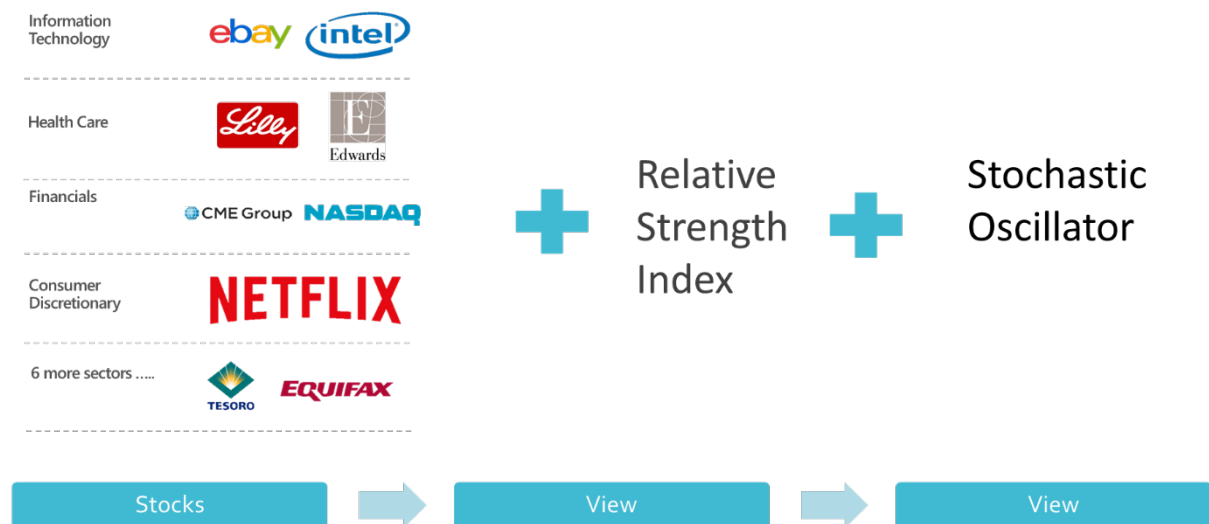


Figure 1: Black-Litterman flow for the non-professional users

- Stocks can be chosen from the list we provide from S&P 500. The list is generated based on the top earning per share performances in different sectors
- Views are generated using Relative Strength Index, and Stochastic Oscillator
- The market equilibrium portfolio is calculated from market capitalizations of portfolio components and market risk aversion as inputs; covariance matrix will be calculated in python every time based on the different combination of the assets

2.3.1 Mean Variance Optimization (MVO)

MVO (Mean Variance Optimization) serves as the foundation of many portfolio management techniques used today. Mean variance optimization takes expected return and risk of assets as inputs, and generates a plot that identifies efficient portfolios composed of weightings of the assets that give the most favorable risk-expected return profile. From the efficient portfolios, the investor can pick the risk-return tradeoff that best suits their level of risk aversion or target return. It can be represented as the problem below:

$$\begin{aligned}
&\text{Minimize} && \sum \sigma_{ij} x_i x_j \\
&\text{Where} && \sum u_i x_i \geq R \\
&&& \sum x_i = 1 \\
&x_i, x_j \geq 0 \quad \forall i, j = 1, 2, 3, \dots, n
\end{aligned}$$

Assume that there are n assets. The mean of the assets (or expected rate of return) are u_1, u_2, \dots, u_n and σ_{ij} represents the covariance matrix of the n assets for $i, j = 1, 2, \dots, n$. A portfolio is defined by a set of n weights $x_i, x_j \geq 0 \quad \forall i, j = 1, 2, 3, \dots, n$ that sum to one. To find a minimum-variance portfolio, we choose either a desired minimum level of return or maximum level of risk that we want to achieve or are willing to accept with the combination of the assets. Then we try to minimize the covariance between each asset while maximizing the return subject to constraints we set. (G.Luenberger, 2013)

In addition, there will be some non-negative weights when short selling is allowed. Short selling is allowed to reflect realistic investment strategies and for greater portfolio diversification.

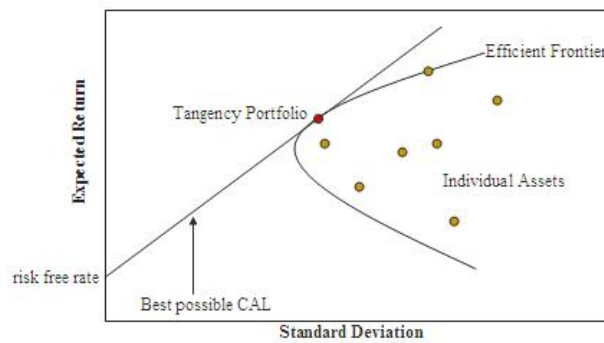


Figure 2: Markov's efficient frontier contains portfolios with the highest expected return for a given standard deviation.

According to Markowitz's theory, there is an optimal portfolio that could be designed with a perfect balance between risk and return. Portfolios that lie below the efficient frontier are sub-optimal, because they do not provide enough return for the level of risk. Portfolios that lie on the efficient frontier may include short selling, while if we are limited by long positions, we may be choosing a sub-optimal and less diverse portfolio (G.Luenberger, 2013).

Even though MVO is a good tool to construct a decision support system, MVO also possesses many limitations that could potentially derail the result produced. Expected returns and covariance are generated from historical data and assumed certain. Realistically, this is not always the case, so using MVO alone could produce a biased portfolio.

2.3.2 Black-Litterman model

The Black-Litterman model combines CAPM, MVO, and specific user views into a single model, creating a sophisticated, yet customizable model. The model has two distinguishing features on how it approaches the problem of asset allocation. First, it uses the equilibrium market portfolio, as a starting point for estimation of asset returns. Second, the Black-Litterman model provides a clear way to specify (Duffie, 1997) investors views on returns and to blend the investors views with prior information.

The problem can be expressed below (Idzorek, 2005):

$$E[R] = [(\tau\Sigma)^{-1} + P'\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\Pi + P'\Omega^{-1}Q]$$

Where τ is a small positive number,

$\Sigma = \text{cov}(r)$, and r is a vector of asset returns,

P is a matrix that identifies the assets involved in the views,

Ω is a diagonal covariance matrix representing the confidence level of the view,

Π is the implied equilibrium return vector,

Q is the view vector.

In addition (Walters, 2014),

$$\begin{aligned}\Pi &= \lambda\Sigma w_{mkt} \\ w &= (\lambda\Sigma)^{-1} \times u \\ \lambda &= \frac{(r - r_f)}{\sigma^2}\end{aligned}$$

Where λ is a risk aversion coefficient that calculated based on the selected assets,

w is the vector of weights invested in each asset,

r is the total return on the market portfolio,

r_f is the risk free rate,

σ^2 is the variance of the market portfolio.

The variance we considered in the model is

$$p_k \Sigma p_k$$

where p_k is a single $1 \times N$ row vector from Matrix P that corresponds to the k th view,

Σ is the covariance matrix of excess returns.

In contrast to MVO, the Black-Litterman model uses reverse optimization to mitigate the issue of sensitivity in MVO to create a diverse portfolio, preventing problems such as concentrating portfolio assets in only a small fraction of the assets considered.

2.3.3 Relative Strength Index

We have used Relative Strength Index to generate one of the view for non-professional users in the Black Litterman. Relative Strength Index (RSI) is developed by J. Welles Wilder, it is used as a momentum oscillator that measures the speed and change of price movements.

RSI oscillates range between zero and 100, and will be considered overbought when above 80 and oversold when below 20. (The Relative Strength Index by Bruce Faber, 1994)

RSI can be calculated using:

$$RSI = 100 - \frac{100}{1 + RS}$$
$$RS = \text{Average Gain} - \text{Average Loss}$$

where

$$\text{First Average Gain} = \frac{\text{Sum of Gains over the past } n \text{ periods}}{n}$$
$$\text{First Average Loss} = \frac{\text{Sum of losses over the past } n \text{ periods}}{n}$$
$$\text{Average Gain} = \frac{\text{previous Average Gain} \times (n - 1) + \text{current gain}}{n}$$
$$\text{Average loss} = \frac{\text{previous Average Loss} \times (n - 1) + \text{current gain}}{n}$$

Wilder's formula normalizes RS and turns it into an oscillator that fluctuates between zero and 100, and provide a visual overview on the stock potential performance. Based on this concept, one of the view is that for all the stock price that has RSI index over 80 and less than 20 will be less than 30% by RSI index between 20 and 80. (Improving the Win-Loss Ratio with the Relative Strength Index by Thomas Bulkowski, 1998)

2.3.4 Stochastic Oscillator

The Stochastic Oscillator was developed by George C. Lane in 1950s. This Stochastic Oscillator is another momentum indicator that shows the location of the close relative to the high-low range over a set number of periods. The calculation has shown below:

$$\%K = \frac{\text{Current Close} - \text{Lowest Low}}{(\text{Highest High} - \text{Lowest Low}) \times 100}$$

Where Lowest Low = lowest low for the look-back period,
Highest High = highest high for the look-back period,
%K is multiplied by 100 to move the decimal point two places.

Since this oscillator is range bound, it can be used to identify overbought and oversold as well. The oscillator ranges from zero to one hundred. Traditional settings use 80 as the overbought threshold and 20 as the oversold threshold. Thus another view of the Black-Litterman will use this characteristic, and displayed as %K over 80 and less than 20 will be less than 30% by %K between 20 and 80. (The Stochastic Oscillator by Joe Luisi, 1997)

2.3.5 Earning Per Share

We chose to use “Earnings Per Share” as a metric in selecting assets for non-professional users. It is the portion of a company's profit allocated to each outstanding share of common stock. Earnings per share serves as an indicator of a company's profitability, and its calculation is list below:

$$EPS = \frac{\text{Net Income} - \text{Dividends on Preferred Stock}}{\text{Average Outstanding Shares}}$$

Stocks with higher EPS are considered to deliver better returns. Thus, after we ask users how many assets from each industry sector they would like to use in the optimization, stocks in each sector will be ranked by their EPS performance, and the top assets will be used. This goal of this selection is to create both a diverse and high performance portfolio.

3. Backend

We use Flask as the backend framework because our team knows python and there are many libraries readily available. Daily financial data is downloaded using python pandas data reader from Yahoo Finance and stored in the PostgreSQL database, which is hosted on the Heroku server. The stored data is then fed to the two completed models (Black Litterman for non-professional quick create and Black-Litterman for professional customized input), which calculates results to be fed to the frontend. The backend will download S&P 500 data and select top n assets (n = 1~5) from each sector based on highest EPS. The size of the portfolio will range from 10 to 50 based on the choice of the assets.

3.1 Get S&P 500 Data

We have two data sources: yahoo finance for all the assets prices data, and data package for the S&P5000 data. When we start the optimization, we get S&P 500 data from the data package, which contains the following information,

'Symbol', 'Name', 'Sector', 'Price', 'Dividend Yield', 'Price/Earnings', 'Earnings/Share',
'Book Value', '52 week low', '52 week high', 'Market Cap', 'EBITDA', 'Price/Sales', 'Price/Book',
'SEC Filings'

This S&P 500 data is saved in a dataframe called *SP500*.

The columns of *SP500* dataset is also used to generate the list of assets that are relevant for the quick create model, which is saved in a list called *list_asset*.

Index	Symbol	Name	Sector	Price	Dividend Yield	Price/Earnings	Earnings/Share	Book Value
0	MMM	3M Company	Industrials	177.12	2.53	22.77	7.78	19.34
1	ABT	Abbott Laboratories	Health Care	41.89	2.55	25.79	1.62	14.10
2	ABBV	AbbVie	Health Care	64.16	3.60	19.29	3.33	2.87
3	ACN	Accenture plc	Information Technology	115.11	1.94	19.45	5.92	11.45
4	ATVI	Activision Blizzard	Information Technology	41.29	0.64	37.06	1.11	11.31
5	AYI	Acuity Brands Inc	Industrials	264.62	0.21	43.23	6.12	36.50
6	ADBE	Adobe Systems Inc	Information Technology	96.79	0.00	54.68	1.77	14.53
7	AAP	Advance Auto Parts	Consumer Discretionary	164.85	0.15	25.20	6.40	35.82
8	AES	AES Corp	Utilities	12.32	3.60	28.99	0.43	4.86
9	AET	Aetna Inc	Health Care	117.00	0.87	17.62	6.64	47.96
10	AMG	Affiliated Managers Group	Financials	138.85	None	15.55	8.93	53.30
11	AFL	AFLAC Inc	Financials	72.49	2.27	11.93	6.07	48.22
12	A	Agilent Technologies Inc	Health Care	45.48	1.04	32.35	1.38	12.79
13	APD	Air Products & Chemicals Inc	Materials	144.35	2.46	56.99	2.53	32.01
14	AKAM	Akamai Technologies Inc	Information Technology	56.38	None	31.85	1.77	17.69
15	ALK	Alaska Air Group Inc	Industrials	59.86	1.88	8.65	6.92	19.81
16	ALB	Albemarle Corp	Materials	83.58	1.57	18.13	4.61	31.53
17	AA	Alcoa Inc	Materials	9.82	1.31	None	-0.44	9.25
18	ALXN	Alexion Pharmaceuticals	Health Care	124.42	None	188.23	0.66	35.96
19	ALLE	Alliegon	Industrials	69.74	0.70	40.59	1.72	0.77
20	AGN	Allergan plc	Health Care	240.59	0.10	21.53	11.18	183.41
21	ADS	Alliance Data Systems	Information Technology	200.96	None	22.65	8.87	29.51
22	LNT	Alliant Energy Corp	Utilities	40.55	3.18	24.30	3.34	16.58
23	ALL	Allstate Corp	Financials	69.66	1.95	17.00	4.10	49.58

Figure 3: get entire stocks list from S&P 500

3.2 Get Prices data

For the quick create model, the $n \times 10$ stocks' daily prices data are then obtained from yahoo finance using the list *list_assets*, which is saved in a pandas dataframe called *result_data*. The close prices data is used in two processes.

For the professional customizable model, the *list_assets* is generated by the user input from the frontend using *request.form.getlist* method from the flask built-in library.

3.2.1 Return data

Per standard industry practices, the data will first be stored locally before being used by the model. The return data will be used across all models to calculate implied returns and covariance. The market cap data will be used to calculate weights of the market portfolio in the Black-Litterman model.

First, the prices data are resampled to monthly end price data. The return data is calculated based on the percentage change from the month end price data. The Return data is saved in dataframes called *return data*.

