A Project Report

On

Portfolio Optimization with Known Parameters: Stock Market

Submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

Submitted By: **Under the Supervision of:**

Apoorva Anand (17102089) Dr. Ankit Garg

Department of Electronics and Communication Engineering

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

December, 2019

TABLE OF CONTENTS

CERTIFICATE

It is certified that the work contained in the project report titled “**Portfolio Optimization with known Parameters**” by “**Apoorva Anand**” has been carried out under my supervision and this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr. Ankit Garg

ECE

JIIT NOIDA

December, 2019

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude and deep regards to my mentor Dr. Ankit Garg for the continuous support in the B.Tech Minor Project, for his patience, motivation, enthusiasm and immense knowledge. I owe the success of this project to him. His guidance helped me in all time of my project and writing of report. I have to thank my external Mr. Shivaji Tyagi in my intermediate project evaluation and presentation for his encouragement, insightful comments and hard questions.

Also would like to extend the gratefulness to my parents and family for the constant love and support.

Signature of Students

Apoorva Anand

Chapter 1 – INTRODUCTION

1.1 Financial Engineering

Financial engineering is the use of mathematical techniques to solve financial problems. Financial engineering uses tools and knowledge from the fields of computer science, electrical engineering, statistics, economics, and applied mathematics to address current financial issues as well as to devise new and innovative financial products.

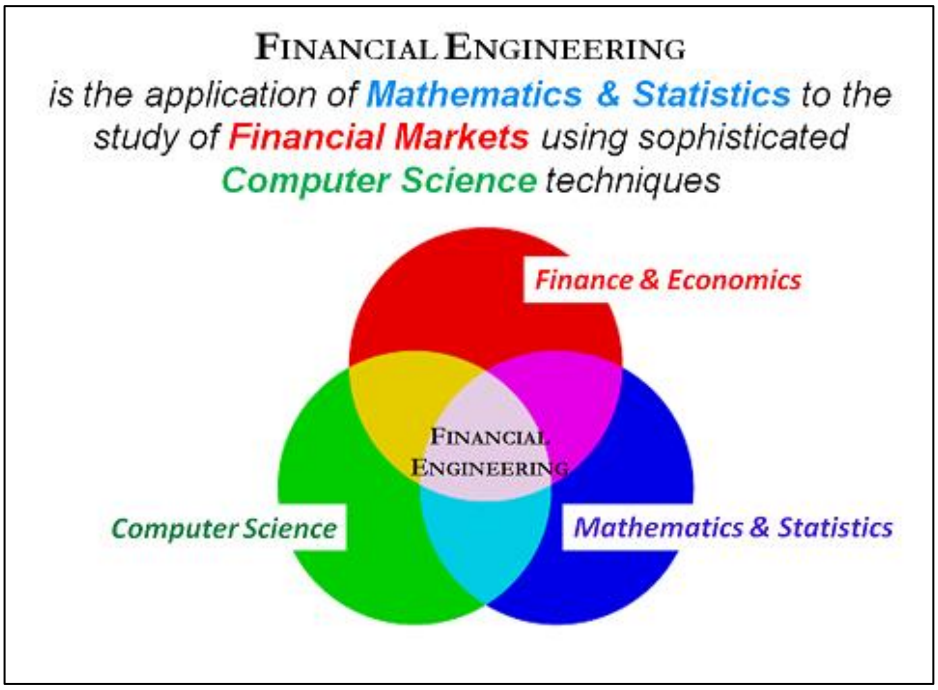
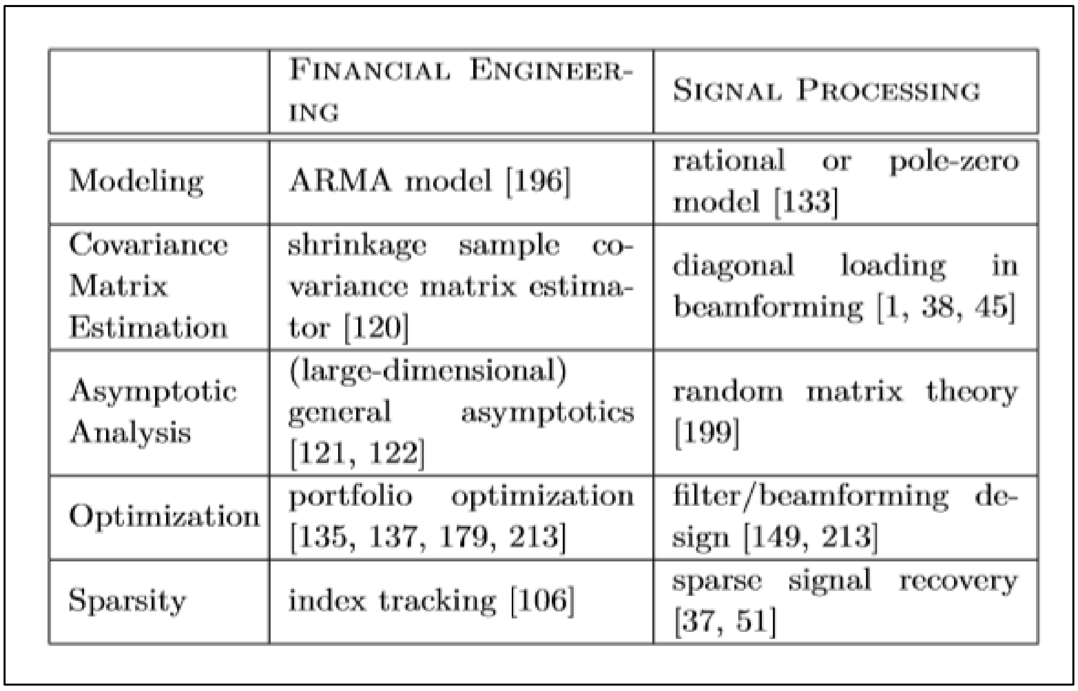

FINANCIAL ENGINEERING
is the application of Mathematics & Statistics to the study of Financial Markets using sophisticated Computer Science techniques
Finance & Economics
FINANCIAL ENGINEERING
Computer Science
Mathematics & Statistics

Fig 1. Financial Engineering

1.2 A Signal Processing Perspective on Financial Engineering

Despite the different natures of financial engineering and electrical engineering, both areas are intimately connected on a mathematical level. The foundations of financial engineering lie on the statistical analysis of numerical time series and the modeling of the behavior of the financial markets in order to perform predictions and systematically optimize investment strategies. Similarly, the foundations of electrical engineering, for instance, wireless communication systems, lie on statistical signal processing and the modeling of communication channels in order to perform predictions and systematically optimize transmission strategies.

Thus, this monograph is about investment in financial assets treated as a signal processing and optimization problem



	Financial Engineering	Signal Processing
Modeling	ARMA model [196]	rational or pole-zero model [133]
Covariance Matrix Estimation	shrinkage sample covariance matrix estimator [120]	diagonal loading in beamforming [1, 38, 45]
Asymptotic Analysis	(large-dimensional) general asymptotics [121, 122]	random matrix theory [199]
Optimization	portfolio optimization [135, 137, 179, 213]	filter/beamforming design [149, 213]
Sparsity	index tracking [106]	sparse signal recovery [37, 51]

Fig 2. Connections between Financial Engineering and Signal Processing

1.3 Portfolio Optimization

Portfolio:-A portfolio is a grouping of financial assets such as stocks, bonds, commodities, currencies and cash equivalents, as well as their fund counterparts. Investors should construct an investment portfolio in accordance with their risk tolerance and investing objectives

Optimization:- An optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries

The portfolio optimization plays an important role in determining the portfolio strategies for investors. What investors hope to achieve through a portfolio optimization is to maximize portfolio return and minimize portfolio risk. Since the return changes based on risk investors have to balance the contradiction between risk and return for their investment. There is therefore no single optimized portfolio to satisfy all investors. The optimal portfolio is determined by the preferences of the investor’s risk and return .

1.4 Markowitz Mean Variance Model

Until the publication of the modern portfolio theory by Harry Markowitz in 1952 in the article ”Portfolio Selection” , investors were focused on assessing the risk and return of individual securities when constructing portfolios. Investment decisions were based on identifying securities with the highest return and at less risk, then they were included in the investment portfolio. Markowitz offers another approach called diversification, where construction of the portfolio is made after evaluation of the overall portfolio risk, i.e. selectable portfolio in aggregate, not to construct a portfolio of distinct, preselected securities. Thus the emphasis is on the relationships between the characteristics of the assets rather than on the actual characteristics.

In 1990, Harry Markowitz, together with Morton Miller and William Sharpe, was awarded the Nobel Prize in Economics for his work on portfolio theory. The basic assumption in it is that the return securities for a period of time are random variables and therefore mathematical expectation and standard deviation can be calculated as the standard deviation is considered a measure of investment risk. The expected return on the portfolio E(rP) is a linear combination of the expected return on assets included in it, such factors are the relative shares of the assets in the portfolio. Investment risk is measured by the standard deviation σP, which depends on nonlinear standard deviations and covariances of return on individual assets.

The idea of diversification of Markowitz is subject to the fact that with increasing number of assets in the portfolio, the corresponding number of covariances becomes significantly larger than the number of assets and therefore the risk of the portfolio will depend to a greater degree on covariance between assets rather than on risk of individual assets.

1.5 Black Litterman Model

The Markowitz model has different problems, such as unintuitivity, ill-behaved portfolios and error maximization. Although the model has been an academic success, these problems have rendered the model of little use for investors. In 1992 Robert Litterman and Fischer Black proposed a improvement to this model that where suppose to make quantitative optimization tools of more use to investors. They claimed that by taking the CAPM equilibrium into consideration, they would significantly improve the model. With the equilibrium portfolio as a starting point, the investor would then make adjustments depending on the investors views on the market. One of the benefits of the

B-L model is that there is no need for the investor to have views on all the portfolio assets. The investor only adds his/her views when they have one, and else use the equilibrium expected excess returns. Although the B-L model is a large improvement from the Markowitz model, it isn’t actually changing the model. In the article ”Portfolio Selection” Harry Markowitz write about the two stages of portfolio optimization. The first stage being the collection and final beliefs about future returns, and the second stage the use of these beliefs to create a portfolio. As Markowitz states, his model is part of the last stage, concerning the use of available information and the creation of an efficient portfolio. The B-L model only tries to improve the Markowitz model by interfering with the first stage. So, the B-L model, rather than being a whole new optimization model, is just an add-on to the Markowitz model, interfering with the mean variance input. But this doesn’t make it any less useful, the model generates portfolios with considerable differences from those created by the Markowitz model.

Chapter 2 – TECHNICAL DESCRIPTION

2.1 Software Used

I. JUPYTER NOTEBOOK

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook interface

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

IPython

ØMQ

Tornado (web server)

jQuery

Bootstrap (front-end framework)

MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

Fig 3. Jupyter Notebook

II . NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Fig 4.Numpy Module

III. Pandas

“pandas” is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy andintuitive.It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's data frame provides and much more. "pandas" is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that "pandas" does well:

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets

- Flexible reshaping and pivoting of data sets
- Hierarchical labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. “pandas” is the ideal tool for all of these tasks.

Fig 5. Pandas Module

IV. Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community,and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

The image part with relationship ID rId17 was not found in the file.

Fig 6. Matplotlib logo

V. Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Here is some of the functionality that seaborn offers:

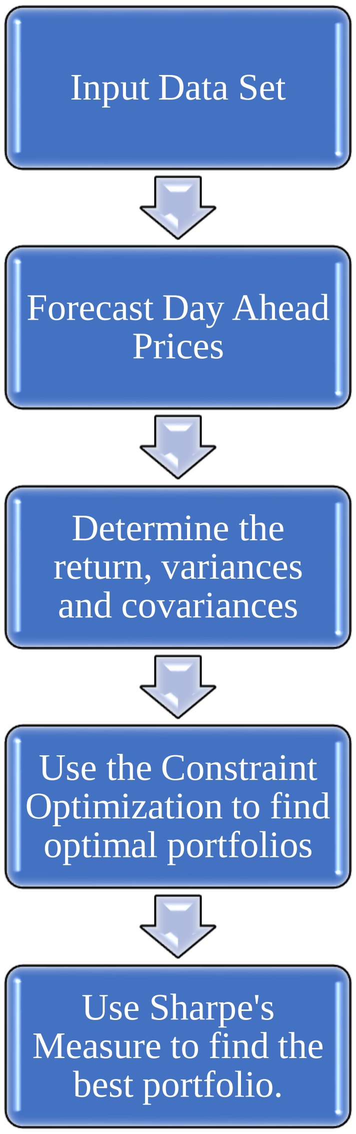
A dataset-oriented API for examining relationships between multiple variables

- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds dependent variables
- Convenient views onto the overall structure of complex datasets
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- Concise control over matplotlib figure styling with several built-in themes
- Tools for choosing color palettes that faithfully reveal patterns in your data

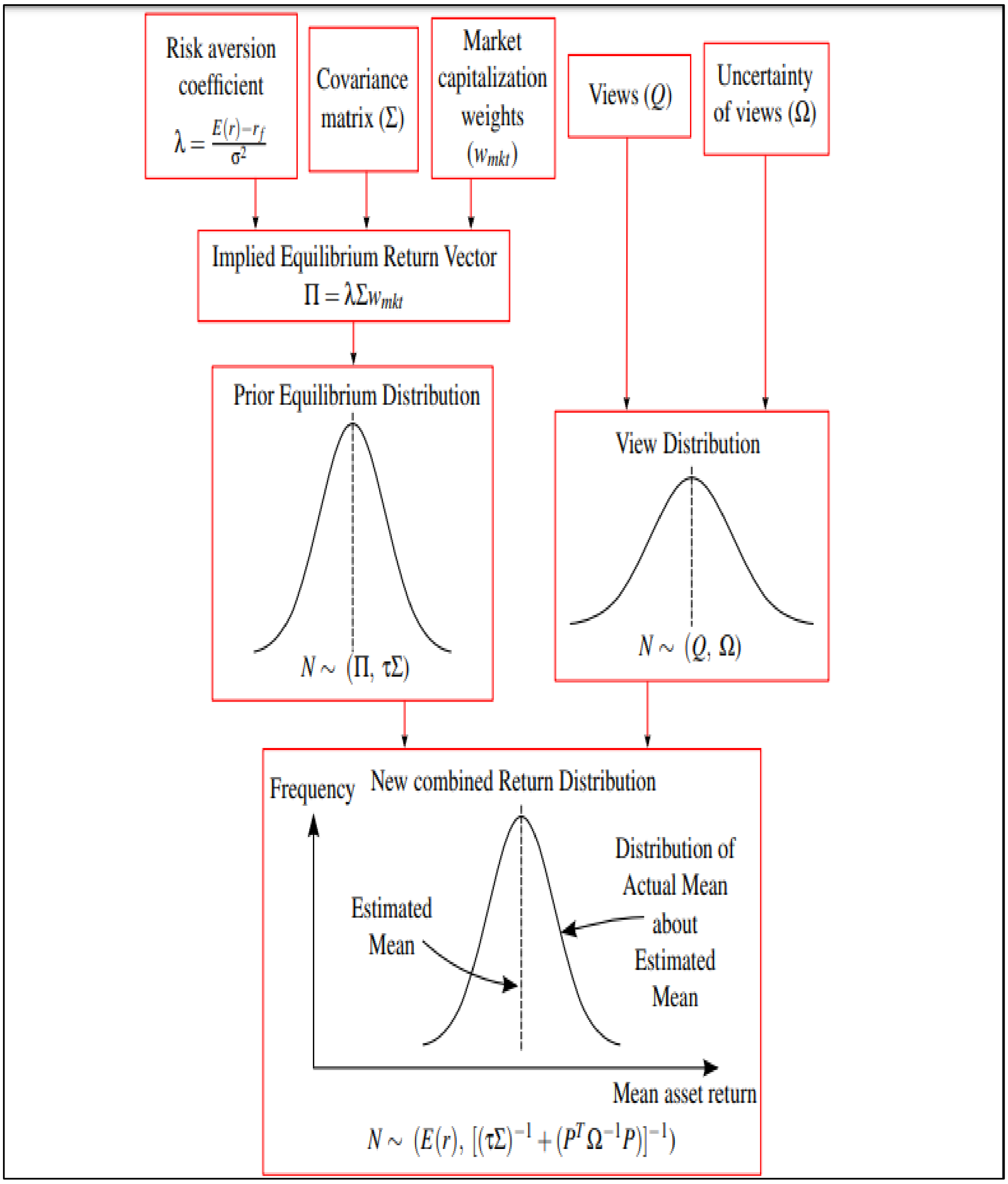
Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and array containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

2.2 Algorithm Used

I. MARKOWITZ MODEL


Input Data Set
Forecast Day Ahead Prices
Determine the return, variances and covariances
Use the Constraint Optimization to find optimal portfolios
Use Sharpe's Measure to find the best portfolio.

II. BLACK LITTERMAN MODEL



Risk aversion coefficient $\lambda = \frac{E(r)-r_f}{\sigma^2}$
Covariance matrix (Σ)
Market capitalization weights (w_{mkt})
Views (Q)
Uncertainty of views (Ω)
Implied Equilibrium Return Vector $\Pi = \lambda\Sigma w_{mkt}$
Prior Equilibrium Distribution
$N \sim (\Pi, \tau\Sigma)$
View Distribution
$N \sim (Q, \Omega)$
Frequency
New combined Return Distribution
Estimated Mean
Distribution of Actual Mean about Estimated Mean
Mean asset return
$N \sim (E(r), [(\tau\Sigma)^{-1} + (P^T\Omega^{-1}P)]^{-1})$

2.3 Parameters Used

I.MARKOWITZ MODEL

1. Rate of Return

$r_t = \frac{Pt-Pt-1}{Pt-1}$

2. Expected Return

$\mu_i = E(r^i) = \frac{\sum_{t=1}^{m} r_t^i}{m}$

3. Variance and standard deviation

$$\sigma_i^2 = \mathrm{Var}(r^i) = \frac{\sum_{t=1}^{m} (r_t^i - \mu_i)^2}{m-1}$$

$$\sigma_i = \sqrt{\sigma_i^2} = \sqrt{\frac{\sum_{t=1}^{m} (r_t^i - \mu_i)^2}{m-1}}$$

4. Covariance

$$\sigma_{ij} = Cov(r^i, r^j) = \frac{\sum_{t=1}^{m} (r_i^t - \mu_i)(r_j^t - \mu_j)}{m}$$

5. Eficient Frontier

   Every possible asset combination can be plotted in risk-return space, and the collection of all such possible portfolios defines a region in this space. The line along the upper edge of this region is known as the efficient frontier sometimes called the ”Markowitz bullet”. Mathematically, the efficient frontier is the intersection of the set of portfolios with minimum risk and the set of portfolios with maximum return

$$\max\ E(r_p) = \max \sum_{i=1}^{n} \omega_i \mu_i$$

$$\min\ \sigma_p = \min \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i} \omega_j \sigma_{ij}$$

$$0 \le \omega_i \le 1, i = 1,.....,n$$

$$\sum_{i=1}^{n} \omega_i = 1$$

6. Sharpe's Ratio

   All the portfolios on the efficient frontier are optimal depending on the investor's risk profile, that is, the choice of the parameters. However, one may still ask which portfolio may be the most meaningful in practice. Sharpe's measure is a measure of portfolio performance that gives the risk premium per unit of total risk, which is measured by the portfolio's standard deviation of return. The risk premium on a portfolio itself is the total portfolio return minus the risk-free rate.

   We maximize the sharpe ratio to get **the best optimized portfolio** which also falls on the Capital Market Line.

$$\underset{w}{\max imize}\ \frac{w^T\mu - r_f}{\sqrt{w^T \sum w}}$$

   subject to $w^T 1 = 1$

Where:

ω_i is a percentage of capital that will be invested in asset i;

r_i is the return on asset i;

μ_i the expected return on asset i;

μ_{ij} is the covariance between the return on assets i and j;

$E(r_p)$ is the expected return of the portfolio;

σ_p is the risk of the portfolio

II. BLACK LITTERMAN MODEL

1. λ : The risk aversion coefficient of the market portfolio.

$$\lambda = \frac{\mu_m - r_f}{\sigma_m^2}$$

2. τ: A number that specifies the uncertainty of the prior estimate of the mean ret.

3. Information about views: If m is the number of views and n the number assets, the model views are specified using the P-matrix (if more than 1 view) and Q-vector:

• P: is a m×n-matrix that identifies which assets are involved in either absolute or relative views (the Q-vector)

• Q: is the "views", a vector of n×1 elements

4. Σ: The size n×n variance-covariance matrix of excess returns (in the following, it is designated as the covariance matrix but the diagonal contains the variances for each asset)

5. w_{eq}: Equilibrium market capitalization weights (a vector of n × 1 weight factors for all assets that totals to 100 %). It can be calculated with an unconstrained mean-variance reverse optimization process , using the definition of the implied excess equilibrium return vector Π (n×1-vector), the risk aversion coefficient λ and Σ2:

$$\Pi = \lambda\Sigma weq$$

6. Standard BL Equation

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

The first factor can be considered as a normalization factor and the second factor is a vector composed of equilibrium returns Π and estimates Q. In the second factor, it can be seen that the first term has $\tau\Sigma)^{-1}$ is a weighting factor and $P^T\Omega^{-1}$ acts as a weighting factor for the second. It is also seen that if there are no views:

E(r) = �$[(\tau\Sigma)^{-1}]^{-1}$�$[(\tau\Sigma)^{-1} \Pi] = \Pi$ (equil. returns)