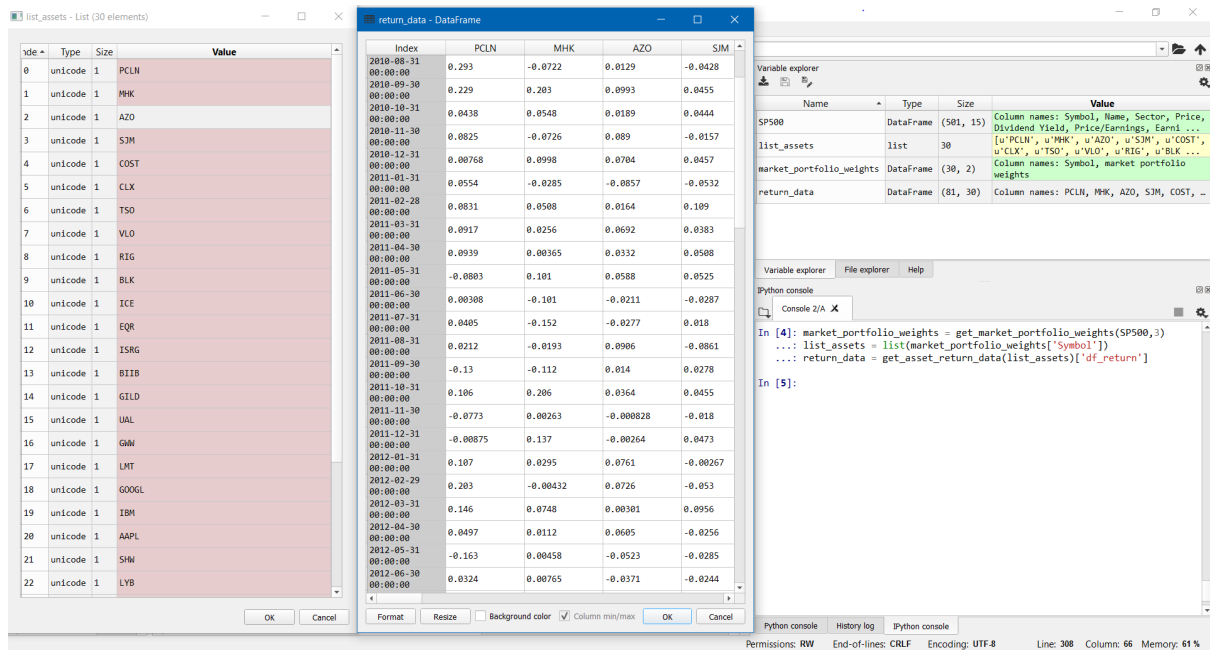


Figure 4: the image on the left side is the 30 selected stocks list from S&P 500, and the image in the middle is their expected return for different period

3.2.2 Find Momentum Indicators

Second, two momentum indicators, the Relative Strength Index (RSI) and Stochastic Oscillator (STO), are calculated based on the prices data. The results RSI and Stochastic Indicators for each stock were saved in the dataframes *RSI* and *STO*, respectively.

The *RSI* dataframe is shown below:

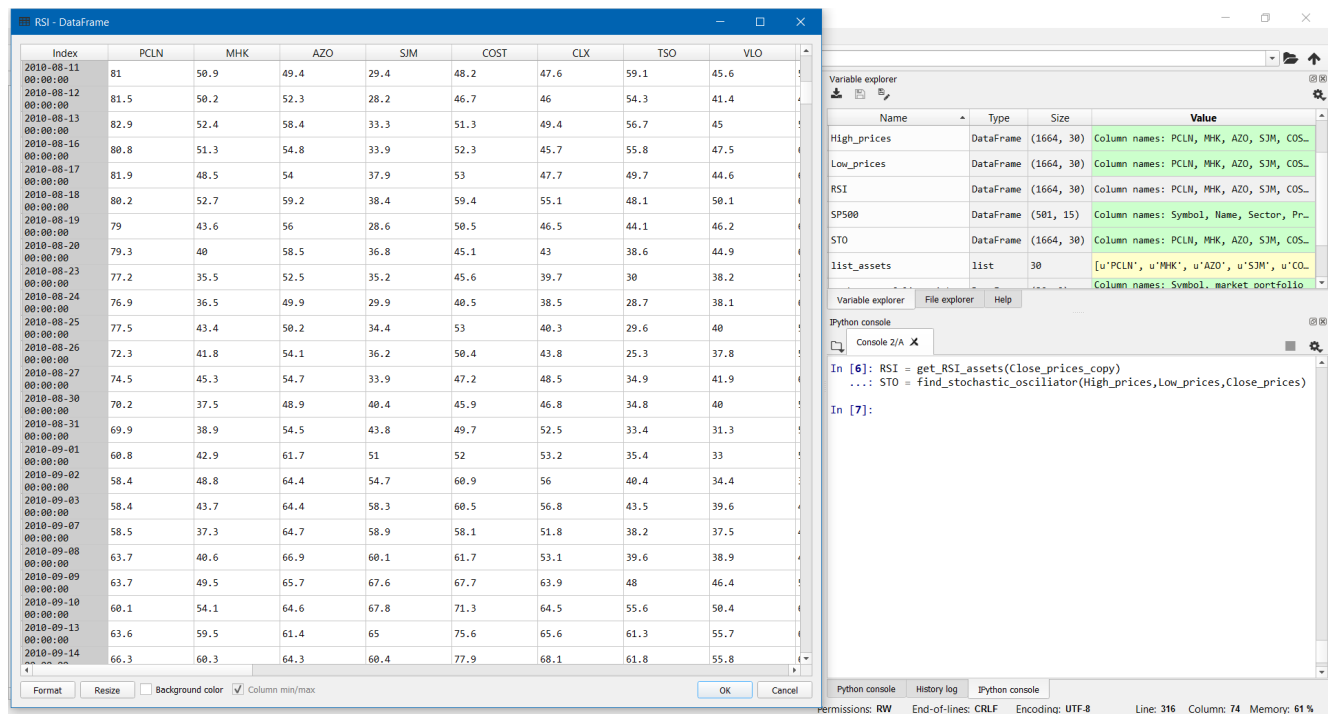


Figure 5: RSI numbers calculated for the selected 30 stocks from S&P 500

Here is the STO dataframe:

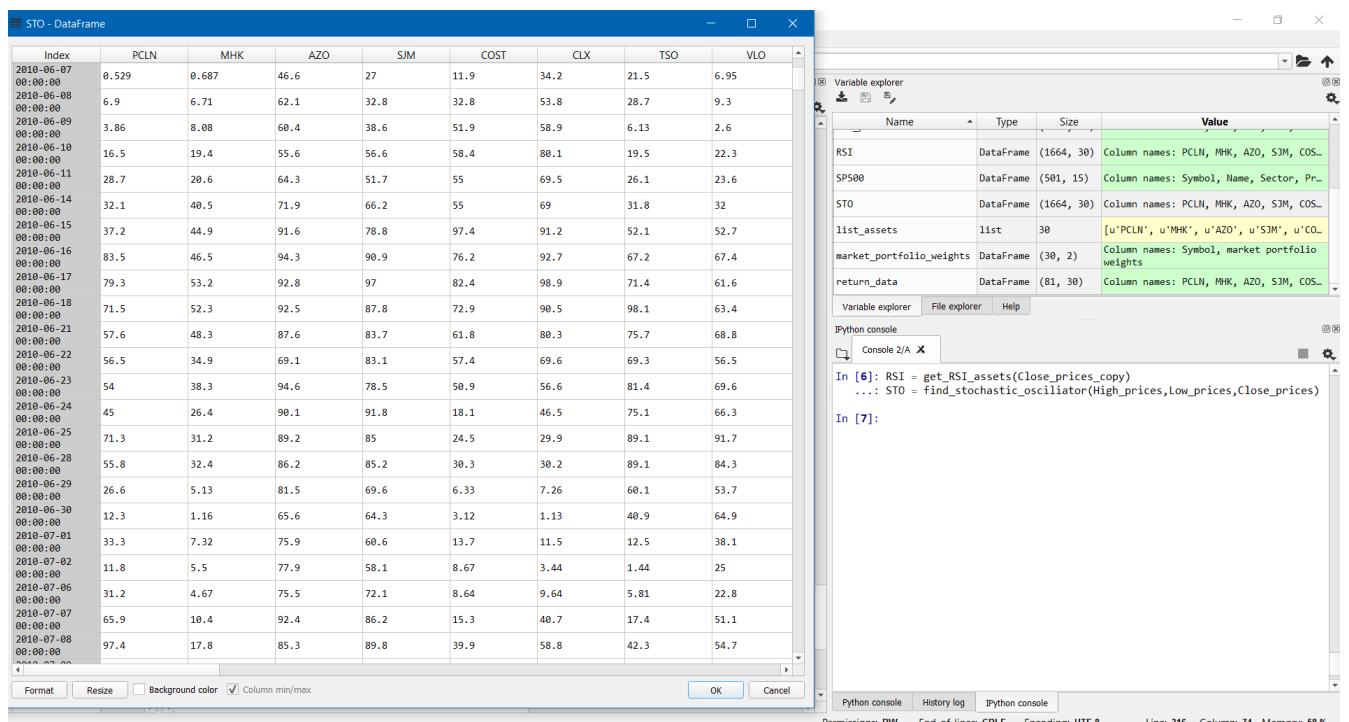


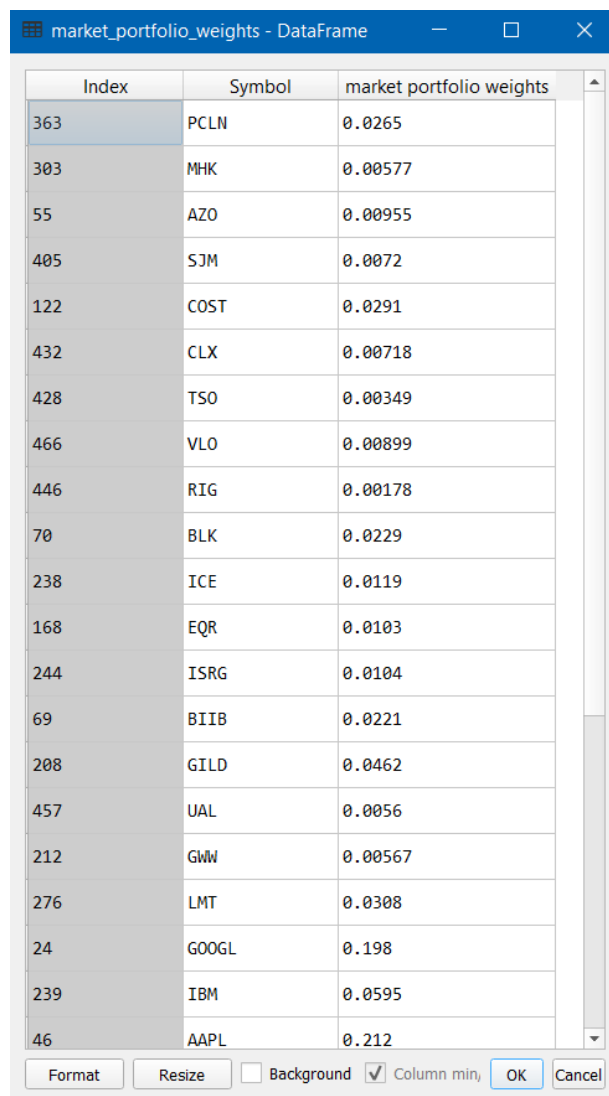
Figure 6: K% calculated for the selected 30 stocks from S&P 500

3.3 Get Market Portfolio Weights

After these two processes are completed, the market cap data is retrieved from the *SP500* table, the market weight for each of the 30 stocks are calculated based on the following calculation:

$$w_i = \frac{\text{market cap of stock } i}{\text{total market cap of 30 stocks}}$$

By calculating the market equilibrium weights based on market cap, we formed an equilibrium market portfolio weights that closely resembled the real life situation (since most indexes in the financial markets are market cap weighted). This market portfolio weight is then used as a parameter in the Black-Litterman model. Sample result can be seen in Figure



Index	Symbol	market portfolio weights
363	PCLN	0.0265
303	MHK	0.00577
55	AZO	0.00955
405	SJM	0.0072
122	COST	0.0291
432	CLX	0.00718
428	TSO	0.00349
466	VLO	0.00899
446	RIG	0.00178
70	BLK	0.0229
238	ICE	0.0119
168	EQR	0.0103
244	ISRG	0.0104
69	BIIB	0.0221
208	GILD	0.0462
457	UAL	0.0056
212	GW	0.00567
276	LMT	0.0308
24	GOOGL	0.198
239	IBM	0.0595
46	AAPL	0.212

Figure 7: Market Portfolio Weight

3.4.1 Use Momentum Indicators to generate views for the Quick Create Model

The calculated Relative Strength Index and Stochastic Oscillator were passed into a function that identifies the “most likely to rise” stock (smallest Relative Strength Index or smallest Stochastic Indicator) and the “most likely to drop” stock (largest Relative Strength Index or largest Stochastic Indicator). If the largest Relative Strength Index (or largest Stochastic Indicator) is greater than 80 and the smallest Relative Strength Index (smallest Stochastic Indicator) is less than 20, we set the views as:

- “according to RSI, the most likely to rise stock will outperform the most likely to drop stock by 2%”

Similarly, if the largest Stochastic Oscillator (or largest Stochastic Indicator) is greater than 80 and the smallest Stochastic Oscillator (smallest Stochastic Indicator) is less than 20, we set the views as:

- “according to STO, the most likely to rise stock will outperform the most likely to drop stock by 1%”

The views are then put into a list form, which matches the *update_views* functions.

```
In [12]: RSI_views
Out[12]: [[u'UAL', u'ICE'], [-1, 1], 0.03]

In [13]: STO_views
Out[13]: [[u'UAL', u'GILD'], [-1, 1], 0.01]

In [14]: |
```

Figure 6: Views format for the Black-Litterman

The first item in the list are the relevant assets that are used in the inequality, i.e the most overbought and the most oversold assets. The second item in the list is the indicates the direction of the inequality, for example in this case $ICE > UAL$. The third item in the list indicates the extent of outperform, in this case $ICE > UAL$ for 3%.

3.4.2 Use *request.form.getlist()* to get user input views for Professional Model

Professional users will be given the option to input their own views by plain language. For example, if the user believe that Google stock will outperform Apple stock by 2%, just input “GOOGL > AAPL by 2%” is going to translate the view. The program will translate the plain language into the form like shown in Figure 6.

3.5 Update Views

The generated views using the momentum indicators or user inputs are fed into the *update_views* function, which matches the dimensions of the views and the optimizing equations. For example, in the case where we used 30 stocks, the optimizer need to have the correct dimensions of matrices of 30 for the solver to get the correct answer.

The P matrix is a n x 2 (n=30) matrix arranged as only the relevant asset's position get changed to -1 or 1, the rest of the values are left to be 0.

$$P = \begin{bmatrix} 0 & \dots & 0 & 0, -1, 0 & \dots & 0, 1, 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0, 1, 0 & \dots & 0, -1, 0 \end{bmatrix}$$

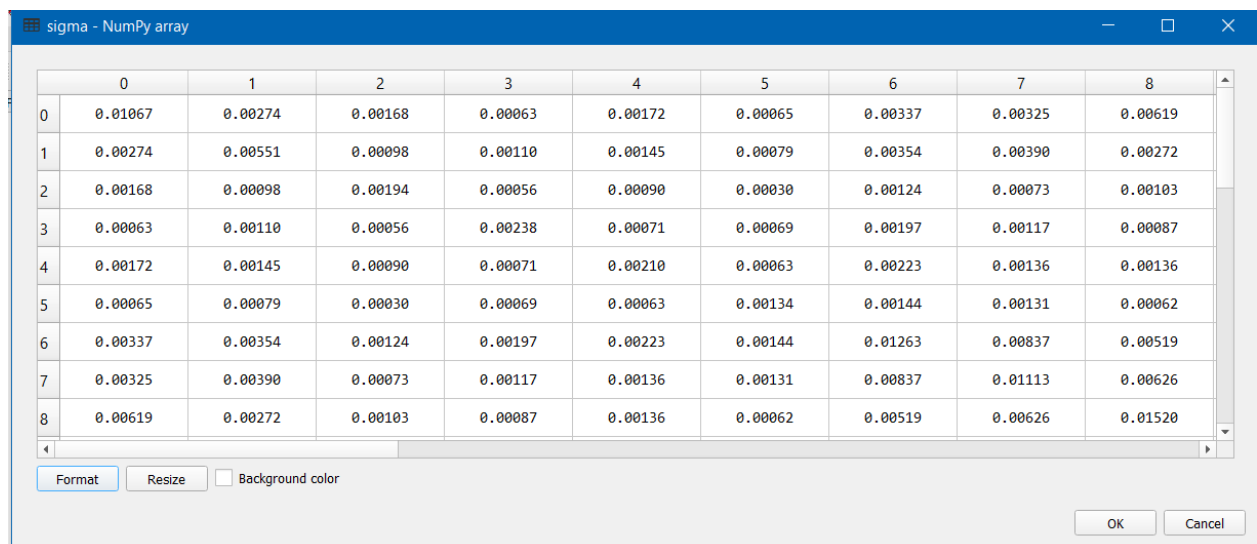
$$Q = \begin{bmatrix} 0.03 \\ 0.01 \end{bmatrix}$$

The Q matrix is a 2 x 1 matrix that contains the view values and These results are returned in a list [P,Q]

3.6 Input parameters into Black-Litterman

With the assets list, return data, and views ready, we are able to put all these parameters into the Black-Litterman Model. The standard industry practice has covariance matrices stored for auditing purposes. However, for the purpose of this project, storing the covariance matrices is not necessary. We plan to have the application to be ran on real time data. Since the result will vary from day to day the calculated results such as covariance matrix will not be stored in the backend tables. Instead, it will just be directly calculated from a stored backend table, which is a relatively simple process. Thus, the covariance matrix is calculated within the model and used in the in the black-Litterman optimization. The result weights of the optimizations are saved in a dataframe, and printed out on the frontend. Each run of the result is also saved in the database table *Weights* as one new row.

The screenshot of Covariance matrix is shown below. This screenshot only showed 7 x 7 of the 30 x 30 covariance matrix.



	0	1	2	3	4	5	6	7	8
0	0.01067	0.00274	0.00168	0.00063	0.00172	0.00065	0.00337	0.00325	0.00619
1	0.00274	0.00551	0.00098	0.00110	0.00145	0.00079	0.00354	0.00390	0.00272
2	0.00168	0.00098	0.00194	0.00056	0.00090	0.00030	0.00124	0.00073	0.00103
3	0.00063	0.00110	0.00056	0.00238	0.00071	0.00069	0.00197	0.00117	0.00087
4	0.00172	0.00145	0.00090	0.00071	0.00210	0.00063	0.00223	0.00136	0.00136
5	0.00065	0.00079	0.00030	0.00069	0.00063	0.00134	0.00144	0.00131	0.00062
6	0.00337	0.00354	0.00124	0.00197	0.00223	0.00144	0.01263	0.00837	0.00519
7	0.00325	0.00390	0.00073	0.00117	0.00136	0.00131	0.00837	0.01113	0.00626
8	0.00619	0.00272	0.00103	0.00087	0.00136	0.00062	0.00519	0.00626	0.01520

Figure 8: covariance matrix for the 30 selected stocks for the Black-Litterman

According to Thomas M. Idzorek's paper on Incorporating user specified confidence levels, our confidence in views, the standard error in investors views are calculated, and change from assets to assets.

The Returned Result is shown as below

Optimal solution found.

	Holding
PCLN	0.0006%
MHK	0.0001%
AZO	0.0002%
SJM	6.2853%
COST	0.0006%
CLX	0.2856%
TSO	0.0002%
VLO	0.0001%
RIG	4.3801%
BLK	0.0001%
ICE	24.9785%
EQR	0.0003%
ISRG	0.7202%
BIIB	0.0001%
GILD	2.2170%
UAL	0.0000%
GMW	3.1711%
LMT	0.0491%
GOOGL	16.7618%
IBM	1.5893%
AAPL	19.4857%
SHW	0.0004%
LYB	1.7408%
EMN	0.0001%
LVL	0.8608%
VZ	7.0646%
T	10.4064%
SRE	0.0002%
AEP	0.0001%

Figure 9: resulted optimized weights using the Black-Litterman

All of these results summed up to a total weight of 99.9999%, which is really close to a 100%.

The return data and the result weights are saved in the database.

Here is the complete backend flow chart

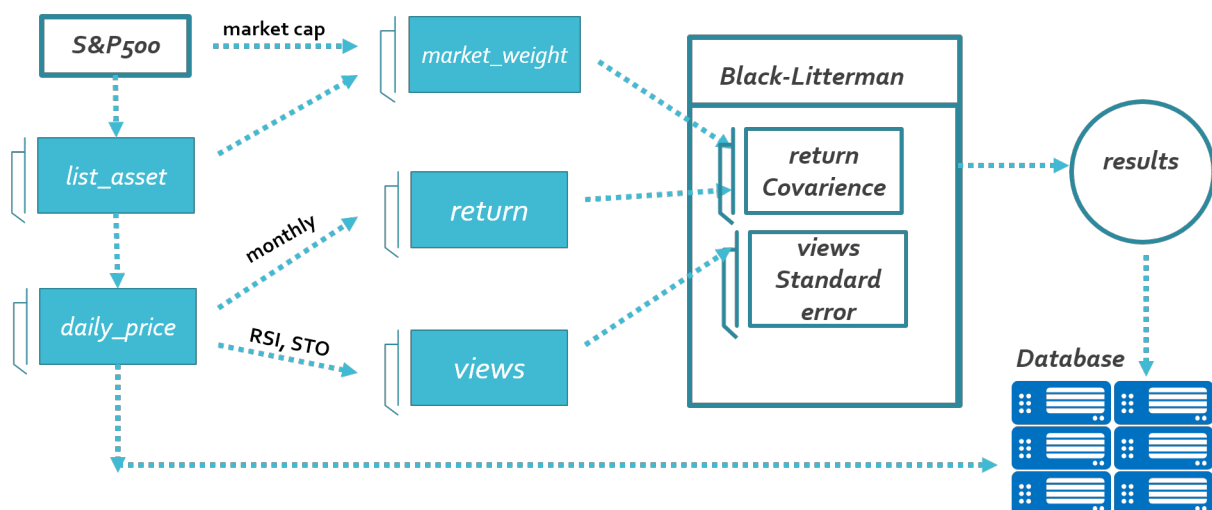


Figure 10: back-end flow chart

3.7 Package used

Numpy	The library for mathematical calculations
Scipy	The scientific library for mathematical calculations, linear algebra in particular
datapackage	The library we used to gather S&P 500 data
cvxpy	The optimizer we used in the model
Pandas	The dataframe library for data analysis
Matplotlib	The graphing plugin for us to generate pie chart
flask	The flask framework which contains the model, view and controllers for the app structure
StringIO	Convert the graph generated by Matplotlib into a PNG format, which could be displayed on the frontend

4. Front End

The front end of the application connects the business logic and data stored in the back end to the user. In making design decisions for the front end, we prioritized scalability and usability so that the front end could adapt to changes in the back end easily, and users would find interacting with the application to be enjoyable and intuitive.

4.1 Technology choices

A Flask RESTful API controlled our application routes and argument passing. We had previously considered AngularJS as our front end framework but choosing full stack Flask was simpler to integrate into the backend.

Jinja was used as the template engine to create dynamic html. For example, the recommended portfolio weights were turned into an html table and then passed as an argument to a function that would insert the html snippet into a dynamic block of the results page. In addition, Jinja made it convenient to keep a consistent layout between all pages. We defined a base page template that included the navigation bar and scripts. The contents of the pages were in separate html files and only had to include the base page template's path.

We recognize that users may want to access the application on various devices. Bootstrap was used to create pages that were responsive to the size of the window. Their pre-defined in grid elements made it simple implement dynamic UI elements.

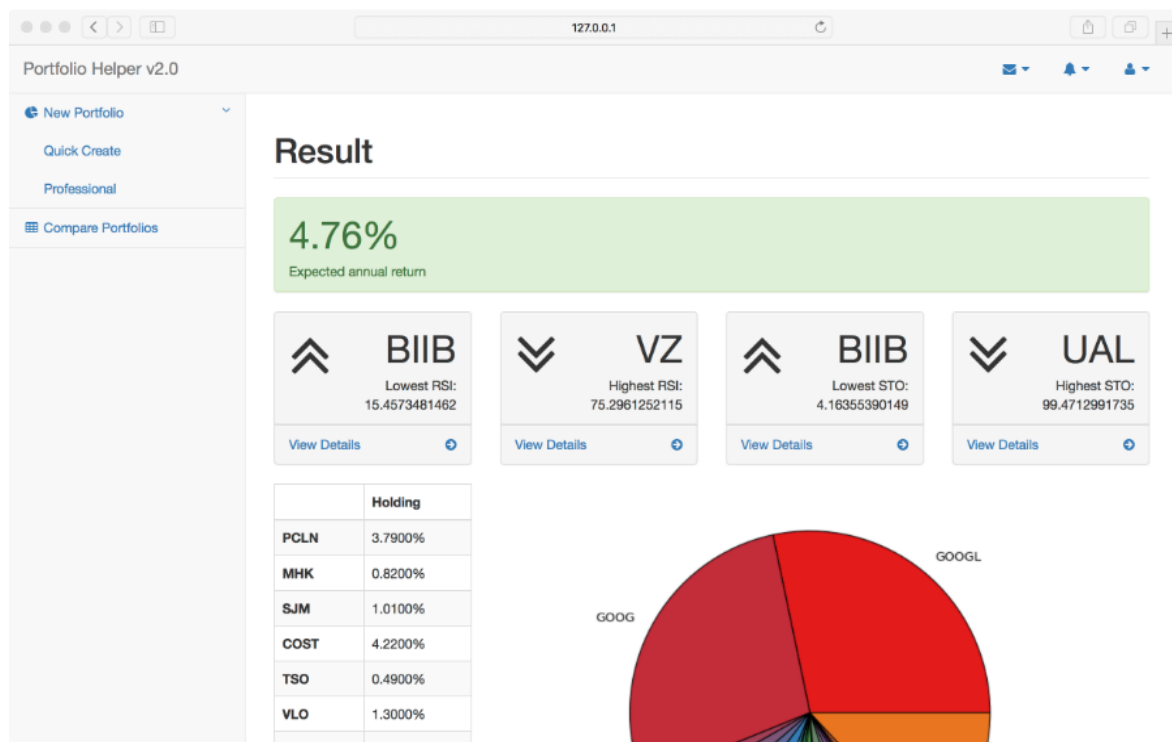


Figure 11: Website Overall look

We used the DataTables package to create responsive charts. Our main concern here was how a user might navigate a large table because the list of assets might be large as they can both use and view the results of an optimization that uses up to all 500 assets in the S&P500. DataTables provided a way to create tables with scrolling, which we implemented in our Compare Portfolios page. In addition we also liked the option to sort the contents of columns and embed buttons into the table cells, although we did not implement those.

Pie charts were generated using the matplotlib package since the data handled in business logic relied on Pandas dataframes. Working with Pandas dataframes and matplotlib was relatively straightforward to implement and provided many chart customization options.

4.2 Landing Page and Navigation User Interface

Users encounter a landing page when they first open the application.

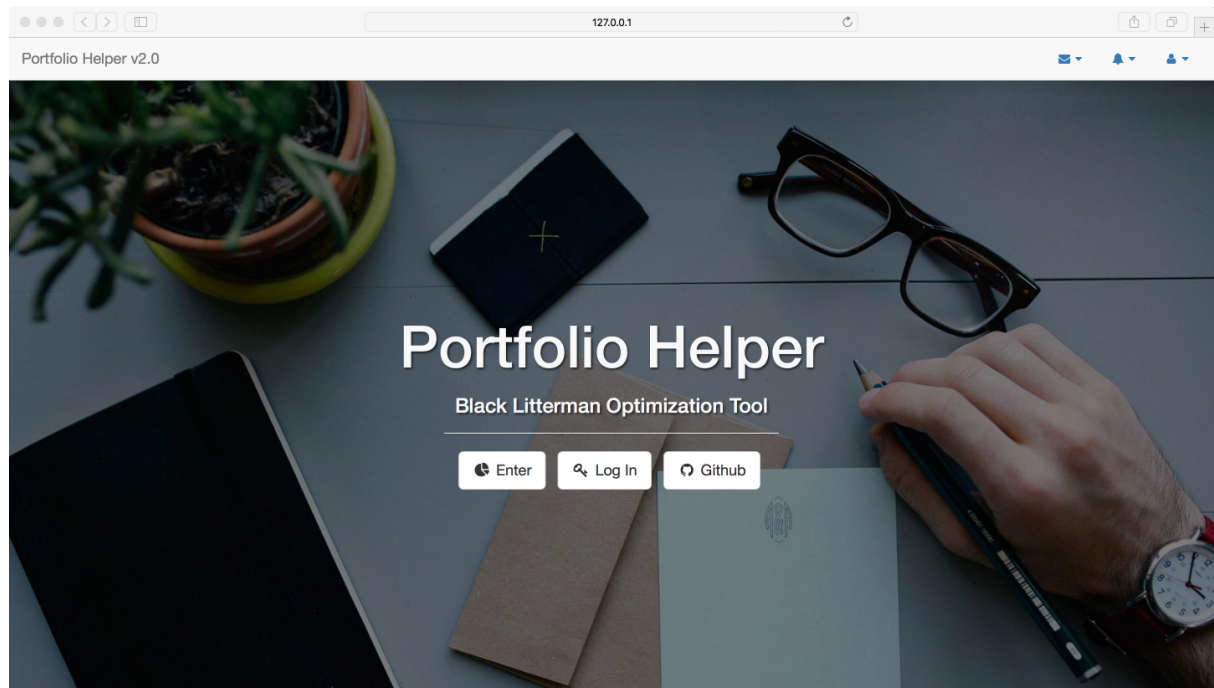


Figure 11: landing page

After logging in, users are redirected to the default Quick Create page. Since if we were to make our application live, most our users will be non-experienced investors or relatively new users who would want to try out the application quickly, the Quick Create page is the most suitable page to show.