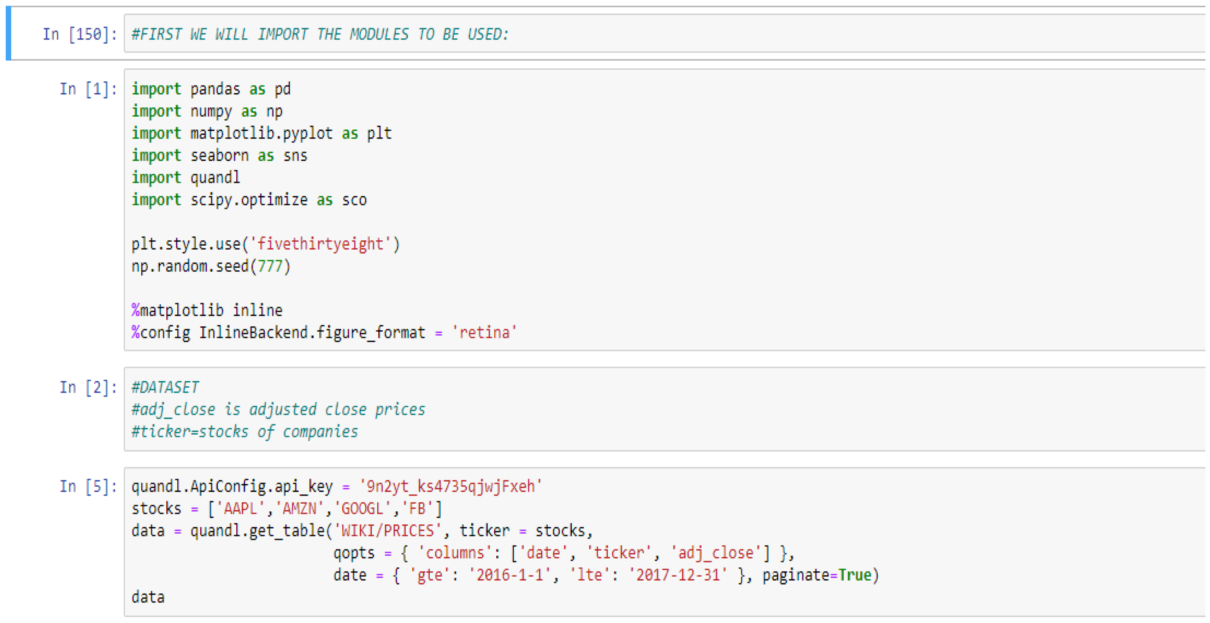
If there is no estimation error, $\Sigma^{-1} \rightarrow \infty$, hence

E(r) ≈ � $[P^T \Omega^{-1} P]^{-1} [P^T \Omega^{-1} Q] = P^{-1} Q$ (view returns)

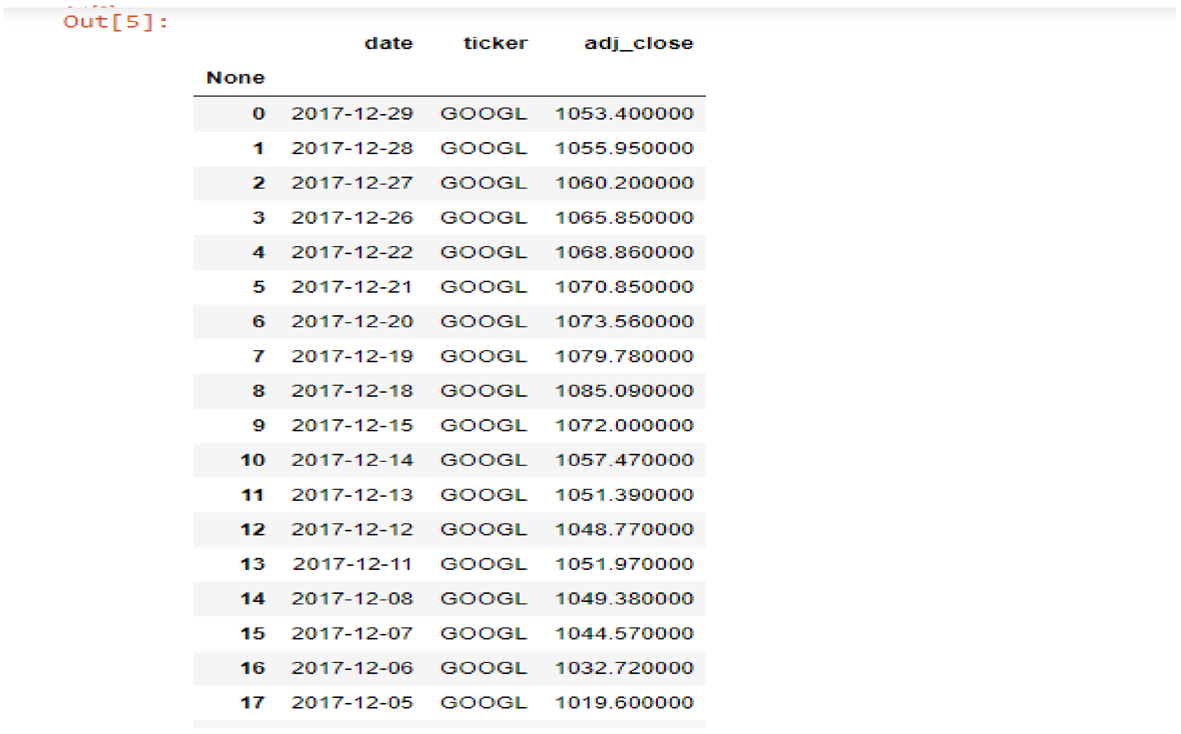
Chapter 3 – CODE

1. MARKOWITZ MODEL

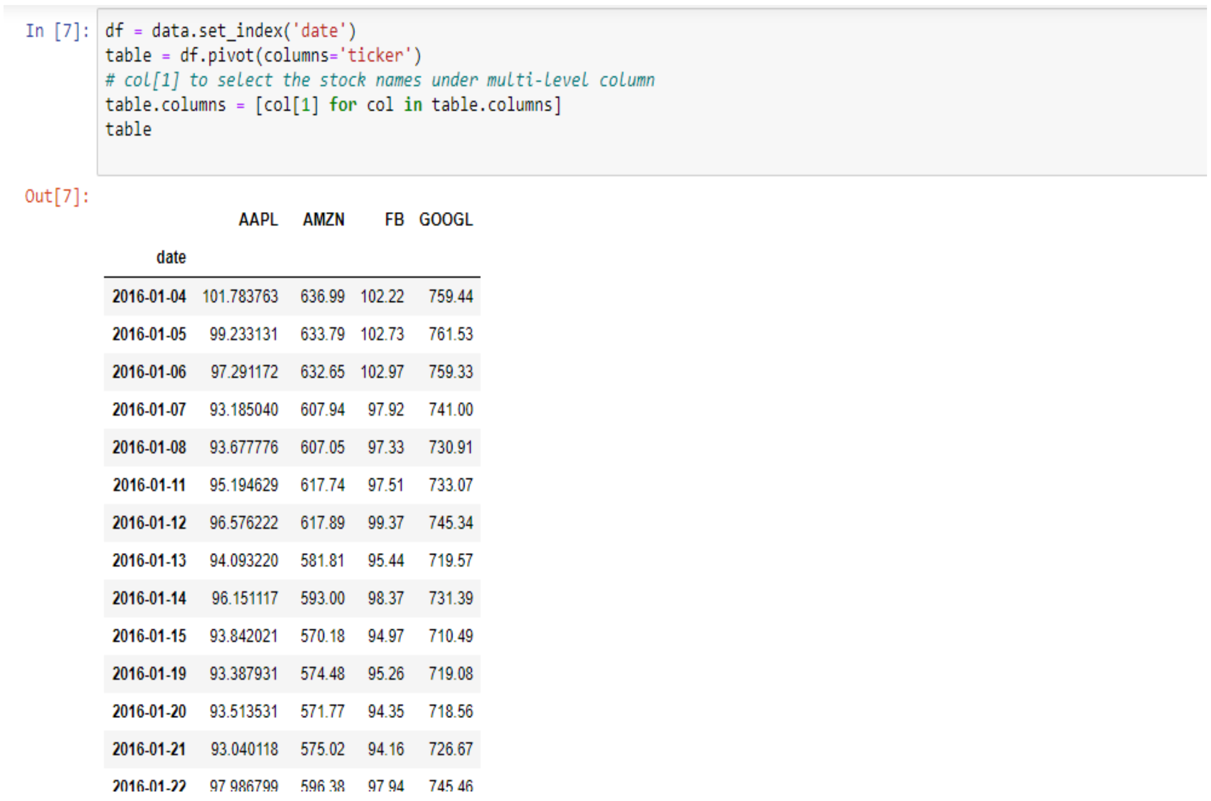
   Importing the modules and obtaining the data set.



```
In [150]: #FIRST WE WILL IMPORT THE MODULES TO BE USED:

In [1]:   import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns
          import quandl
          import scipy.optimize as sco

          plt.style.use('fivethirtyeight')
          np.random.seed(777)

          %matplotlib inline
          %config InlineBackend.figure_format = 'retina'

In [2]:   #DATASET
          #adj_close is adjusted close prices
          #ticker=stocks of companies

In [5]:   quandl.ApiConfig.api_key = '9n2yt_ks4735qjwjFxeh'
          stocks = ['AAPL','AMZN','GOOGL','FB']
          data = quandl.get_table('WIKI/PRICES', ticker = stocks,
                                  qopts = { 'columns': ['date', 'ticker', 'adj_close'] },
                                  date = { 'gte': '2016-1-1', 'lte': '2017-12-31' }, paginate=True)
          data
```



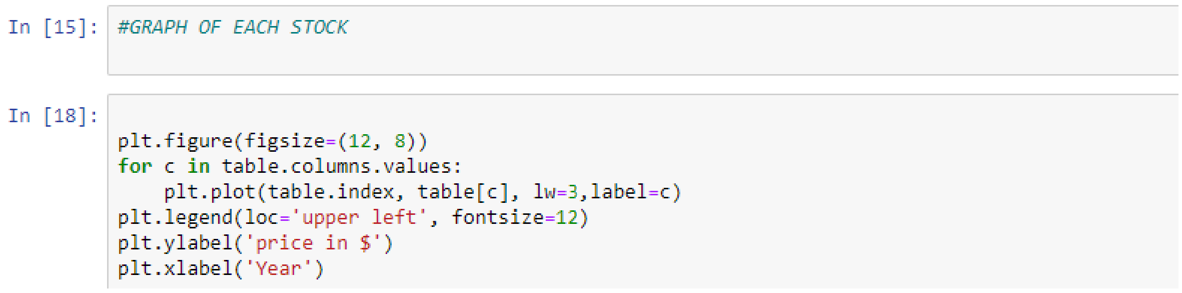
```
Out[5]:
                 date     ticker    adj_close
      None
         0   2017-12-29   GOOGL   1053.400000
         1   2017-12-28   GOOGL   1055.950000
         2   2017-12-27   GOOGL   1060.200000
         3   2017-12-26   GOOGL   1065.850000
         4   2017-12-22   GOOGL   1068.860000
         5   2017-12-21   GOOGL   1070.850000
         6   2017-12-20   GOOGL   1073.560000
         7   2017-12-19   GOOGL   1079.780000
         8   2017-12-18   GOOGL   1085.090000
         9   2017-12-15   GOOGL   1072.000000
        10   2017-12-14   GOOGL   1057.470000
        11   2017-12-13   GOOGL   1051.390000
        12   2017-12-12   GOOGL   1048.770000
        13   2017-12-11   GOOGL   1051.970000
        14   2017-12-08   GOOGL   1049.380000
        15   2017-12-07   GOOGL   1044.570000
        16   2017-12-06   GOOGL   1032.720000
        17   2017-12-05   GOOGL   1019.600000
```



```
In [7]: df = data.set_index('date')
        table = df.pivot(columns='ticker')
        # col[1] to select the stock names under multi-level column
        table.columns = [col[1] for col in table.columns]
        table
```

Out[7]:

date	AAPL	AMZN	FB	GOOGL
2016-01-04	101.783763	636.99	102.22	759.44
2016-01-05	99.233131	633.79	102.73	761.53
2016-01-06	97.291172	632.65	102.97	759.33
2016-01-07	93.185040	607.94	97.92	741.00
2016-01-08	93.677776	607.05	97.33	730.91
2016-01-11	95.194629	617.74	97.51	733.07
2016-01-12	96.576222	617.89	99.37	745.34
2016-01-13	94.093220	581.81	95.44	719.57
2016-01-14	96.151117	593.00	98.37	731.39
2016-01-15	93.842021	570.18	94.97	710.49
2016-01-19	93.387931	574.48	95.26	719.08
2016-01-20	93.513531	571.77	94.35	718.56
2016-01-21	93.040118	575.02	94.16	726.67
2016-01-22	97.986799	596.38	97.94	745.46



```
In [15]: #GRAPH OF EACH STOCK

In [18]: plt.figure(figsize=(12, 8))
         for c in table.columns.values:
             plt.plot(table.index, table[c], lw=3,label=c)
         plt.legend(loc='upper left', fontsize=12)
         plt.ylabel('price in $')
         plt.xlabel('Year')
```

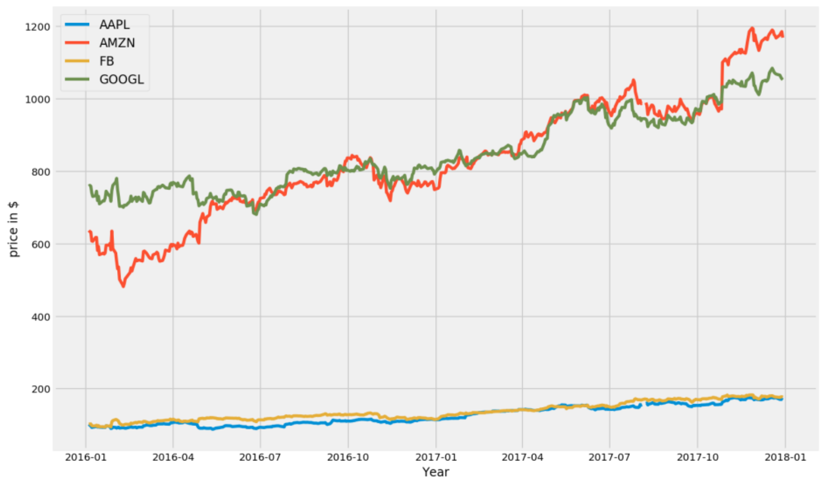


Fig7. Stock Price Variation

```
In [19]: returns1 = table.pct_change()
         plt.figure(figsize=(12, 6))
         for c in returns1.columns.values:
             plt.plot(returns1.index, returns1[c], lw=3,label=c)
         plt.legend(loc='upper right', fontsize=12)
         plt.ylabel('daily returns')

Out[19]: Text(0, 0.5, 'daily returns')
```

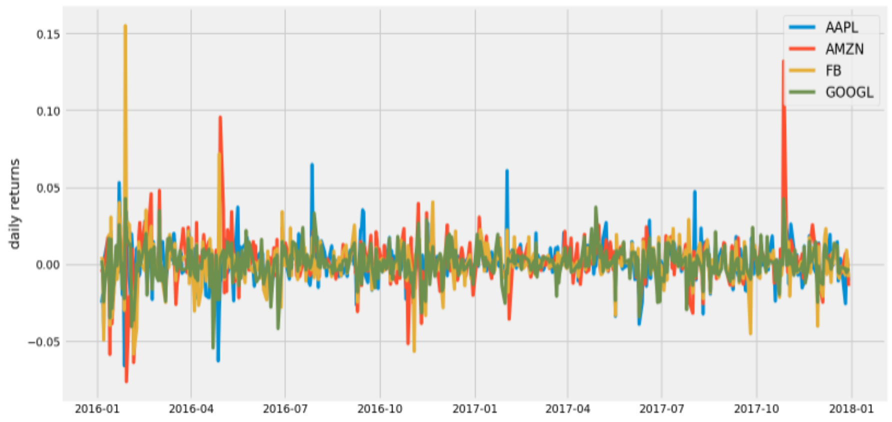
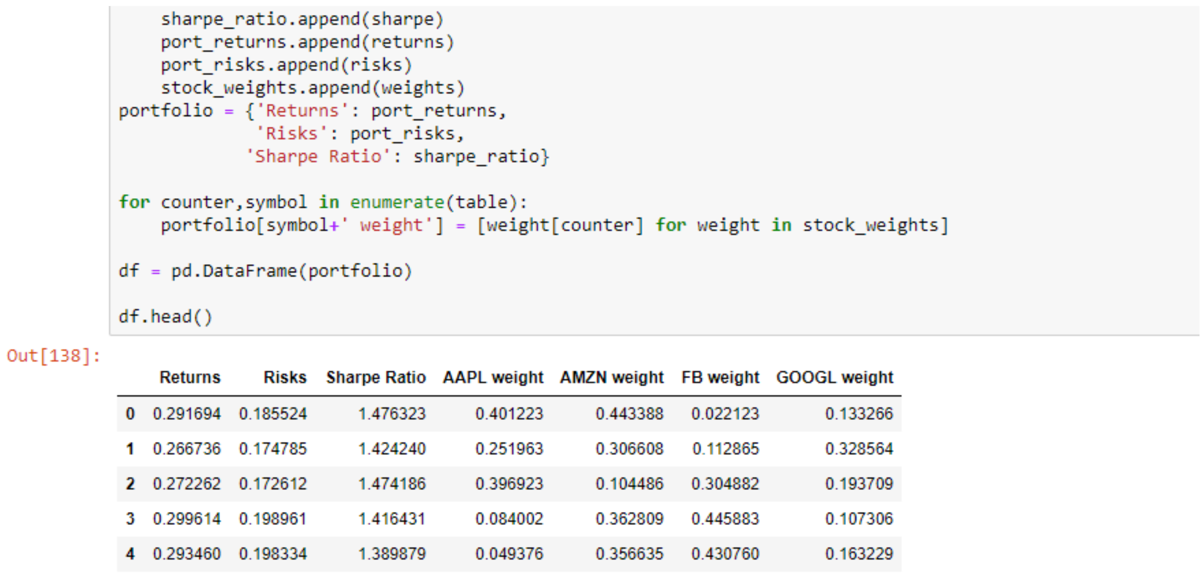


Fig 8.Daily Returns Plot

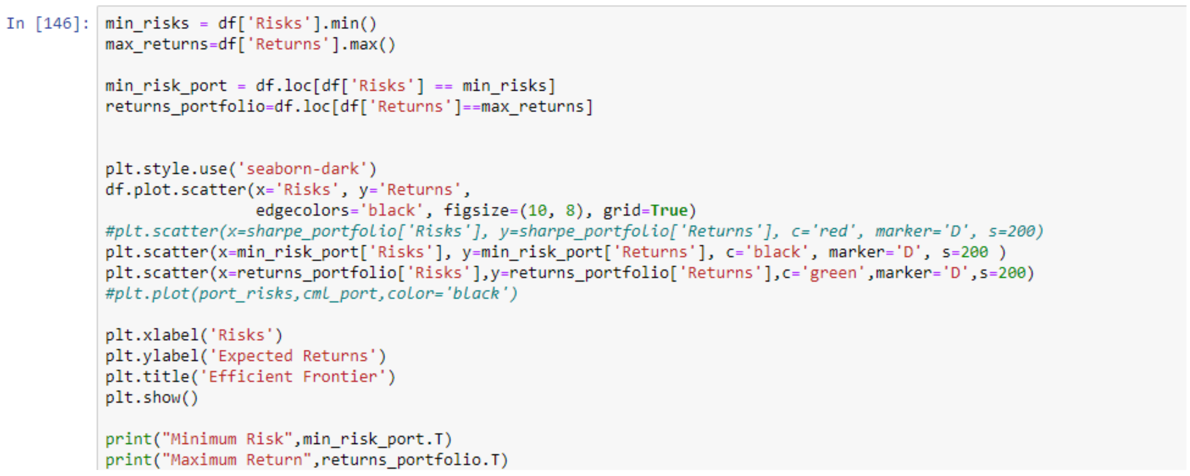
```
In [138]: port_returns = []
          port_risks = []
          stock_weights = []
          sharpe_ratio=[]
          cml_port=[]
          returns1=table.pct_change()
          mean_returns = returns1.mean()
          returns_annual=mean_returns*252
          cov_matrix = returns1.cov()
          cov_annual=cov_matrix*252
          num_assets = 4
          num_portfolios = 25000
          rf=0.0178
          np.random.seed(101)

          for single_portfolio in range(num_portfolios):
              weights = np.random.random(num_assets)
              weights /= np.sum(weights)
              #returns_annual = (mean_returns*weights ) *252
              returns = np.dot(weights, returns_annual)
              sh_returns=(np.dot(weights, returns_annual))-rf
              #cov_annual=np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights))) * np.sqrt(252)
              risks = np.sqrt(np.dot(weights.T, np.dot(cov_annual, weights)))
              cml=1.545374*risks+0.0178
              cml_port.append(cml)
              sharpe = sh_returns / risks
              sharpe_ratio.append(sharpe)
              port_returns.append(returns)
              port_risks.append(risks)
              stock_weights.append(weights)
          portfolio = {'Returns': port_returns,
```



```
    sharpe_ratio.append(sharpe)
    port_returns.append(returns)
    port_risks.append(risks)
    stock_weights.append(weights)
portfolio = {'Returns': port_returns,
             'Risks': port_risks,
            'Sharpe Ratio': sharpe_ratio}

for counter,symbol in enumerate(table):
    portfolio[symbol+' weight'] = [weight[counter] for weight in stock_weights]

df = pd.DataFrame(portfolio)

df.head()
```

Out[138]:

	Returns	Risks	Sharpe Ratio	AAPL weight	AMZN weight	FB weight	GOOGL weight
0	0.291694	0.185524	1.476323	0.401223	0.443388	0.022123	0.133266
1	0.266736	0.174785	1.424240	0.251963	0.306608	0.112865	0.328564
2	0.272262	0.172612	1.474186	0.396923	0.104486	0.304882	0.193709
3	0.299614	0.198961	1.416431	0.084002	0.362809	0.445883	0.107306
4	0.293460	0.198334	1.389879	0.049376	0.356635	0.430760	0.163229