To the left of the application is a vertical navigation panel.

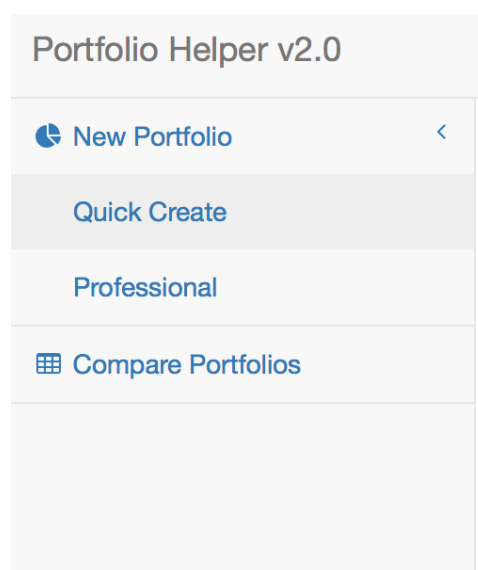


Figure 11: Vertical navigation

4.3 Non-Professional User Front-End Interface

The Quick Create option is designed for a less experienced user. The only input the user has to provide is the number of stocks from each industry sector represented in the S&P500 they would like to include in the optimization. By minimizing the number of technical jargon and options that a less experienced user might be confused by and cause them to provide bad inputs, the portfolio recommendations generated will be more reliable.

Portfolio Helper v2.0

New Portfolio

Quick Create

Professional

Compare Portfolios

Run Portfolio Optimization

The Quick Create option was created with the goal of giving non-institutional users access to similar optimization models as professionals. Select the number of stocks from each sector of the S&P500 you would like to include in the optimization and our algorithm will determine the portfolio weights based on the assets' historical returns and trends.

Number of Assets from Each Sector

1
2
3
4

Optimize

Figure 11: Non-Professional Users input interface

After the optimize button is clicked and the Black-Litterman optimization is done, the user is redirected to the Results page. Expected annual return is displayed in a green highlighted box to signify the importance of the value. Also the RSI and Stochastic values that were generated and used in the views are displayed in addition to the recommended stock holdings, as this information may be of interest, but are in grey boxes as they are not essential information.

Holdings are displayed as percentages and a pie chart is presented to the right to help users visually obtain an idea of the distribution of weights.

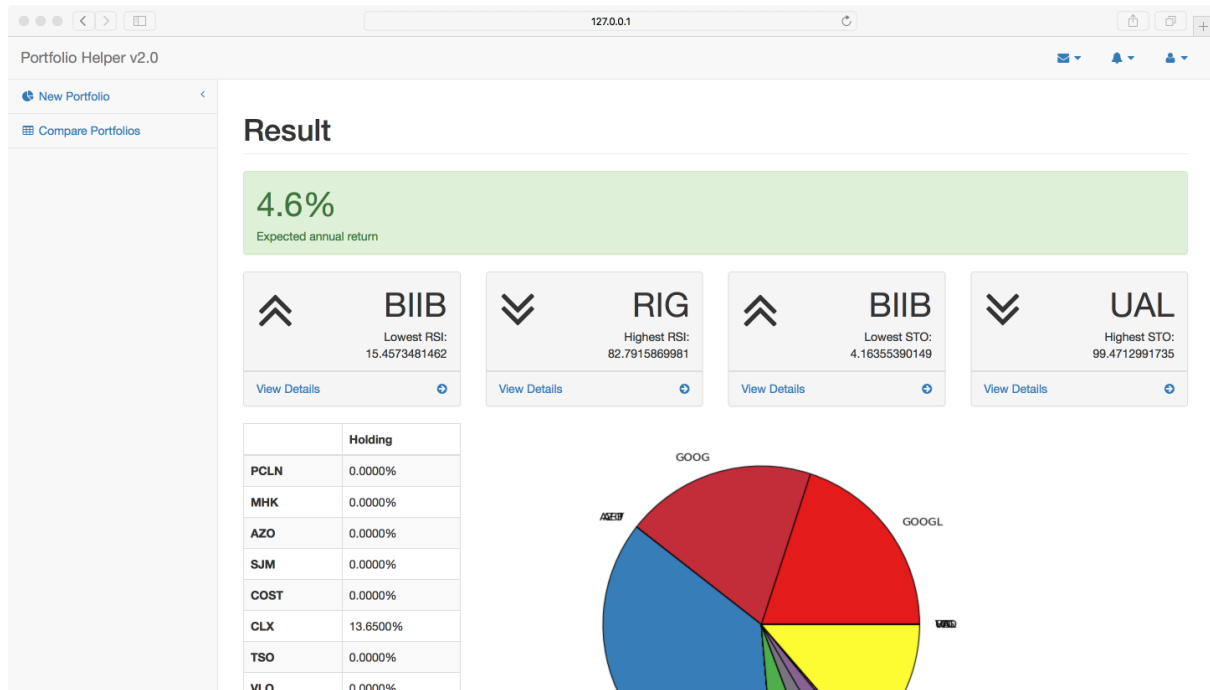


Figure 12: results of quick create displayed

4.3 Professional User Front-End Interface

Professional users have a different interface since they have more experience with finance modeling. The Professional option lets users customize both the stocks in their portfolio as well as specify relative views. Stocks are selected by typing in their tickers separated by commas into a text field. Relative views are entered also with a text field, by typing in the format Asset1 > Asset2 by X%.

Portfolio Helper v2.0

New Portfolio

Quick Create

Professional

Compare Portfolios

Run Customized Portfolio Optimization

Custom portfolios are designed for a professional user who desires a more flexible model. Enter both the assets to be included and related views.

Asset Tickers

GOOGL, AAPL, MSFT

Relative View 1

GOOGL > MSFT by 2%

Relative View 2

MSFT < AAPL by 3%

Optimize

Figure 13: Stocks and views can be chosen on the Professional customized portfolio optimization page

The results page is similar to the nonprofessional Quick Create option, but without the technical indicator information since we the professional investor generated views with an external method.

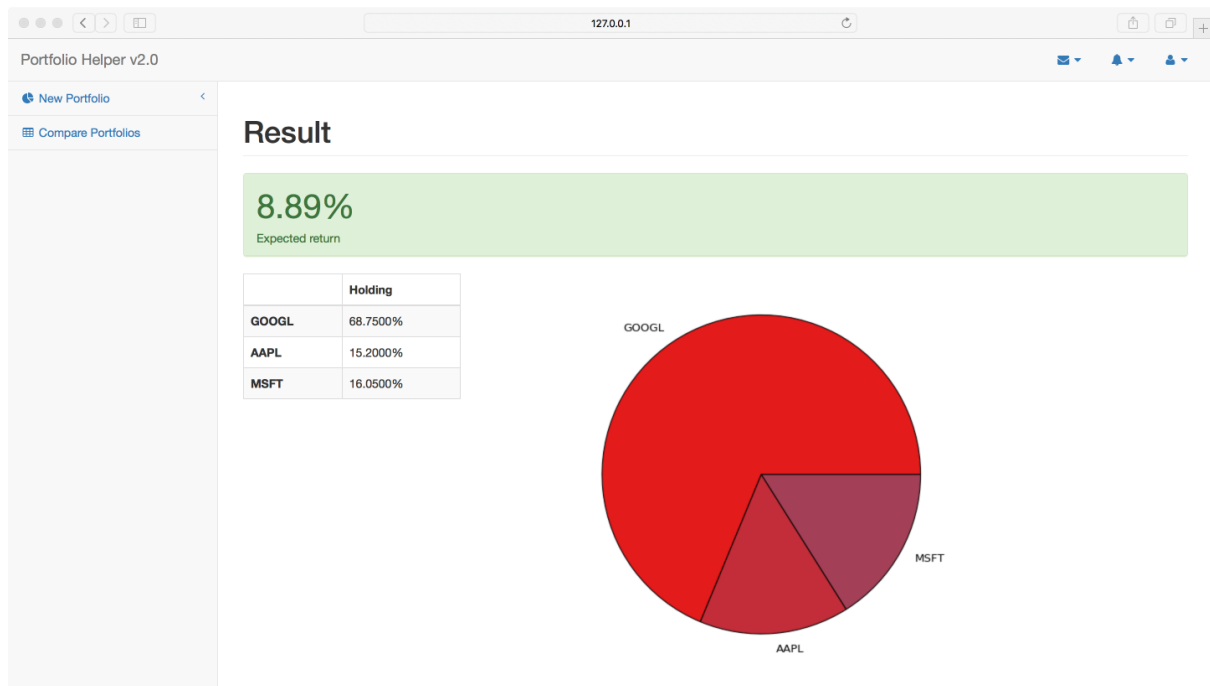
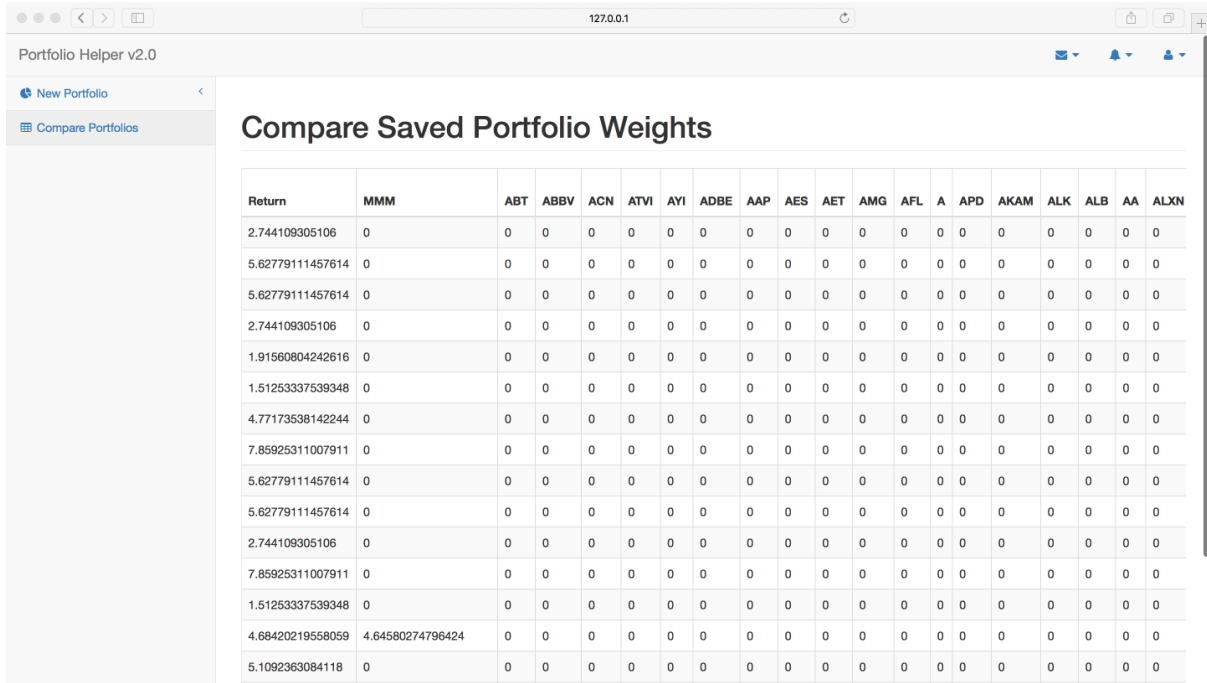


Figure 14: Results displayed for the customized portfolio

4.4 Professional User Front-End Interface

Compare Portfolios allows you to view recently generated portfolios and compare their allocations and returns.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1'. The application title is 'Portfolio Helper v2.0'. On the left, there is a sidebar with two menu items: 'New Portfolio' and 'Compare Portfolios', with the latter being selected. The main content area is titled 'Compare Saved Portfolio Weights' and contains a table with 18 columns and 18 rows of data.