In [146]:

```
min_risks = df['Risks'].min()
max_returns=df['Returns'].max()

min_risk_port = df.loc[df['Risks'] == min_risks]
returns_portfolio=df.loc[df['Returns']==max_returns]


plt.style.use('seaborn-dark')
df.plot.scatter(x='Risks', y='Returns',
                edgecolors='black', figsize=(10, 8), grid=True)
#plt.scatter(x=sharpe_portfolio['Risks'], y=sharpe_portfolio['Returns'], c='red', marker='D', s=200)
plt.scatter(x=min_risk_port['Risks'], y=min_risk_port['Returns'], c='black', marker='D', s=200 )
plt.scatter(x=returns_portfolio['Risks'],y=returns_portfolio['Returns'],c='green',marker='D',s=200)
#plt.plot(port_risks,cml_port,color='black')

plt.xlabel('Risks')
plt.ylabel('Expected Returns')
plt.title('Efficient Frontier')
plt.show()

print("Minimum Risk",min_risk_port.T)
print("Maximum Return",returns_portfolio.T)
```

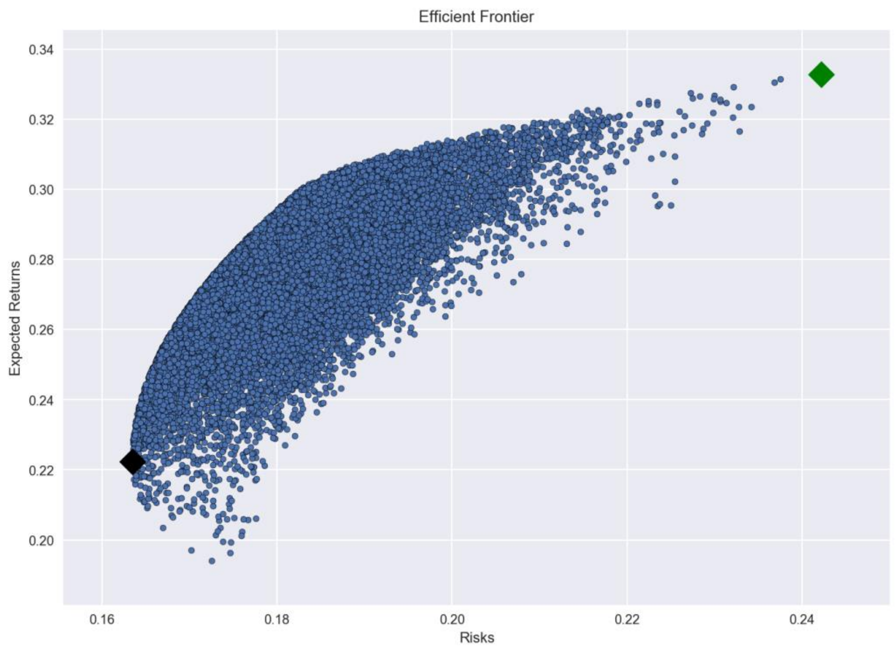


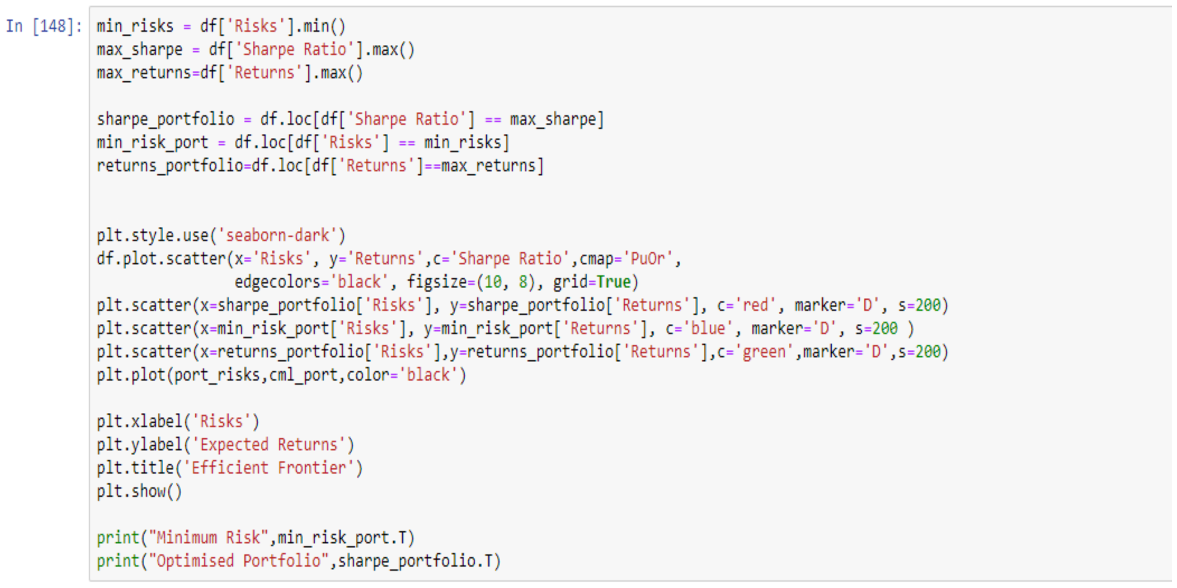
Fig 9. Efficient Frontier curve with minimum risk and maximum return point.

- **Minimum Risk** 22918

- Returns 0.222417
- Risks 0.163472
- Sharpe Ratio 1.251693
- AAPL weight 0.344007
- AMZN weight 0.005949
- FB weight 0.063331
- GOOGL weight 0.586713

- **Maximum Return** 2329

- Returns 0.332948
- Risks 0.242187
- Sharpe Ratio 1.301259
- AAPL weight 0.042407
- AMZN weight 0.913218
- FB weight 0.040182
- GOOGL weight 0.004193



```
In [148]: min_risks = df['Risks'].min()
          max_sharpe = df['Sharpe Ratio'].max()
          max_returns=df['Returns'].max()

          sharpe_portfolio = df.loc[df['Sharpe Ratio'] == max_sharpe]
          min_risk_port = df.loc[df['Risks'] == min_risks]
          returns_portfolio=df.loc[df['Returns']==max_returns]


          plt.style.use('seaborn-dark')
          df.plot.scatter(x='Risks', y='Returns',c='Sharpe Ratio',cmap='PuOr',
                          edgecolors='black', figsize=(10, 8), grid=True)
          plt.scatter(x=sharpe_portfolio['Risks'], y=sharpe_portfolio['Returns'], c='red', marker='D', s=200)
          plt.scatter(x=min_risk_port['Risks'], y=min_risk_port['Returns'], c='blue', marker='D', s=200 )
          plt.scatter(x=returns_portfolio['Risks'],y=returns_portfolio['Returns'],c='green',marker='D',s=200)
          plt.plot(port_risks,cml_port,color='black')

          plt.xlabel('Risks')
          plt.ylabel('Expected Returns')
          plt.title('Efficient Frontier')
          plt.show()

          print("Minimum Risk",min_risk_port.T)
          print("Optimised Portfolio",sharpe_portfolio.T)
```

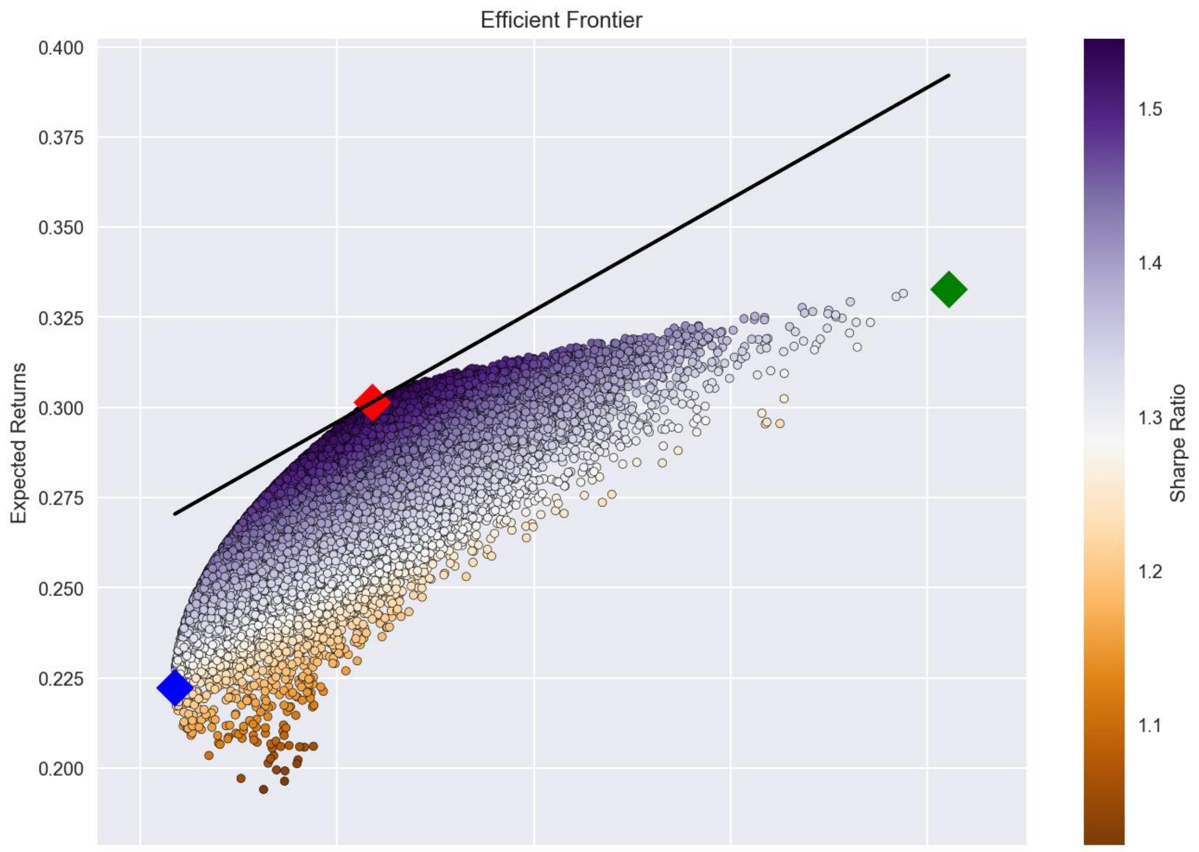
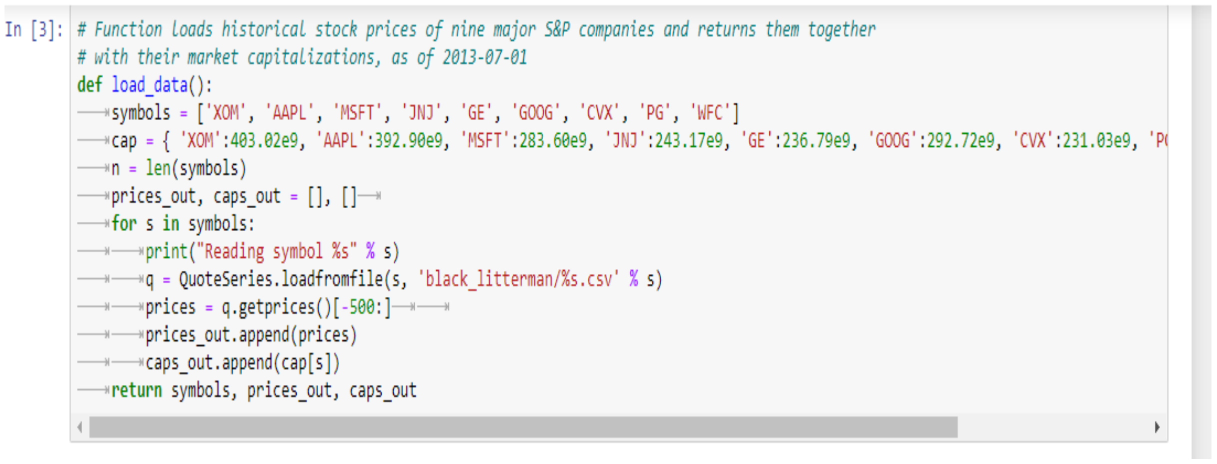


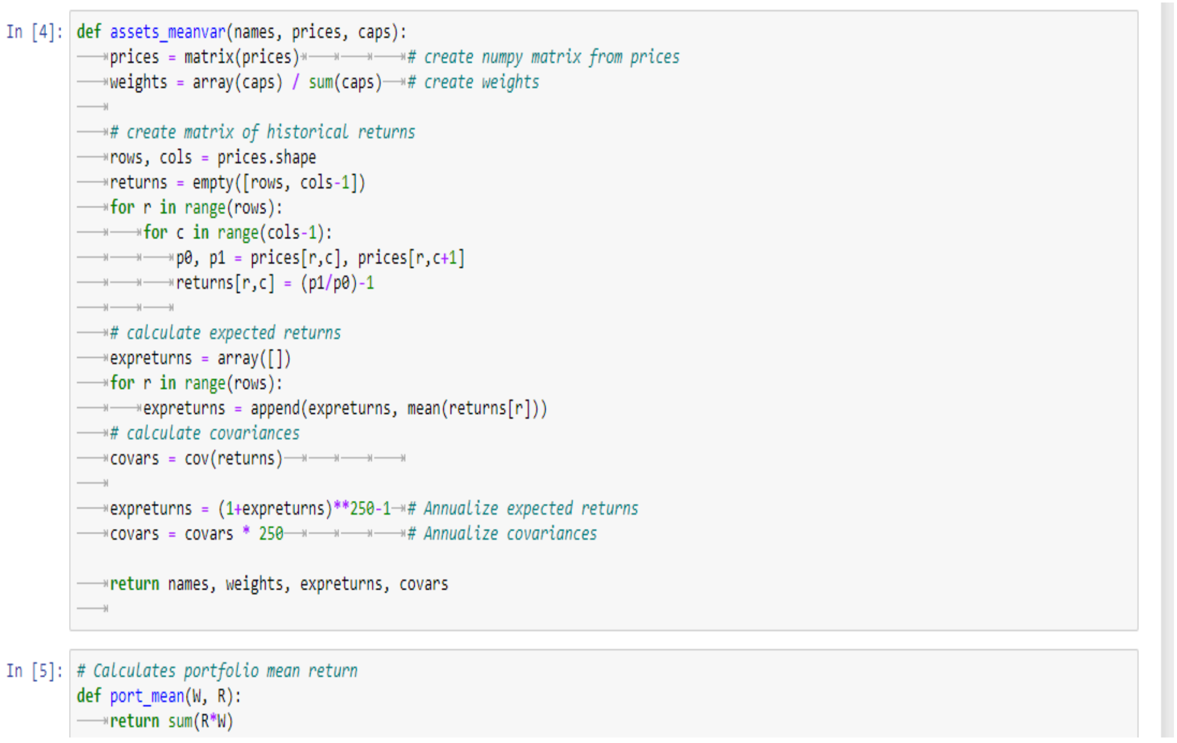
Fig 10.Curve with the sharpe's ratio portfolio(red point).

- Optimised Portfolio 11544
- Returns 0.301506
- Risks 0.183584
- Sharpe Ratio 1.545374
- AAPL weight 0.435712
- AMZN weight 0.295533
- FB weight 0.267486
- GOOGL weight 0.001269

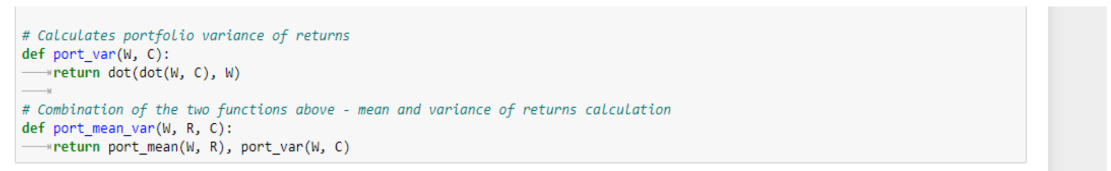
2.BLACK LITTERMAN MODEL



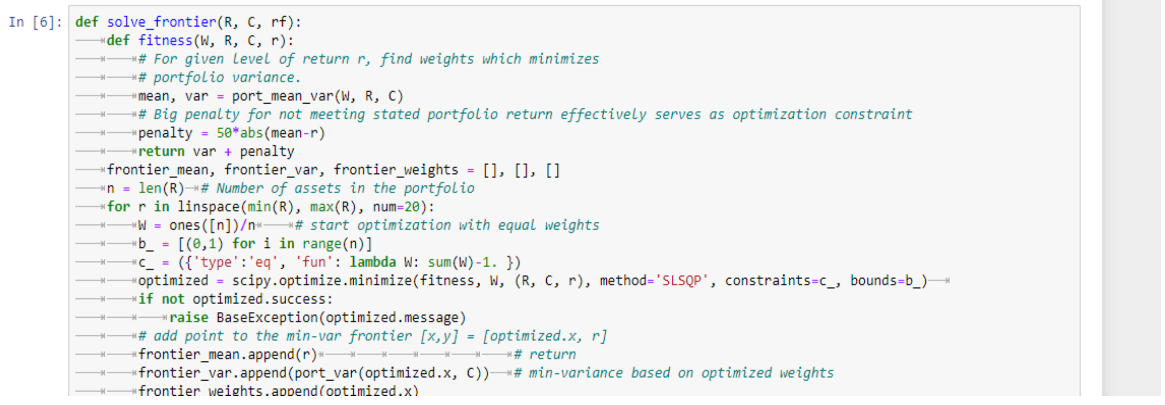
```
In [3]: # Function loads historical stock prices of nine major S&P companies and returns them together
        # with their market capitalizations, as of 2013-07-01
        def load_data():
            symbols = ['XOM', 'AAPL', 'MSFT', 'JNJ', 'GE', 'GOOG', 'CVX', 'PG', 'WFC']
            cap = { 'XOM':403.02e9, 'AAPL':392.90e9, 'MSFT':283.60e9, 'JNJ':243.17e9, 'GE':236.79e9, 'GOOG':292.72e9, 'CVX':231.03e9, 'P
            n = len(symbols)
            prices_out, caps_out = [], []
            for s in symbols:
                print("Reading symbol %s" % s)
                q = QuoteSeries.loadfromfile(s, 'black_litterman/%s.csv' % s)
                prices = q.getprices()[-500:]
                prices_out.append(prices)
                caps_out.append(cap[s])
            return symbols, prices_out, caps_out
```