Return	MMM	ABT	ABBV	ACN	ATVI	AYI	ADBE	AAP	AES	AET	AMG	AFL	A	APD	AKAM	ALK	ALB	AA	ALXN
2.744109305106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.62779111457614	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.62779111457614	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.744109305106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.91560804242616	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.51253337539348	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.77173538142244	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7.85925311007911	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.62779111457614	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.62779111457614	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2.744109305106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7.85925311007911	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.51253337539348	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4.68420219558059	4.64580274796424	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.1092363084118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15: Saved results from the previously generated portfolios

4.5 Sample results

4.5.1 Quick create

Inputs: assets per sector: 1

Outputs: expected annual return: 7.85%

ISRG lowest RSI of 41.3123343469

UAL highest RSI of 0.0736329844

ISRG lowest STO of 6.71155854782

UAL highest STO of 99.4712991735

	Holding
PCLN	8.7900%
SJM	2.3400%
TSO	1.1300%
BLK	7.5900%
ISRG	3.3900%
UAL	1.8000%
GOOGL	65.0400%
SHW	3.7000%
LVL	2.4600%
SRE	3.7600%

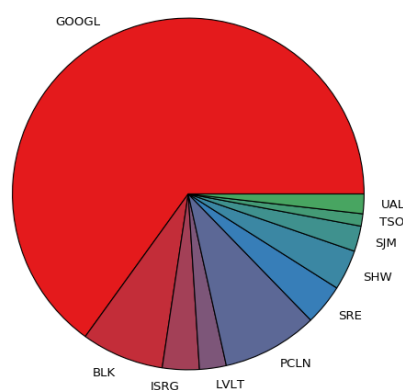


Figure 16: results for the Quick Create example 1

Input: assets per sector: 5

Outputs: expected return : 17.56%

BIIB lowest RSI of 15.4573481462
TRV highest RSI of 84.3922928177
MDLZ lowest STO of 2.54240591783
TRV highest RSI of 99.7019300886

	Holding
BIIB	22.6500%
GOOGL	37.4900%
DOW	39.8600%

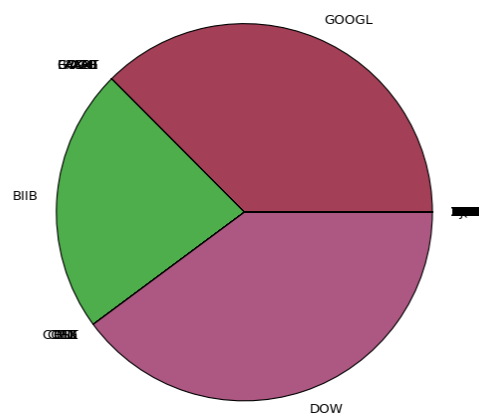


Figure 16: results for the Quick Create example 2

4.5.2 Professional

Input: asset tickers: GOOGL, MSFT, AAPL
 relative view 1: GOOGL < MSFT by 2%
 relative view 2: MSFT < AAPL by 1%

Output: expected annual return: 8.89%

	Holding
GOOGL	68.7500%
MSFT	16.0500%
AAPL	15.2000%

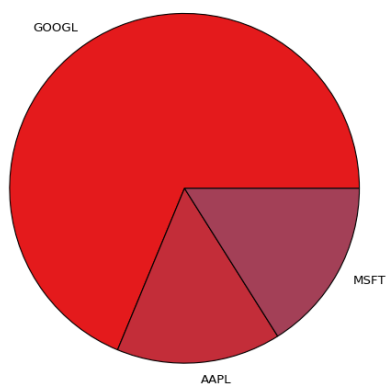
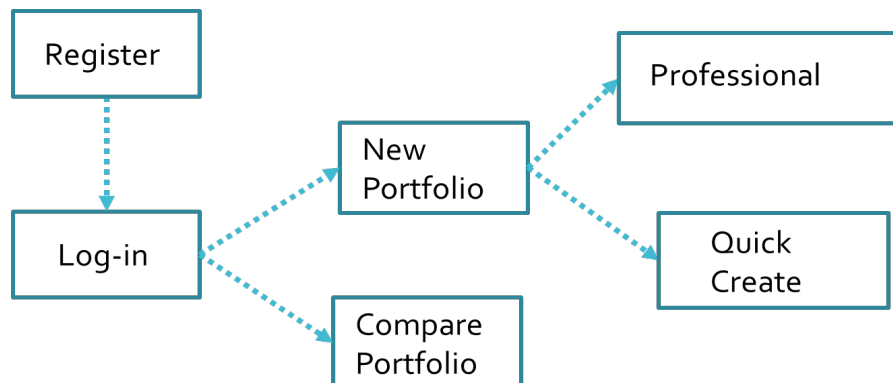


Figure 16: results for the Professional

5. Product Design Flow



The website will allow users register in the beginning, and ask them to login in to access their personal customized portfolio. Then they will get the chance to select the type of new portfolios they want to create.

There are two kinds of the portfolios they can create designed by the targeting groups: professional and non-professional users. Both of them will use Black-Litterman as their main tool to optimize their portfolio. Professional users will be able to input their selected stocks and views, and generate the portfolio weights based on the views and market equilibrium expected return. Non-professional users will get the customized results based on our recommendation on the stocks and momentum indicators implemented in the Black-Litterman preferred views.

In details, if users go to the “Professional” page, they can:

- Select stocks from S&P500
- Input their preferred views
- The market equilibrium portfolio is calculated from market capitalizations of portfolio components
- Covariance matrix will be calculated every time based on the different combination of the assets

If users go to the “Quick Create”, they customized portfolio will be based on:

- Stocks are selected from the list we created from S&P 500. The list is generated based on the top earning per share performances in different sectors
- Views are generated using Relevant Asset Index and Stochastic Oscillator
- The market equilibrium portfolio is calculated from market capitalizations of portfolio components
- Covariance matrix will be calculated in python every time based on the different combination of the assets

In addition, users can go to “Compare Portfolio” to view their previous saved portfolios to make comparisons and review performance.

6. Evaluation

6.1 Strength

We chose a modified approach to Black-Litterman for non-professional users that stands out from the Black-Litterman optimizers existing in products already on the market. Our product uses the Black Litterman Model with the momentum indicators RSI and Stochastic Oscillators to generate views. By choosing the extreme values (20, 80) for the indicators, we avoid errors in predicting the stock’s directions, and make views that are relatively conservative and probable. By using technical indicators as views generators without the need for user input on the views, the final design enables to provide a quite unique portfolio management tool for non-professional investors.

Our product allows to select a large combination of assets from the S&P500 as well as providing some ways to make personal choices. Customizability is allowed for both non-professional and professional users, although to different extents. Professional users might find this feature especially attractive as they may have proprietary view generation systems that they may like to test. Our application provides a simple platform for them to try multiple portfolio setups with minimal interactions to code.

The application is in the form of a web application and easily accessible to everyone. No installation of programs or libraries is required to perform an optimization. The application has also been designed to be functional on windows of all sizes, broadening the use to include both desktop and mobile users.

6.2 Weakness and Improvements

Transaction cost have not been implemented in the model. However, by not including transaction costs we are ignoring an important step in how one would actually build their portfolio. We initially attempted to incorporate the transaction cost in the Black-Litterman model, but doing so have made the optimizations non-convex and the solver was unable to deliver the weights. If possible in the future, we would like to figure out a way to include the transaction cost to make the system more realistic.

We removed MVO as a model in our application in the middle stage as it provided biased portfolios, so the Black-litterman became the only model we have considered in the support system. Our model has all the disadvantages of the Black-Litterman and provides no alternatives. It does not give the best possible portfolio, merely the best portfolio given the views stated. In addition, the Black-Litterman is sensitive to its assumption. It assumes that views are independent of each other, which

may not be true in reality. Further models can be cooperated to improve the result, like VaR and CVaR.

Slow loading times were a problem with our design. If we were to implement more complicated technical indicators in our views generation, the problem would be exacerbated. This could be improved by implementing some sort of caching to minimize the number of requests the client would have to make to the server to get historical returns, technical indicator values, price, and other data.

7. Conclusion

Our asset allocation decision support system is a simple and professional system that enables users of different experience levels to create a portfolio using the Black-Litterman model. The entire system is user friendly in terms of entering data and displaying results, as well as smart in choosing to display only the information that each user group would find most helpful. We believe that financial decision support systems will be in huge demand in the near future as users grow to include more professional users and also branch out into the non-professional investment community. With improvements, a product such as ours that considers the needs of each user group fills a highly demanded gap in the current market and could perform well.

Appendix A: Proposal Feedback from Minh

Business Logic

- How do you guys estimate the expected return/covariance matrix? Do you guys use sample mean? sample covariance? why that specific method?
- How do you guys incorporate transaction cost? In the constraint? objective function?
- Any other constraints that the users might be interested in? Like no shorting constraints, lower/upper bounds on the weights, etc.
- Black-Litterman - What is w ? It is very important to understand the model you guys are using, and to clearly show that you understand it. How does MVO relate to Black-Litterman?
- Is VaR convex? What about CVaR?
- Any other alternative formulation of MVO? How do you guys incorporate the risk aversion coefficient?
- How does the investment horizon affect your estimations? How do you incorporate it into the estimation?

Front-end

- Sketch of interface would be helpful in planning for implementation
- How does the front-end code communicate with the backend?

Backend

- “Display” the portfolio graph/result should be in the front-end? What does this mean when you guys put this on the diagram?
- Do you guys store calculated data? Such as covariance matrix, etc.
- What is your plan in storing financial data?
- You need to keep track of how your portfolio grows over time. How do you address that here?

Appendix B: Works Cited

401K Plus. (2015). Portfolio Construction.

Duffie, D. a. (1997). An Overview of Value at Risk. The Journal of Derivatives 4.3.

G.Luenberger, D. (2013). Investment Science. Stanford: Oxford University Press.

Idzorek, T. M. (2005). A STEP-BY-STEP GUIDE TO THE BLACK-LITTERMAN MODEL. Chicago.

Improving the Win-Loss Ratio with the Relative Strength Index by Thomas Bulkowski. (1998). In Stocks & Commodities.

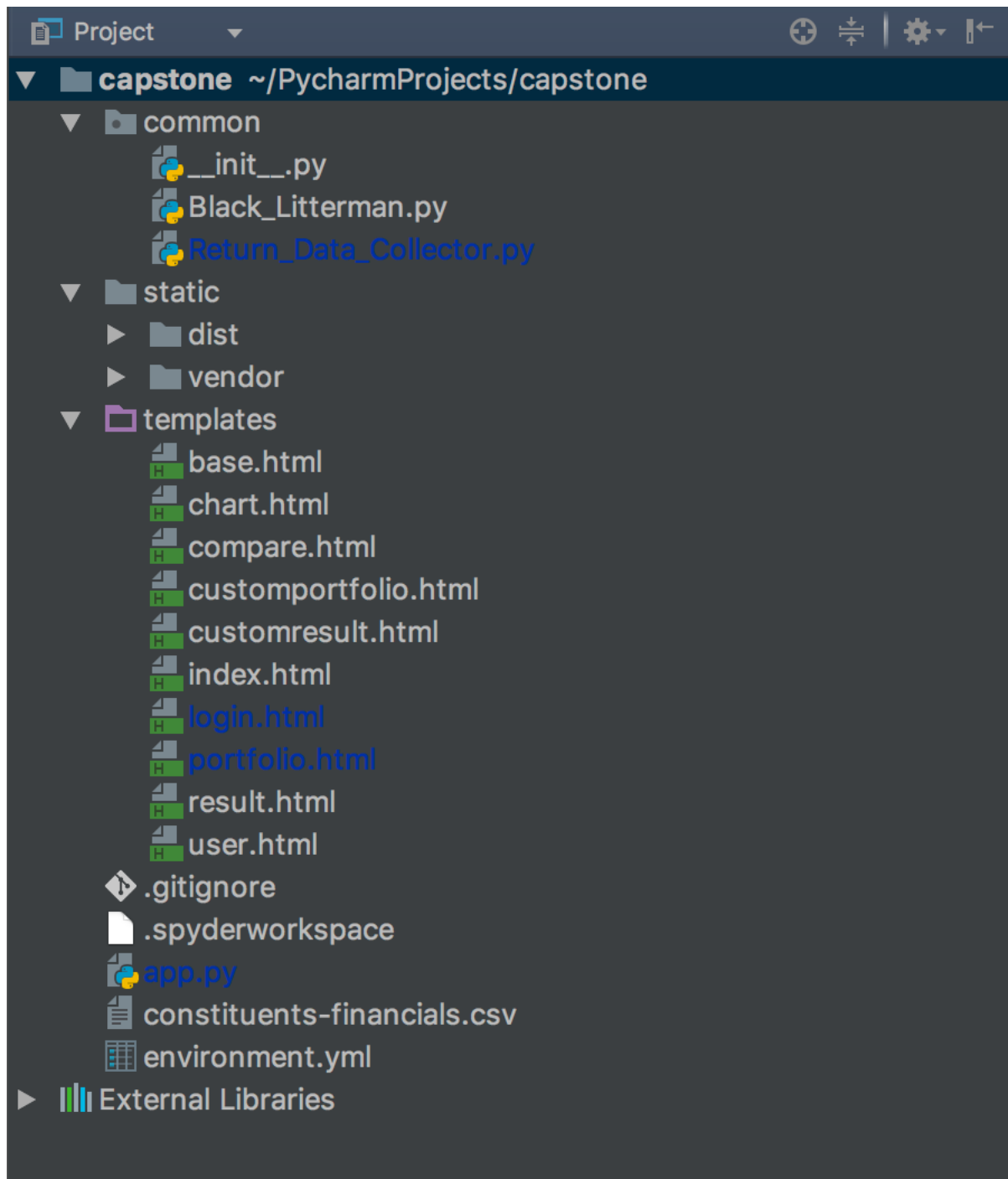
SmartFolio. (2015). Multi-period Asset Allocation. Retrieved from <http://smartfolio.fileburst.com/download/Theory%20Help.pdf>

The Relative Strength Index by Bruce Faber. (1994). In Stocks & Commodities V. 12:9 (pp. 381-384). Retrieved from StockCharts.

The Stochastic Oscillator by Joe Luisi. (1997). In Stocks & Commodities.

Walters, J. (2014). The Black-Litterman Model In Detail.

Appendix C: File Structure



Appendix D: Back end

app.py

```
from flask import Flask, make_response, session
from flask_sqlalchemy import SQLAlchemy
from flask import request, redirect, url_for, render_template
from common.Return_Data_Collector import get_asset_return_data,
get_SP500, get_market_portfolio_weights,
get_market_portfolio_weights_customized, get_price_changes_data
from common.Black_Litterman import
Black_Litterman, update_views, combine_momentum_oscillator_views,
get_RSI_assets, update_relevant_assets_RSI,
find_stochastic_osciliator, update_relevant_assets_Stochastic
import pandas as pd
import numpy as np
from matplotlib.backends.backend_agg import FigureCanvasAgg as
FigureCanvas
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import StringIO

from flask_security import Security, SQLAlchemyUserDatastore,
UserMixin, RoleMixin, login_required

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql://username:password@hostname/database_name'
app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql://zbkogjiiodhmxbob:3LpsDLCEaBHv1b_cu99otyPdY6@ec2-54-235-
119-29.compute-1.amazonaws.com:5432/ddv0hgedai3tgo'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True

app.config['SECRET_KEY'] = 'super-secret'
app.config['SECURITY_REGISTERABLE'] = True

app.debug = True
db = SQLAlchemy(app)

"""
Models
"""
roles_users = db.Table('roles_users',
                        db.Column('user_id', db.Integer(),
db.ForeignKey('user.id')),
                        db.Column('role_id', db.Integer(),
db.ForeignKey('role.id'))))

class Role(db.Model, RoleMixin):
    id = db.Column(db.Integer(), primary_key=True)
```

```

name = db.Column(db.String(80), unique=True)
description = db.Column(db.String(255))

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(255), unique=True)
    password = db.Column(db.String(255))
    active = db.Column(db.Boolean())
    confirmed_at = db.Column(db.DateTime())
    roles = db.relationship('Role', secondary=roles_users,
backref=db.backref('users', lazy='dynamic'))

#Setup Flask-Security
user_datastore = SQLAlchemyUserDatastore(db, User, Role)
security = Security(app, user_datastore)

"""
Routes
"""

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login')
def login():
    return render_template('security/login.html')

@app.route('/portfolio')
@login_required
def portfolio():
    return render_template('portfolio.html', name='Run Portfolio
Optimization')

@app.route('/customportfolio')
@login_required
def customportfolio():
    return render_template('customportfolio.html', name='Run
Customized Portfolio Optimization')

@app.route('/portfolio/get_optimal_portfolio_black_litterman',
methods=['GET', 'POST'])
# call Black_litterman function and save results
# this is the function for the quick create users
def get_optimal_portfolio_black_litterman():
    number_assets = request.args.get("number")

```

```

print number_assets

SP500 = get_SP500()
market_portfolio_weights = get_market_portfolio_weights(SP500,
int(number_assets))
list_assets = list(market_portfolio_weights['Symbol'])
return_data = get_asset_return_data(list_assets)['df_return']
return_data.to_sql("returns", db.get_engine(app),
if_exists='replace')
SP500.to_sql("SP500", db.get_engine(app), if_exists='replace')
market_weights = np.array(market_portfolio_weights['market
portfolio weights'])

Close_prices = get_price_changes_data(list_assets)
Close_prices_copy = Close_prices.copy(deep=True)
High_prices = get_price_changes_data(list_assets,
price_type='High')
Low_prices = get_price_changes_data(list_assets,
price_type='Low')
RSI = get_RSI_assets(Close_prices_copy)
STO = find_stochastic_oscillator(High_prices, Low_prices,
Close_prices)
alpha = 2.5

RSI_views = update_relevant_assets_RSI(RSI)
STO_views = update_relevant_assets_Stochastic(STO)

RSI_big = RSI_views[3]
RSI_small = RSI_views[4]
STO_big = STO_views[3]
STO_small = STO_views[4]

RSI_views = RSI_views[:3]
STO_views = STO_views[:3]

relevant_assets = combine_momentum_oscillator_views(RSI_views,
STO_views)[0]
P_views_values = combine_momentum_oscillator_views(RSI_views,
STO_views)[1]
Q_views_values = combine_momentum_oscillator_views(RSI_views,
STO_views)[2]

Views_Matrices = update_views(list_assets, relevant_assets,
P_views_values, Q_views_values)
P = Views_Matrices[0]
Q = Views_Matrices[1]
weights, Return = Black_Litterman(return_data, alpha, P, Q,
market_weights)
Return = ((1 + Return) ** 12 - 1) * 100
Return_dict = [{'Return':Return}]
Return_fr = pd.DataFrame(Return_dict)
Return_fr = Return_fr.transpose()
Return_fr = Return_fr.rename(columns={0: 'Holding'})
weights = weights.append(Return_fr)

```

```

        weights.to_sql("Optimal Weight", db.get_engine(app),
if_exists='replace')

        weights = weights[weights.Holding > 0.00001]
        weights = weights.round(2)

        weightsport = weights[: -1].to_html(classes = 'table table-
striped table-bordered table-hover id="portfolio')

        session['data'] = weightsport
        session['Return'] = Return.round(2)
        session['rsi_small'] = RSI_small
        session['rsi_big'] = RSI_big
        session['sto_small'] = STO_small
        session['sto_big'] = STO_big

        #saving
        weights = pd.read_sql_table("Optimal Weight",
db.get_engine(app))

        weights = weights.transpose()
        weights.columns = weights.iloc[0]

        weights[1:].to_sql("Weights", db.get_engine(app), index=False,
if_exists='append')

        return redirect('/result')

@app.route('/customportfolio/get_optimal_customportfolio_black_litte
rman', methods=['GET', 'POST'])
# This is the function for professional users
def get_optimal_customportfolio_black_litterman():
    list_assets = request.form.getlist("assets")
    list_assets = ''.join(list_assets)
    list_assets = list_assets.replace(" ", "")
    list_assets = list_assets.split(",")
    print list_assets

    view1 = request.form.getlist("view1")
    view2 = request.form.getlist("view2")

    view1 = ''.join(view1)
    view1 = view1.replace("%", "")
    view1 = view1.replace("by", "")
    view1 = view1.split(" ")
    if "" in view1: view1.remove("")

    view2 = ''.join(view2)
    view2 = view2.replace("%", "")
    view2 = view2.replace("by", "")
    view2 = view2.split(" ")
    if "" in view2: view2.remove("")

```

```

print view1
print view2

SP500 = get_SP500()
market_portfolio_weights =
get_market_portfolio_weights_customized(SP500, list_assets)
print market_portfolio_weights
'''list_assets = list(market_portfolio_weights['Symbol'])'''
return_data = get_asset_return_data(list_assets)['df_return']
market_weights = np.array(market_portfolio_weights['market
portfolio weights'])

alpha = 2.5

#list asset, first two are only needed for relative view
relevant_assets = [[view1[0], view1[2]], [view2[0], view2[2]]]

P_views_values = [[1, -1], [1, -1]]
Q_views_values = [float(str(view1[3]))/100.0,
float(str(view2[3]))/100.0]

Views_Matrices = update_views(list_assets, relevant_assets,
P_views_values, Q_views_values)
P = Views_Matrices[0]
Q = Views_Matrices[1]
weights, Return = Black_Litterman(return_data, alpha, P, Q,
market_weights)
Return = ((1 + Return) ** 12 - 1) * 100
Return_dict = [{'Return':Return}]
Return_fr = pd.DataFrame(Return_dict)
Return_fr = Return_fr.transpose()
Return_fr = Return_fr.rename(columns={0: 'Holding'})
weights = weights.append(Return_fr)

weights.to_sql("Optimal Weight", db.get_engine(app),
if_exists='replace')

weights = weights[weights.Holding > 0.00001]
weights = weights.round(2)

weightscust = weights[:-1].to_html(classes = 'table table-
striped table-bordered table-hover id="portfolio')

session['data'] = weightscust
session['Return'] = Return.round(2)

weights = pd.read_sql_table("Optimal Weight",
db.get_engine(app))
weights = weights.transpose()
weights.columns = weights.iloc[0]

weights[1:].to_sql("Weights", db.get_engine(app), index=False,
if_exists='append')

```

```

        return redirect('/customresult')
    '''return render_template('customportfolio.html', name="Custom
Optimal Portfolio", data=weightscust)'''

```

```

@app.route('/chart')
def chart():
    return render_template('chart.html', name='chart')

```

```

@app.route('/plot.png')
def plot():
    weights = pd.read_sql_table("Weights", db.get_engine(app))
    weight = weights.ix[:, weights.columns != 'Return'].tail(1)

    weight = weight.loc[:, (weight > 0.001).any(axis=0)]

    labels = list(weight.columns.values)
    values = weight.values.tolist()

    cs = cm.Set1(np.arange(40) / 40.)

    fig = plt.figure()
    fig.patch.set_facecolor('white')
    plt.pie(values[0], labels=labels, colors=cs)
    plt.axis('equal')

    canvas = FigureCanvas(fig)
    output = StringIO.StringIO()
    canvas.print_png(output)
    response = make_response(output.getvalue())
    response.mimetype = 'image/png'
    return response

```

```

@app.route('/compare')
def compare():
    weights = pd.read_sql_table("Weights", db.get_engine(app))
    weights = weights.fillna(0)
    weights = weights.round(2)

    weightsmod = weights.to_html(classes = 'table table-striped
table-bordered table-hover id="example')

    return render_template('compare.html', name='Compare Saved
Portfolio Weights', data=weightsmod)

```

```

@app.route('/result')
def result():
    data = session.get('data', None)
    Return = session.get('Return', None)
    RSI_small = session.get('rsi_small', None)

```

```

RSI_big = session.get('rsi_big', None)
STO_small = session.get('sto_small', None)
STO_big = session.get('sto_big', None)

return render_template('result.html', name='Result', data=data,
upRSIstock=RSI_small[0], upRSI=RSI_small[1],
downRSIstock=RSI_big[0],
downRSI=RSI_big[1], upSTOstock=STO_small[0], upSTO=STO_small[1],
downSTOstock=STO_big[0],
downSTO=STO_big[1], Return=Return)

@app.route('/customresult')
def customresult():
    data = session.get('data', None)
    Return = session.get('Return', None)
    print Return
    return render_template('customresult.html', name='Result',
data=data, Return=Return)

@app.route('/result/save_data', methods=['GET', 'POST'])
def save_data():
    data = session.get('data', None)

    weights = pd.read_sql_table("Optimal Weight",
db.get_engine(app))
    weights = weights.transpose()
    weights.columns = weights.iloc[0]

    weights[1:].to_sql("Weights", db.get_engine(app), index=False,
if_exists='append')

    return render_template('result.html', name='Result', data=data)

```

Appendix E: Front end

base.html

```

<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>index</title>

```



```

    <!-- Bootstrap Core CSS -->
    <link href="../../static/vendor/bootstrap/css/bootstrap.min.css"
rel="stylesheet">

    <!-- MetisMenu CSS -->
    <link href="../../static/vendor/metisMenu/metisMenu.min.css"
rel="stylesheet">

    <!-- Custom CSS -->
    <link href="../../static/dist/css/sb-admin-2.css" rel="stylesheet">

    <!-- Morris Charts CSS -->
    <link href="../../static/vendor/morrisjs/morris.css"
rel="stylesheet">

    <!-- DataTables CSS -->
    <link href="../../static/vendor/datatables-
plugins/dataTables.bootstrap.css" rel="stylesheet">

    <!-- DataTables Responsive CSS -->
    <link href="../../static/vendor/datatables-
responsive/dataTables.responsive.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="../../static/vendor/font-awesome/css/font-
awesome.min.css" rel="stylesheet" type="text/css">

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></scr
ipt>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></
script>
    <![endif]-->

</head>

<body>

<div id="wrapper">

    <!-- Navigation -->
    <nav class="navbar navbar-default navbar-static-top"
role="navigation" style="margin-bottom: 0">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">

```

```

        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="/">Portfolio Helper
v2.0</a>
</div>
<!-- /.navbar-header -->

<ul class="nav navbar-top-links navbar-right">
    <li class="dropdown">
        <a class="dropdown-toggle" data-toggle="dropdown"
href="#">
            <i class="fa fa-envelope fa-fw"></i> <i
class="fa fa-caret-down"></i>
        </a>
        <ul class="dropdown-menu dropdown-messages">
            <li>
                <a href="#">
                    <div>
                        <strong>John Smith</strong>
                        <span class="pull-right text-muted">
                            <em>Yesterday</em>
                        </span>
                    </div>
                    <div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque eleifend...</div>
                </a>
            </li>
            <li class="divider"></li>
            <li>
                <a href="#">
                    <div>
                        <strong>John Smith</strong>
                        <span class="pull-right text-muted">
                            <em>Yesterday</em>
                        </span>
                    </div>
                    <div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque eleifend...</div>
                </a>
            </li>
            <li class="divider"></li>
            <li>
                <a href="#">
                    <div>
                        <strong>John Smith</strong>
                        <span class="pull-right text-muted">
                            <em>Yesterday</em>
                        </span>
                    </div>
                    <div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque eleifend...</div>
                </a>
            </li>
        </ul>
    </li>
</ul>

```

```

        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a class="text-center" href="#">
            <strong>Read All Messages</strong>
            <i class="fa fa-angle-right"></i>
        </a>
    </li>
</ul>
<!-- /.dropdown-messages -->
</li>
<!-- /.dropdown -->
<li class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown"
href="#">
        <i class="fa fa-bell fa-fw"></i> <i class="fa
fa-caret-down"></i>
    </a>
    <ul class="dropdown-menu dropdown-alerts">
        <li>
            <a href="#">
                <div>
                    <i class="fa fa-comment fa-fw"></i>
                    <span class="pull-right text-muted
small">4 minutes ago</span>
                </div>
            </a>
        </li>
        <li class="divider"></li>
        <li>
            <a href="#">
                <div>
                    <i class="fa fa-twitter fa-fw"></i>
                    <span class="pull-right text-muted
small">12 minutes ago</span>
                </div>
            </a>
        </li>
        <li class="divider"></li>
        <li>
            <a href="#">
                <div>
                    <i class="fa fa-envelope fa-fw"></i>
                    <span class="pull-right text-muted
small">4 minutes ago</span>
                </div>
            </a>
        </li>
        <li class="divider"></li>
        <li>

```

```

        <a href="#">
            <div>
                <i class="fa fa-tasks fa-fw"></i>
New Task
                <span class="pull-right text-muted
small">4 minutes ago</span>
            </div>
        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a href="#">
            <div>
                <i class="fa fa-upload fa-fw"></i>
Server Rebooted
                <span class="pull-right text-muted
small">4 minutes ago</span>
            </div>
        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a class="text-center" href="#">
            <strong>See All Alerts</strong>
            <i class="fa fa-angle-right"></i>
        </a>
    </li>
</ul>
<!-- /.dropdown-alerts -->
</li>
<!-- /.dropdown -->
<li class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown"
href="#">
        <i class="fa fa-user fa-fw"></i> <i class="fa
fa-caret-down"></i>
    </a>
    <ul class="dropdown-menu dropdown-user">
        <li><a href="#"><i class="fa fa-user fa-fw"></i>
User Profile</a>
        </li>
        <li><a href="#"><i class="fa fa-gear fa-fw"></i>
Settings</a>
        </li>
        <li class="divider"></li>
        <li><a href="../security/login.html"><i
class="fa fa-sign-out fa-fw"></i> Logout</a>
        </li>
    </ul>
    <!-- /.dropdown-user -->
</li>
<!-- /.dropdown -->
</ul>
<!-- /.navbar-top-links -->

```

```

<div class="navbar-default sidebar" role="navigation">
  <div class="sidebar-nav navbar-collapse">
    <ul class="nav" id="side-menu">
      <li>
        <a href="#"><i class="fa fa-pie-chart fa-fw"></i> New Portfolio<span class="fa arrow"></span></a>
        <ul class="nav nav-second-level">
          <li>
            <a href="portfolio">Quick Create</a>
          </li>
          <li>
            <a href="customportfolio">Professional</a>
          </li>
        </ul>
      </li>
      <li>
        <a href="/compare"><i class="fa fa-table fa-fw"></i> Compare Portfolios</a>
      </li>
    </ul>
  </div>
  <!-- /.sidebar-collapse -->
</div>
<!-- /.navbar-static-side -->
</nav>

<div id="page-wrapper">
  <div class="row">
    <div class="col-lg-12">
      <h1 class="page-header">{{ name }}</h1>
    </div>
    <!-- /.col-lg-12 -->
  </div>
  <!-- /.row -->

  {% block content %}{% endblock %}
</div>

</div>
<!-- /#wrapper -->

<!-- jQuery -->
<script src="../../static/vendor/jquery/jquery.min.js"></script>

<!-- Bootstrap Core JavaScript -->
<script src="../../static/vendor/bootstrap/js/bootstrap.min.js"></script>

<!-- Metis Menu Plugin JavaScript -->
<script src="../../static/vendor/metisMenu/metisMenu.min.js"></script>