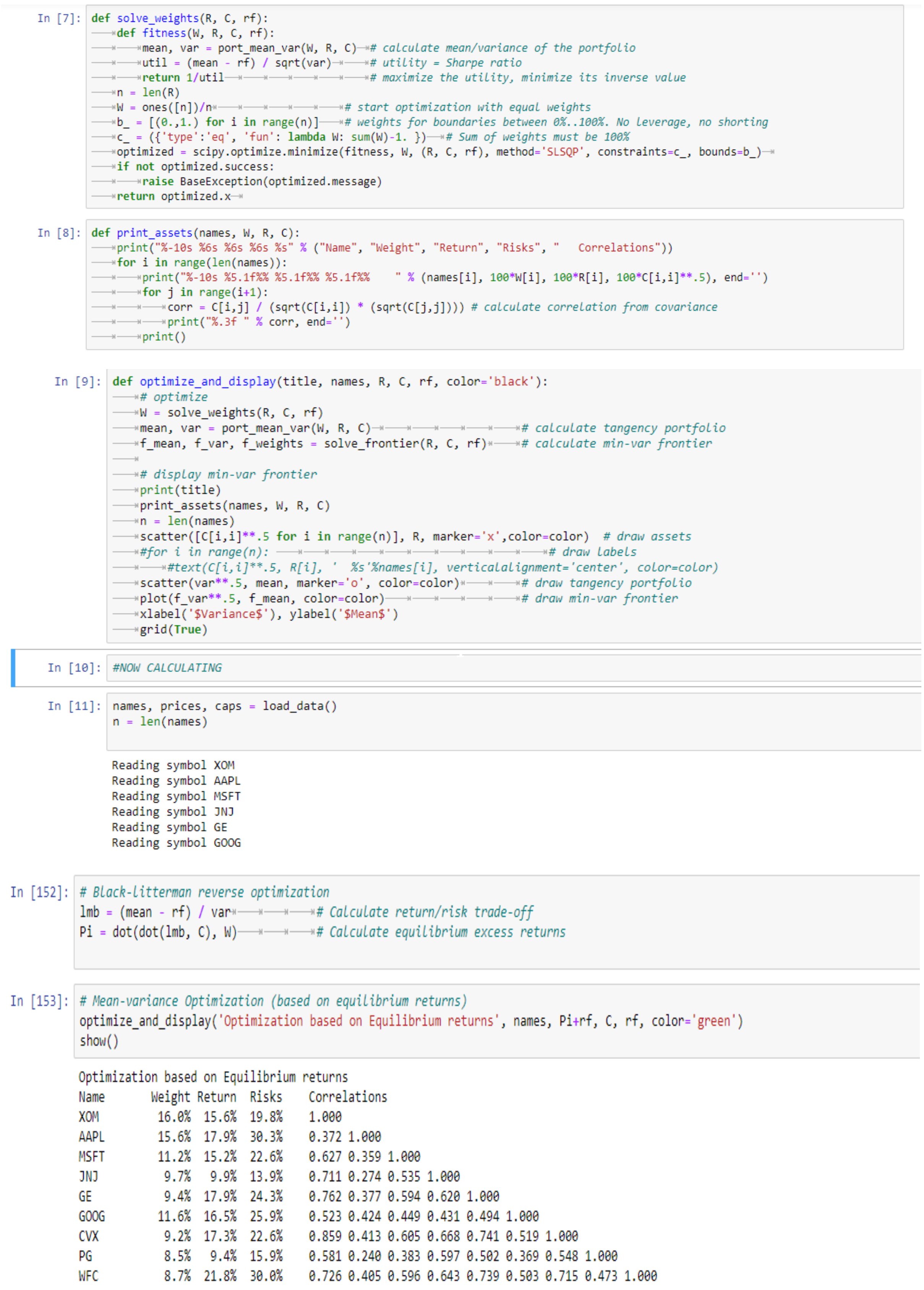
```
In [4]: def assets_meanvar(names, prices, caps):
            prices = matrix(prices)                  # create numpy matrix from prices
            weights = array(caps) / sum(caps)        # create weights

            # create matrix of historical returns
            rows, cols = prices.shape
            returns = empty([rows, cols-1])
            for r in range(rows):
                for c in range(cols-1):
                    p0, p1 = prices[r,c], prices[r,c+1]
                    returns[r,c] = (p1/p0)-1

            # calculate expected returns
            expreturns = array([])
            for r in range(rows):
                expreturns = append(expreturns, mean(returns[r]))
            # calculate covariances
            covars = cov(returns)

            expreturns = (1+expreturns)**250-1   # Annualize expected returns
            covars = covars * 250                # Annualize covariances

            return names, weights, expreturns, covars

In [5]: # Calculates portfolio mean return
        def port_mean(W, R):
            return sum(R*W)
```



```
# Calculates portfolio variance of returns
def port_var(W, C):
    return dot(dot(W, C), W)

# Combination of the two functions above - mean and variance of returns calculation
def port_mean_var(W, R, C):
    return port_mean(W, R), port_var(W, C)
```



```
In [6]: def solve_frontier(R, C, rf):
            def fitness(W, R, C, r):
                # For given level of return r, find weights which minimizes
                # portfolio variance.
                mean, var = port_mean_var(W, R, C)
                # Big penalty for not meeting stated portfolio return effectively serves as optimization constraint
                penalty = 50*abs(mean-r)
                return var + penalty
            frontier_mean, frontier_var, frontier_weights = [], [], []
            n = len(R)  # Number of assets in the portfolio
            for r in linspace(min(R), max(R), num=20):
                W = ones([n])/n      # start optimization with equal weights
                b_ = [(0,1) for i in range(n)]
                c_ = ({'type':'eq', 'fun': lambda W: sum(W)-1. })
                optimized = scipy.optimize.minimize(fitness, W, (R, C, r), method='SLSQP', constraints=c_, bounds=b_)
                if not optimized.success:
                    raise BaseException(optimized.message)
                # add point to the min-var frontier [x,y] = [optimized.x, r]
                frontier_mean.append(r)                          # return
                frontier_var.append(port_var(optimized.x, C))    # min-variance based on optimized weights
                frontier_weights.append(optimized.x)
```



```
In [7]: def solve_weights(R, C, rf):
            def fitness(W, R, C, rf):
                mean, var = port_mean_var(W, R, C)  # calculate mean/variance of the portfolio
                util = (mean - rf) / sqrt(var)      # utility = Sharpe ratio
                return 1/util                       # maximize the utility, minimize its inverse value
            n = len(R)
            W = ones([n])/n                         # start optimization with equal weights
            b_ = [(0.,1.) for i in range(n)]        # weights for boundaries between 0%..100%. No leverage, no shorting
            c_ = ({'type':'eq', 'fun': lambda W: sum(W)-1. })  # Sum of weights must be 100%
            optimized = scipy.optimize.minimize(fitness, W, (R, C, rf), method='SLSQP', constraints=c_, bounds=b_)
            if not optimized.success:
                raise BaseException(optimized.message)
            return optimized.x
```

```
In [8]: def print_assets(names, W, R, C):
            print("%-10s %6s %6s %6s %s" % ("Name", "Weight", "Return", "Risks", "   Correlations"))
            for i in range(len(names)):
                print("%-10s %5.1f%% %5.1f%% %5.1f%%    " % (names[i], 100*W[i], 100*R[i], 100*C[i,i]**.5), end='')
                for j in range(i+1):
                    corr = C[i,j] / (sqrt(C[i,i]) * (sqrt(C[j,j]))) # calculate correlation from covariance
                    print("%.3f " % corr, end='')
                print()
```

```
In [9]: def optimize_and_display(title, names, R, C, rf, color='black'):
            # optimize
            W = solve_weights(R, C, rf)
            mean, var = port_mean_var(W, R, C)                  # calculate tangency portfolio
            f_mean, f_var, f_weights = solve_frontier(R, C, rf)  # calculate min-var frontier

            # display min-var frontier
            print(title)
            print_assets(names, W, R, C)
            n = len(names)
            scatter([C[i,i]**.5 for i in range(n)], R, marker='x',color=color)  # draw assets
            #for i in range(n):                                  # draw labels
                #text(C[i,i]**.5, R[i], '  %s'%names[i], verticalalignment='center', color=color)
            scatter(var**.5, mean, marker='o', color=color)     # draw tangency portfolio
            plot(f_var**.5, f_mean, color=color)                # draw min-var frontier
            xlabel('$Variance$'), ylabel('$Mean$')
            grid(True)
```

```
In [10]: #NOW CALCULATING
```

```
In [11]: names, prices, caps = load_data()
         n = len(names)

         Reading symbol XOM
         Reading symbol AAPL
         Reading symbol MSFT
         Reading symbol JNJ
         Reading symbol GE
         Reading symbol GOOG
```

```
In [152]: # Black-litterman reverse optimization
          lmb = (mean - rf) / var      # Calculate return/risk trade-off
          Pi = dot(dot(lmb, C), W)     # Calculate equilibrium excess returns
```

```
In [153]: # Mean-variance Optimization (based on equilibrium returns)
          optimize_and_display('Optimization based on Equilibrium returns', names, Pi+rf, C, rf, color='green')
          show()

          Optimization based on Equilibrium returns
          Name       Weight Return  Risks   Correlations
          XOM         16.0%  15.6%  19.8%   1.000
          AAPL        15.6%  17.9%  30.3%   0.372 1.000
          MSFT        11.2%  15.2%  22.6%   0.627 0.359 1.000
          JNJ          9.7%   9.9%  13.9%   0.711 0.274 0.535 1.000
          GE           9.4%  17.9%  24.3%   0.762 0.377 0.594 0.620 1.000
          GOOG        11.6%  16.5%  25.9%   0.523 0.424 0.449 0.431 0.494 1.000
          CVX          9.2%  17.3%  22.6%   0.859 0.413 0.605 0.668 0.741 0.519 1.000
          PG           8.5%   9.4%  15.9%   0.581 0.240 0.383 0.597 0.502 0.369 0.548 1.000
          WFC          8.7%  21.8%  30.0%   0.726 0.405 0.596 0.643 0.739 0.503 0.715 0.473 1.000
```

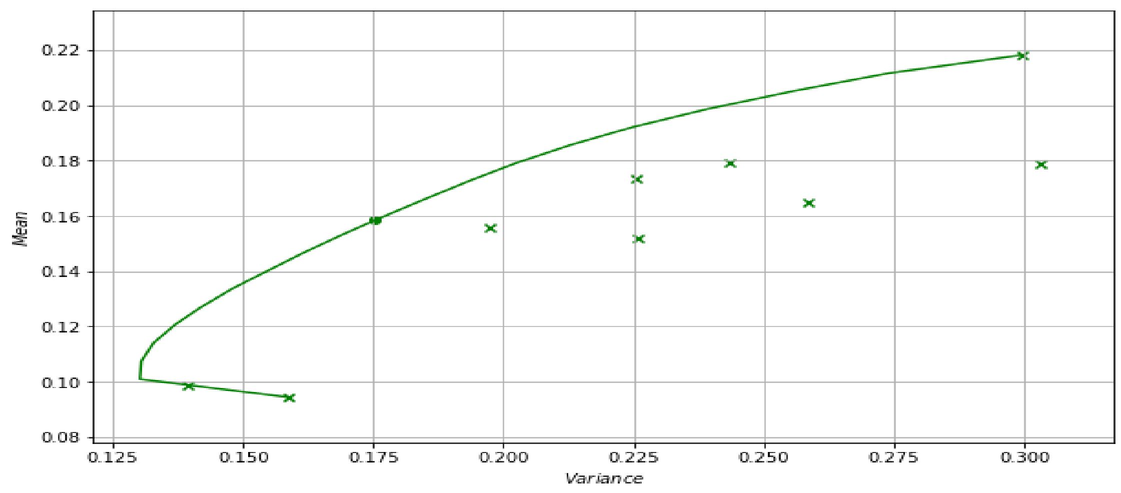
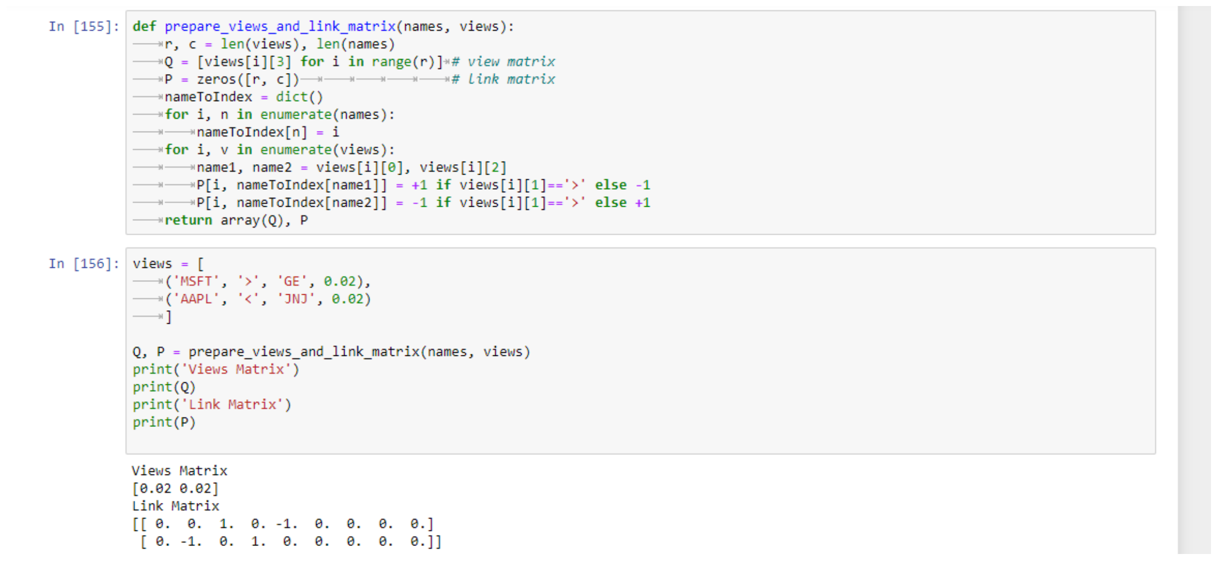
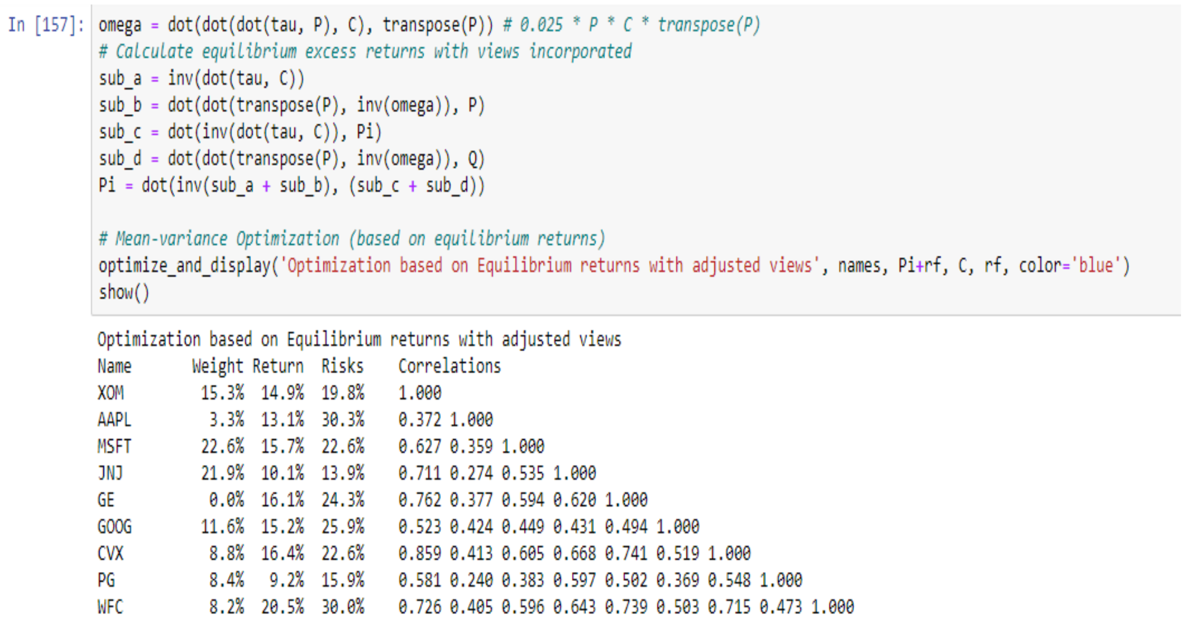