```

```

<!-- Morris Charts JavaScript -->
<script src="../../static/vendor/raphael/raphael.min.js"></script>
<script src="../../static/vendor/morrisjs/morris.min.js"></script>
<script src="../../static/data/morris-data.js"></script>

<!-- DataTables JavaScript -->
<script
src="https://cdn.datatables.net/1.10.12/js/jquery.dataTables.min.js"
></script>
<script>
    $(document).ready(function() {
        $('#example').DataTable({
            "scrollX": true
        });
    });
</script>

<!-- Custom Theme JavaScript -->
<script src="../../static/dist/js/sb-admin-2.js"></script>
</body>
</html>

```

index.html

```

<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>index</title>

    <!-- Bootstrap Core CSS -->
    <link href="../../static/vendor/bootstrap/css/bootstrap.min.css"
rel="stylesheet">

    <!-- MetisMenu CSS -->
    <link href="../../static/vendor/metisMenu/metisMenu.min.css"
rel="stylesheet">

    <!-- Custom CSS -->
    <link href="../../static/dist/css/sb-admin-2.css" rel="stylesheet">

    <!-- Morris Charts CSS -->
    <link href="../../static/vendor/morrisjs/morris.css"
rel="stylesheet">

```

```

<!-- DataTables CSS -->
<link href="../../static/vendor/datatables-
plugins/dataTables.bootstrap.css" rel="stylesheet">

<!-- DataTables Responsive CSS -->
<link href="../../static/vendor/datatables-
responsive/dataTables.responsive.css" rel="stylesheet">

<!-- Custom Fonts -->
<link href="../../static/vendor/font-awesome/css/font-
awesome.min.css" rel="stylesheet" type="text/css">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via
file:// -->
<!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></scr
ipt>
    <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></
script>
<![endif]-->

</head>

<body>

<div id="wrapper">

    <!-- Navigation -->
    <nav class="navbar navbar-default navbar-static-top"
role="navigation" style="margin-bottom: 0">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="/">Portfolio Helper
v2.0</a>
        </div>
        <!-- /.navbar-header -->

        <ul class="nav navbar-top-links navbar-right">
            <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown"
href="#">
                    <i class="fa fa-envelope fa-fw"></i> <i

```

```

class="fa fa-caret-down"></i>
</a>
<ul class="dropdown-menu dropdown-messages">
  <li>
    <a href="#">
      <div>
        <strong>John Smith</strong>
        <span class="pull-right text-muted">
          <em>Yesterday</em>
        </span>
      </div>
      <div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque eleifend...</div>
    </a>
  </li>
  <li class="divider"></li>
  <li>
    <a href="#">
      <div>
        <strong>John Smith</strong>
        <span class="pull-right text-muted">
          <em>Yesterday</em>
        </span>
      </div>
      <div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque eleifend...</div>
    </a>
  </li>
  <li class="divider"></li>
  <li>
    <a href="#">
      <div>
        <strong>John Smith</strong>
        <span class="pull-right text-muted">
          <em>Yesterday</em>
        </span>
      </div>
      <div>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque eleifend...</div>
    </a>
  </li>
  <li class="divider"></li>
  <li>
    <a class="text-center" href="#">
      <strong>Read All Messages</strong>
      <i class="fa fa-angle-right"></i>
    </a>
  </li>
</ul>
<!-- /.dropdown-messages -->
</li>
<!-- /.dropdown -->
<li class="dropdown">
  <a class="dropdown-toggle" data-toggle="dropdown"

```



```

href="#">
    <i class="fa fa-bell fa-fw"></i> <i class="fa
fa-caret-down"></i>
    </a>
    <ul class="dropdown-menu dropdown-alerts">
    <li>
        <a href="#">
            <div>
                <i class="fa fa-comment fa-fw"></i>
New Comment
                <span class="pull-right text-muted
small">4 minutes ago</span>
            </div>
        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a href="#">
            <div>
                <i class="fa fa-twitter fa-fw"></i>
3 New Followers
                <span class="pull-right text-muted
small">12 minutes ago</span>
            </div>
        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a href="#">
            <div>
                <i class="fa fa-envelope fa-fw"></i>
Message Sent
                <span class="pull-right text-muted
small">4 minutes ago</span>
            </div>
        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a href="#">
            <div>
                <i class="fa fa-tasks fa-fw"></i>
New Task
                <span class="pull-right text-muted
small">4 minutes ago</span>
            </div>
        </a>
    </li>
    <li class="divider"></li>
    <li>
        <a href="#">
            <div>
                <i class="fa fa-upload fa-fw"></i>
Server Rebooted

```

```

small">4 minutes ago</span>
    </div>
    </a>
</li>
<li class="divider"></li>
<li>
    <a class="text-center" href="#">
        <strong>See All Alerts</strong>
        <i class="fa fa-angle-right"></i>
    </a>
</li>
</ul>
<!-- /.dropdown-alerts -->
</li>
<!-- /.dropdown -->
<li class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown"
href="#">
        <i class="fa fa-user fa-fw"></i> <i class="fa
fa-caret-down"></i>
    </a>
    <ul class="dropdown-menu dropdown-user">
        <li><a href="#"><i class="fa fa-user fa-fw"></i>
User Profile</a>
        </li>
        <li><a href="#"><i class="fa fa-gear fa-fw"></i>
Settings</a>
        </li>
        <li class="divider"></li>
        <li><a href="security/login.html"><i class="fa
fa-sign-out fa-fw"></i> Logout</a>
        </li>
    </ul>
    <!-- /.dropdown-user -->
</li>
<!-- /.dropdown -->
</ul>
<!-- /.navbar-top-links -->
</nav>
<!-- Header -->
<a name="about"></a>
<div class="intro-header">
    <div class="container">

        <div class="row">
            <div class="col-lg-12">
                <div class="intro-message">
                    <h1>Portfolio Helper</h1>
                    <h3>Black Litterman Optimization Tool</h3>
                    <hr class="intro-divider">
                    <ul class="list-inline intro-social-
buttons">
                        <li>

```

```

                                <a href="portfolio" class="btn btn-
default btn-lg"><i class="fa fa-pie-chart fa-fw"></i> <span
class="network-name">Enter</span></a>
                                </li>
                                <li>
                                <a href="login" class="btn btn-
default btn-lg"><i class="fa fa-key fa-fw"></i> <span
class="network-name">Log In</span></a>
                                </li>
                                <li>
                                <a
href="https://github.com/chenbowen184/capstone" class="btn btn-
default btn-lg"><i class="fa fa-github fa-fw"></i> <span
class="network-name">Github</span></a>
                                </li>
                                </ul>
                                </div>
                                </div>
                                </div>
                                </div>
                                <!-- /.container -->
                                </div>
                                <!-- jQuery -->
                                <script src="../../static/vendor/jquery/jquery.min.js"></script>

                                <!-- Bootstrap Core JavaScript -->
                                <script
src="../../static/vendor/bootstrap/js/bootstrap.min.js"></script>

                                <!-- Metis Menu Plugin JavaScript -->
                                <script src="../../static/vendor/metisMenu/metisMenu.min.js"></script>

                                <!-- Morris Charts JavaScript -->
                                <script src="../../static/vendor/raphael/raphael.min.js"></script>
                                <script src="../../static/vendor/morrisjs/morris.min.js"></script>
                                <script src="../../static/data/morris-data.js"></script>

                                <!-- DataTables JavaScript -->
                                <script
src="https://cdn.datatables.net/1.10.12/js/jquery.dataTables.min.js"
></script>
                                <script>
                                $(document).ready(function() {
                                    $('#example').DataTable({
                                        "scrollX": true
                                    });
                                } );
                                </script>

                                <!-- Custom Theme JavaScript -->
                                <script src="../../static/dist/js/sb-admin-2.js"></script>
                                </body>
                                </html>

```

portfolio.html

```
{% extends "base.html" %}

{% block content %}
    <div class="row">
        <div class="col-xs-12">
            <p>The Quick Create option was created with the goal of
giving non-institutional users access to similar optimization models
as professionals. Select the number of stocks from each sector of
the S&P500 you would like to include in the optimization and our
algorithm will determine the portfolio weights based on the assets'
historical returns and trends.</p>
        </div>
    </div>
    <div class="row">
        <div class="col-md-8">
            <form method="get"
action="/portfolio/get_optimal_portfolio_black_litterman">
                <div class="form-group">
                    <label>Number of Assets
from Each Sector</label>
                    <select multiple
class="form-control" name="number">
                        <option>1</option>
                        <option>2</option>
                        <option>3</option>
                        <option>4</option>
                        <option>5</option>
                    </select>
                </div>
                <button type="submit" class="btn btn-
default">Optimize</button>
                <br>
            </form>
        </div>
    </div>
{% endblock %}
```

login.html

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>index</title>

  <!-- Bootstrap Core CSS -->
  <link href="../../static/vendor/bootstrap/css/bootstrap.min.css"
rel="stylesheet">

  <!-- MetisMenu CSS -->
  <link href="../../static/vendor/metisMenu/metisMenu.min.css"
rel="stylesheet">

  <!-- Custom CSS -->
  <link href="../../static/dist/css/sb-admin-2.css" rel="stylesheet">

  <!-- Morris Charts CSS -->
  <link href="../../static/vendor/morrisjs/morris.css"
rel="stylesheet">

  <!-- DataTables CSS -->
  <link href="../../static/vendor/datatables-
plugins/dataTables.bootstrap.css" rel="stylesheet">
```

```

    <!-- DataTables Responsive CSS -->
    <link href="../../static/vendor/datatables-responsive/datatables.responsive.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="../../static/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->
</head>

<body>

    <div class="container">
        <div class="row">
            <div class="col-md-4 col-md-offset-4">
                <div class="login-panel panel panel-default">
                    <div class="panel-heading">
                        <h3 class="panel-title">Please Sign In</h3>
                    </div>
                    <div class="panel-body">
                        <form role="form">
                            <fieldset>
                                <div class="form-group">
                                    <input class="form-control"
placeholder="E-mail" name="email" type="email" autofocus>
                                </div>
                                <div class="form-group">
                                    <input class="form-control"
placeholder="Password" name="password" type="password" value="">
                                </div>
                                <div class="checkbox">
                                    <label>
                                        <input name="remember"
type="checkbox" value="Remember Me">Remember Me
                                    </label>
                                </div>
                                <!-- Change this to a button or input when using this as a form -->
                                <a href="portfolio" class="btn btn-lg btn-success btn-block">Login</a>
                            </fieldset>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        </fieldset>
    </form>
</div>
</div>
</div>
</div>
</div>

<!-- jQuery -->
<script src="../../static/vendor/jquery/jquery.min.js"></script>

<!-- Bootstrap Core JavaScript -->
<script
src="../../static/vendor/bootstrap/js/bootstrap.min.js"></script>

<!-- Metis Menu Plugin JavaScript -->
<script src="../../static/vendor/metisMenu/metisMenu.min.js"></script>

</body>
</html>

```

customportfolio.html

```

{% extends "base.html" %}

{% block content %}
    <div class="row">
        <div class="col-xs-12">
            <p>Custom portfolios are designed for a professional
user who desires a more flexible model. Enter both the assets to be
included and related views.</p>
        </div>
    </div>
    <form method="post"
action="/customportfolio/get_optimal_customportfolio_black_litterman
">
        <div class="row">
            <div class="col-md-8">
                <label>Asset Tickers</label>
                <input class="form-control" placeholder="GOOGL,

```

```

AAPL, MSFT" name="assets">
    </div>
</div>

</br>
<div class="row">
    <div class="col-md-8">
        <label>Relative View 1</label>
        <input class="form-control" placeholder="GOOGL >
MSFT by 2%" name="view1">
    </div>
</div>

</br>
<div class="row">
    <div class="col-md-8">
        <label>Relative View 2</label>
        <input class="form-control" placeholder="MSFT <
AAPL by 3%" name="view2">
    </div>
</div>

</br>
<div class="row">
    <div class="col-md-8">
        <button type="submit" class="btn btn-
default">Optimize</button>
    </div>
</div>
</form>

{% endblock %}

```

result.html

```

{% extends "base.html" %}

{% block content %}

    <div class="row">
        <div class="col-xs-12">
            <div class="panel panel-success">
                <div class="panel-heading">
                    <div class="row">
                        <div class="col-xs-12 text-left">
                            <div class="huge">{{ Return|safe
}}%</div>
                        <div>Expected annual
return</div>

```



```

        </div>
    </div>
</div>
</div>
</div>
</div>
<!-- /.row -->
<div class="row">
    <div class="col-lg-3 col-md-6">
        <div class="panel panel-default">
            <div class="panel-heading">
                <div class="row">
                    <div class="col-xs-3">
                        <i class="fa fa-angle-double-up
fa-5x"></i>
                    </div>
                    <div class="col-xs-9 text-right">
                        <div class="huge">{{
upRSIstock|safe }}</div>
                        <div>Lowest RSI: </div>
                        <div>{{ upRSI|safe }}</div>
                    </div>
                </div>
            </div>
            <a href="#">
                <div class="panel-footer">
                    <span class="pull-left">View
Details</span>
                    <span class="pull-right"><i
class="fa fa-arrow-circle-right"></i></span>
                    <div class="clearfix"></div>
                </div>
            </a>
        </div>
    </div>
    <div class="col-lg-3 col-md-6">
        <div class="panel panel-default">
            <div class="panel-heading">
                <div class="row">
                    <div class="col-xs-3">
                        <i class="fa fa-angle-double-
down fa-5x"></i>
                    </div>
                    <div class="col-xs-9 text-right">
                        <div class="huge">{{
downRSIstock|safe }}</div>
                        <div>Highest RSI: </div>
                        <div>{{ downRSI|safe }}</div>
                    </div>
                </div>
            </div>
            <a href="#">
                <div class="panel-footer">
                    <span class="pull-left">View

```

```

Details</span>
        <span class="pull-right"><i
class="fa fa-arrow-circle-right"></i></span>
        <div class="clearfix"></div>
    </div>
</a>
</div>
</div>
<div class="col-lg-3 col-md-6">
    <div class="panel panel-default">
        <div class="panel-heading">
            <div class="row">
                <div class="col-xs-3">
                    <i class="fa fa-angle-double-up
fa-5x"></i>
                </div>
                <div class="col-xs-9 text-right">
                    <div class="huge">{{
upST0stock|safe }}</div>
                    <div>Lowest ST0: </div>
                    <div>{{ upST0|safe }}</div>
                </div>
            </div>
        </div>
        <a href="#">
            <div class="panel-footer">
                <span class="pull-left">View
Details</span>
                <span class="pull-right"><i
class="fa fa-arrow-circle-right"></i></span>
                <div class="clearfix"></div>
            </div>
        </a>
    </div>
</div>
<div class="col-lg-3 col-md-6">
    <div class="panel panel-default">
        <div class="panel-heading">
            <div class="row">
                <div class="col-xs-3">
                    <i class="fa fa-angle-double-
down fa-5x"></i>
                </div>
                <div class="col-xs-9 text-right">
                    <div class="huge">{{
downST0stock|safe }}</div>
                    <div>Highest ST0: </div>
                    <div>{{ downST0|safe }} </div>
                </div>
            </div>
        </div>
        <a href="#">
            <div class="panel-footer">
                <span class="pull-left">View

```

```

Details</span>
                                <span class="pull-right"><i
class="fa fa-arrow-circle-right"></i></span>
                                <div class="clearfix"></div>
                                </div>
                                </a>
                                </div>
                                </div>
                                </div>
                                <!-- /.row -->
                                <div class="row">
                                    <div class="col-md-3">
                                        <div class="table-responsive">
                                            {{ data|safe }}
                                        </div>
                                    </div>
                                    <div class="col-md-6">
                                        
                                    </div>
                                </div>
                                <!-- /.row -->
                                {% endblock %}

```

customresult.html

```
{% extends "base.html" %}

{% block content %}

    <div class="row">
        <div class="col-xs-12">
            <div class="panel panel-success">
                <div class="panel-heading">
                    <div class="row">
                        <div class="col-xs-12 text-left">
                            <div class="huge">{{ Return|safe
}}%</div>
                                <div>Expected return</div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <!-- /.row -->
        <div class="row">
            <div class="col-md-3">
                <div class="table-responsive">
                    {{ data|safe }}
                </div>
            </div>
            <div class="col-md-6">
                
            </div>
        </div>
        <!-- /.row -->
    {% endblock %}
```

Appendix F: Business logic

Black_Litterman.py

```
import numpy as np
from scipy import linalg
from Return_Data_Collector import get_asset_return_data, get_SP500,
get_market_portfolio_weights, get_price_changes_data
import cvxpy
import pandas as pd

def find_relative_strength_index(stock):
    # find the gains and losses in stocks
    stock = stock.diff()
    # assemble the gains in one column
    stock['gain'] = np.where(stock[stock.columns[0]]>0,
stock[stock.columns[0]], 0)
    # assemble the losses in one column
    stock['loss'] = np.where(stock[stock.columns[0]]<0,
stock[stock.columns[0]], 0)
    # take the absolute value of losses
    stock['loss'] = stock['loss'].apply(lambda x: abs(x))
    # find the 14 day moving average of gains and losses
    stock['average gain'] = stock['gain'].rolling(window = 14, center
= False).mean()
    stock['average loss'] = stock['loss'].rolling(window = 14, center
= False).mean()
    # find the ration of RS by Average gain/Average loss
    stock['RS'] = (stock['average gain']*1.0)/stock['average loss']
    # find RSI by 100 - 100/RS
    stock['RSI'] = 100 - 100/(1+stock['RS'])

    return stock['RSI']

def get_RSI_assets(assets):
    # Get the RSI values for each stock in list_assets
    for i in assets.columns:
        assets[i] =
find_relative_strength_index(assets[i].to_frame())
    return assets

def find_stochastic_osciliator(high, low, close):
```

```

# Find the 20 days rolling high stock values and low stock values
high_14 = high.rolling(window=20, center = False).max()
low_14 = low.rolling(window=20, center = False).min()

# Find Stochastic Oscillator using (close - low)/(high - low)
stochastic_oscillator = (close-low_14)*100.0/(high_14-low_14)
return stochastic_oscillator

def update_relevant_assets_RSI(RSI):

# Identify the stock with largest RSI and smallest RSI, also save
the RSI values
    largest_rsi_stock = RSI[-1:].idxmax(axis=1)[0]
    smallest_rsi_stock = RSI[-1:].idxmin(axis=1)[0]
    largest_rsi = RSI[-1:].max(axis=1)[0]
    smallest_rsi = RSI[-1:].min(axis=1)[0]

# Append
    relevant_assets = []
    relevant_assets.append(largest_rsi_stock)
    relevant_assets.append(smallest_rsi_stock)
    P_views_values = []
    Q_views_values = []

# If largest RSI value is greater than the smallest RSI value, we
set the view as largest RSI stock under perform smallest RSI stock
by 2%, otherwise, we set market as neutral, i.e no views relvant to
RSI
    if (largest_rsi >= 80) & (smallest_rsi <= 20):
        P_views_values.append(-1)
        P_views_values.append(1)
        Q_views_values.append(0.02)
    else:
        P_views_values.append(0)
        P_views_values.append(0)
        Q_views_values.append(0)

    return [relevant_assets, P_views_values, Q_views_values,
[largest_rsi_stock, largest_rsi], [smallest_rsi_stock,
smallest_rsi]]

def update_relevant_assets_Stochastic(STO):

# Identify the stock with largest STO and smallest STO, also save
the STO values

    largest_STO_stock = STO[-1:].idxmax(axis=1)[0]
    smallest_STO_stock = STO[-1:].idxmin(axis=1)[0]
    largest_STO = STO[-1:].max(axis=1)[0]
    smallest_STO = STO[-1:].min(axis=1)[0]

```

```

relevant_assets = []
relevant_assets.append(largest_STO_stock)
relevant_assets.append(smallest_STO_stock)
P_views_values = []
Q_views_values = []

# If largest STO value is greater than the smallest STO value, we
set the view as largest RSI stock under perform smallest RSI stock
by 2%, otherwise, we set market as neutral, i.e no views relvant to
STO

    if (largest_STO >= 80) & (smallest_STO <= 20):
        P_views_values.append(-1)
        P_views_values.append(1)
        Q_views_values.append(0.01)
    else:
        P_views_values.append(0)
        P_views_values.append(0)
        Q_views_values.append(0)

    return [relevant_assets, P_views_values, Q_views_values,
            [largest_STO_stock, largest_STO], [smallest_STO_stock,
            smallest_STO]]

def combine_momentum_oscillator_views(RSI_views, STO_views):

# Combine the views in the list form that could be processed by
update_views function
    relevant_assets = [RSI_views[0],STO_views[0]]
    P_views_values = [RSI_views[1],STO_views[1]]
    Q_views_values = [RSI_views[2],STO_views[2]]

    return [relevant_assets, P_views_values, Q_views_values]

def update_views(list_assets, relevant_assets, P_views_values,
Q_views_values):

# This function automatically matched dimensions of user views
matrices P and Q, allowing the views input to be in a list form
    num_views = len(relevant_assets)
    P = np.zeros((num_views, len(list_assets)))
    # Update the pick matrix P
    P_views_index = []
    for i in range(num_views):
        view_i_index = []
        for j in relevant_assets[i]:
            view_i_index.append(list_assets.index(j))
        P_views_index.append(view_i_index)

    for i in range(num_views):
        for j in range(len(P_views_index[i])):

```

```

        index = P_views_index[i][j]
        P[i, index] = P_views_values[i][j]
    # update views matrix Q
    Q = np.array([Q_views_values[i] for i in
range(len(Q_views_values))]).reshape(num_views, 1)

    return [P, Q]

def Black_Litterman(return_data, alpha, P, Q, wmkt):
    # P indicates which of the asset is relevant to the
investor's view, right now it is hard coded
    # Q represent the value of the views

    # set tau scalar
    tau = 0.0001
    # find the covariance matrix according to the return data
    sigma = return_data.cov().as_matrix()
    # pre-calculate the product fo tau and sigma
    scaled_sigma = tau*sigma
    # pre-calculate [tau*sigma]^-1
    scaled_sigma_inv = linalg.inv(scaled_sigma)

    name_asset = return_data.columns
    # The implied return of the asset, calculated from CAPM
asset pricing model
    pi = (alpha*sigma).dot(wmkt)

    # The standard error in investor's views
    Omega = (P.dot(scaled_sigma).dot(P.T)) * np.eye(Q.shape[0])
    try:
        Omega_inv = linalg.inv(Omega)
    except:
        Omega_inv = Omega

    # Find combined returns and combined covariance for the
updated quadratic optimization
    combined_return = linalg.inv(scaled_sigma_inv +
P.T.dot(Omega_inv).dot(P)).dot(np.dot(scaled_sigma_inv,pi).reshape(l
en(sigma), 1) + np.dot(np.dot(P.T,Omega_inv),Q))
    combined_covariance = sigma + linalg.inv(scaled_sigma_inv +
P.T.dot(Omega_inv).dot(P))
    #combined_cov_inv = linalg.inv(combined_covariance)

    # For some reason this doesn't work

    num_asset = len(sigma)
    w = cvxpy.Variable(num_asset) #30 assets

    constraints = []
    # constraints.append(cvxpy.abs(w) <= 0.4)
    constraints.append(w >= 0)
    constraints.append(cvxpy.sum_entries(w) == 1)

```



```

        objective = cvxpy.Maximize(combined_return.T * w - 0.5 *
alpha * cvxpy.quad_form(w, combined_covariance))
        # objective = cvxpy.Maximize(combined_return*w -
0.5*alpha*(w.T*combined_covariance*w))
        problem = cvxpy.Problem(objective, constraints)
        problem.solve(solver='CVXOPT', verbose=True)

        Return = (combined_return.T * w - 0.5 * alpha *
cvxpy.quad_form(w, combined_covariance)).value
        pd.options.display.float_format = '{:.4f}%'.format
        weights = pd.DataFrame(w.value, index=name_asset,
columns=['Holding'])
        weights['Holding'] = weights['Holding'].apply(lambda x :
x*100)
        return [weights, Return]

```

Return_Data_Collector.py

```

import pandas as pd
import datetime
from pandas_datareader import data
import datapackage
from dateutil.relativedelta import relativedelta

def get_asset_return_data(asset_list,
                           price_type='Close',
                           source='yahoo',
                           start_date='2013-01-01',
                           end_date=datetime.datetime.today()):
    start_date = pd.to_datetime(start_date)
    end_date = pd.to_datetime(end_date)
    date_range = pd.bdate_range(start_date, end_date)
    try:
        df_price = pd.DataFrame(index=date_range,
columns=asset_list)

```

```

        for ls in asset_list:
            try:
                price_data = data.DataReader(ls, source, start_date,
end_date)[price_type]
                price_data.rename(ls, inplace=True)
                df_price[ls] = price_data.T
            except Exception as dr:
                pass
            # df_price = df_price.drop('NEE', 1)
            df_price.dropna(inplace=True)
            df_return = df_price.resample('M').apply(lambda x: x[-
1]).pct_change()
            df_momentum = df_price.resample('M').apply(lambda x: x[-
1]).diff()
        except Exception as e:
            raise RuntimeError(e)

    return {'df_price': df_price,
            'df_return': df_return,
            'momentum': df_momentum}

def get_price_changes_data(asset_list,
                           price_type='Close',
                           source='yahoo',
                           start_date=datetime.datetime.today() -
relativedelta(months=2),
                           end_date=datetime.datetime.today()):
    start_date = pd.to_datetime(start_date)
    end_date = pd.to_datetime(end_date)
    date_range = pd.bdate_range(start_date, end_date)
    try:
        df_price = pd.DataFrame(index=date_range,
columns=asset_list)

        for ls in asset_list:
            try:
                price_data = data.DataReader(ls, source, start_date,
end_date)[price_type]
                price_data.rename(ls, inplace=True)
                df_price[ls] = price_data.T
            except Exception as dr:
                pass
            # df_price = df_price.drop('NEE', 1)
            df_price.dropna(inplace=True)

    except Exception as e:
        raise RuntimeError(e)

    return df_price

def get_SP500():
    # get the S&P 500 Data from datapackage API
    dp = datapackage.DataPackage('http://data.okfn.org/data/core/s-

```

```

and-p-500-companies/datapackage.json')
    SP500 = pd.DataFrame(dp.resources[1].data)
    SP500 = SP500[['Symbol', 'Name', 'Sector', 'Price', 'Dividend
Yield', 'Price/Earnings', 'Earnings/Share',
                  'Book Value', '52 week low', '52 week high',
'Market Cap', 'EBITDA', 'Price/Sales', 'Price/Book',
                  'SEC Filings']]
# Remove the stocks that contains a large amount of missing data
    SP500 = SP500[SP500['Symbol'] <> 'NEE']
    SP500 = SP500[SP500['Symbol'] <> 'PSX']
    SP500 = SP500[SP500['Symbol'] <> 'ICE']

    return SP500

def get_market_portfolio_weights_customized(SP500, chosen_assets):
    # Find the market portfolio constituents and it weights from S&P
500, takes in the list and find the toal market weights

    SP500 = SP500.drop(list(SP500[SP500['Market
Cap'].isnull()].index.values))

    Portfolio = []

    # find the market weights by summing up all the market cap data,
weighted by its individual data

    for i in chosen_assets:
        Selected_stock = SP500[SP500['Symbol'] == i]

        Portfolio.append(Selected_stock)

    Portfolio = pd.concat(Portfolio)
    total_market_cap = Portfolio['Market Cap'].apply(float).sum()
    Portfolio['market portfolio weights'] = Portfolio['Market
Cap'].apply(float) / total_market_cap

    portfllo_weights = Portfolio[['Symbol', 'market portfolio
weights']]

    return portfllo_weights

def get_market_portfolio_weights(SP500, assets_per_sector):
    # Find the market portfolio constituents and it weights from S&P
500

    SP500 = SP500.drop(list(SP500[SP500['Market
Cap'].isnull()].index.values))

    SP500_sectorspecific = SP500.groupby('Sector')
    Sectors = SP500_sectorspecific['Sector'].unique()

```

```

Portfolio = []

for sector in Sectors:
    Selected_stock =
SP500_sectorspecific.get_group(sector[0]).sort_values(by=['Earnings/
Share'],
ascending=[0]).head(assets_per_sector)
    Portfolio.append(Selected_stock)

Portfolio = pd.concat(Portfolio)
total_market_cap = Portfolio['Market Cap'].apply(float).sum()
Portfolio['market portfolio weights'] = Portfolio['Market
Cap'].apply(float) / total_market_cap

portflilio_weights = Portfolio[['Symbol', 'market portfolio
weights']]

return portflilio_weights

```