Fig 11.Optimization Based on Equilibrium Returns



```
In [155]: def prepare_views_and_link_matrix(names, views):
              r, c = len(views), len(names)
              Q = [views[i][3] for i in range(r)] # view matrix
              P = zeros([r, c])                   # link matrix
              nameToIndex = dict()
              for i, n in enumerate(names):
                  nameToIndex[n] = i
              for i, v in enumerate(views):
                  name1, name2 = views[i][0], views[i][2]
                  P[i, nameToIndex[name1]] = +1 if views[i][1]=='>' else -1
                  P[i, nameToIndex[name2]] = -1 if views[i][1]=='>' else +1
              return array(Q), P

In [156]: views = [
              ('MSFT', '>', 'GE', 0.02),
              ('AAPL', '<', 'JNJ', 0.02)
              ]

          Q, P = prepare_views_and_link_matrix(names, views)
          print('Views Matrix')
          print(Q)
          print('Link Matrix')
          print(P)

          Views Matrix
          [0.02 0.02]
          Link Matrix
          [[ 0.  0.  1.  0. -1.  0.  0.  0.  0.]
           [ 0. -1.  0.  1.  0.  0.  0.  0.  0.]]
```



```
In [157]: omega = dot(dot(dot(tau, P), C), transpose(P)) # 0.025 * P * C * transpose(P)
          # Calculate equilibrium excess returns with views incorporated
          sub_a = inv(dot(tau, C))
          sub_b = dot(dot(transpose(P), inv(omega)), P)
          sub_c = dot(inv(dot(tau, C)), Pi)
          sub_d = dot(dot(transpose(P), inv(omega)), Q)
          Pi = dot(inv(sub_a + sub_b), (sub_c + sub_d))

          # Mean-variance Optimization (based on equilibrium returns)
          optimize_and_display('Optimization based on Equilibrium returns with adjusted views', names, Pi+rf, C, rf, color='blue')
          show()

          Optimization based on Equilibrium returns with adjusted views
          Name        Weight Return  Risks    Correlations
          XOM          15.3%  14.9%  19.8%    1.000
          AAPL          3.3%  13.1%  30.3%    0.372 1.000
          MSFT         22.6%  15.7%  22.6%    0.627 0.359 1.000
          JNJ          21.9%  10.1%  13.9%    0.711 0.274 0.535 1.000
          GE            0.0%  16.1%  24.3%    0.762 0.377 0.594 0.620 1.000
          GOOG         11.6%  15.2%  25.9%    0.523 0.424 0.449 0.431 0.494 1.000
          CVX           8.8%  16.4%  22.6%    0.859 0.413 0.605 0.668 0.741 0.519 1.000
          PG            8.4%   9.2%  15.9%    0.581 0.240 0.383 0.597 0.502 0.369 0.548 1.000
          WFC           8.2%  20.5%  30.0%    0.726 0.405 0.596 0.643 0.739 0.503 0.715 0.473 1.000
```

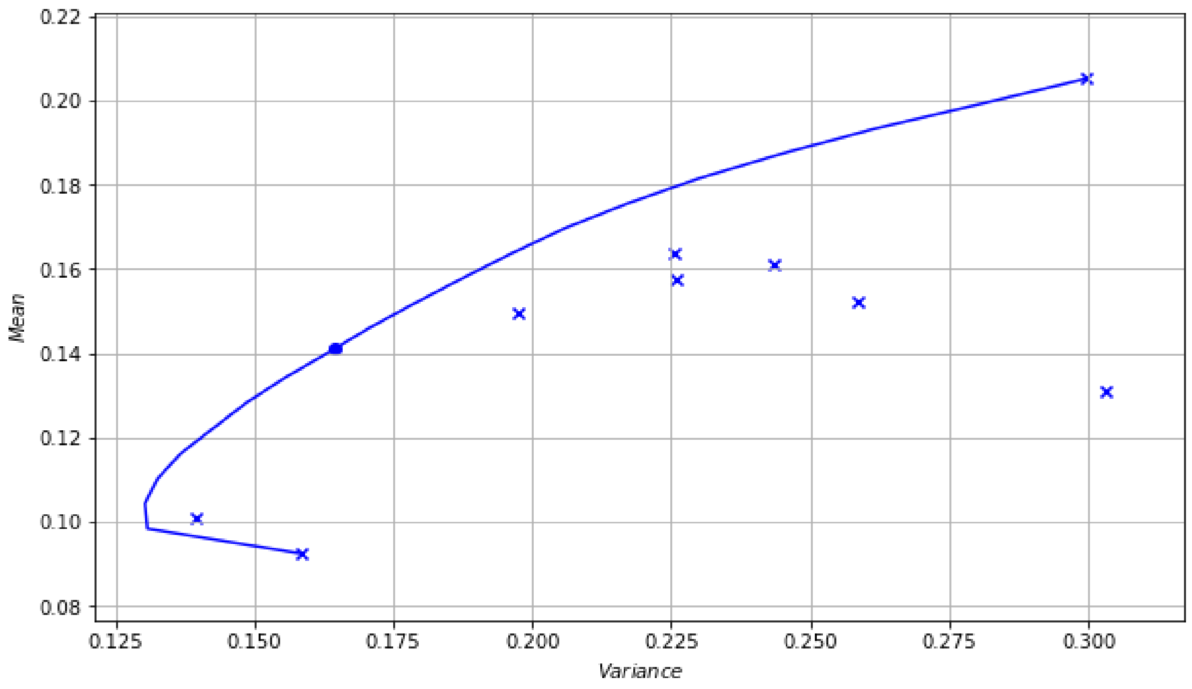
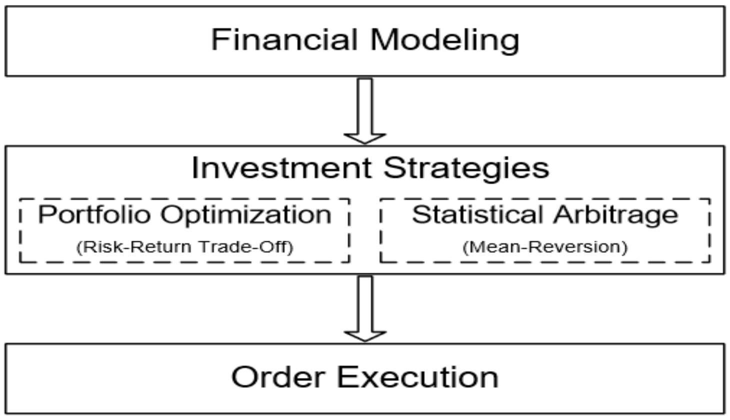


Fig 12. Optimization based on equilibrium return with adjusted views

Chapter 4 – FUTURE SCOPE

Both the models can be used by investors before investing in markets. Modern Portfolio Theory is already in use by many, but as we can see the Black Litterman is a better choice, so shifting of people from first model to later will also be done.

Furthermore, the other models of optimization like Robust Portfolio Optimization will be implemented. And after implementing Financial Modeling and Investment Strategies (Portfolio optimization) , order execution will be implemented.



Chapter 5 – CONCLUSION

This report details the process to develop the investment strategies for investors. Earlier, people used to invest in one asset a time but the introduction of Markowitz model brings the idea of diversification which proofs the increase in return when invested in different assets at a time as the return depends on the covariances of the asset. Later, it was found out that Markowitz model have some limitations like lack of robustness and inappropriate variance, after which Black Litterman model was introduced which enables investors to combine their unique views with the Implied Equilibrium Return Vector to form a New Combined Return Vector. The New Combined Return Vector leads to intuitive, well-diversified portfolios. The two parameters of the Black-Litterman model that control the relative importance placed on the equilibrium returns vs. the view returns, the scalar (τ) and the uncertainty in the views (Ω), are very difficult to specify. The Black-Litterman formula with 100% certainty in the views enables one to determine the implied confidence in a view. Using this implied confidence framework, a new method for controlling the tilts and the final portfolio weights caused by the views is introduced. Overall, the Black-Litterman model overcomes the most-often cited weaknesses of mean-variance optimization (unintuitive, highly concentrated portfolios, input-sensitivity, and estimation error-maximization) helping users to realize the benefits of the Markowitz paradigm. Likewise, the proposed new method for incorporating user-specified confidence levels should increase the intuitiveness and the usability of the Black-Litterman model.

RFERENCES

- A signal processing perspective on financial engineering - Yiyong Feng .
- H.M. Markowitz, portfolio selection: efficient diversification of investments, john Wiley, new york (1959).
- A STEP-BY-STEP GUIDE TO THE BLACK-LITTERMAN MODEL, Thomas M. Idzorek*
- An investigation into the Black-Litterman model Author: Martin Felix Jørgensen Supervisor: Niels Henrik Lehde Pedersen
- R. Mansini, W. Ogryczak, M.G. Speranza, portfolio optimization and transaction costs. In quantitative financial risk management: theory and practice, oxford: wiley (2015), 212-241.
- Application of Markowitz portfolio optimization on bulgarian stock market from 2013 to 2016 M. Ivanova , L. Dospatliev.
- Udemy- Python for Data Science and Machine Learning Bootcamp.
- W.-P. Chen, H. Chung, K.-Y. Ho, T.-L. Hsu, Portfolio optimization models and meanvariance spanning tests, In: Handbook of Quantitative Finance and Risk Management, Cheng-Few Lee, Alice C. Lee, John Lee, eds., Springer (2010), 165-184.
- H. Konno, A. Suzuki, A mean variance skewness optimization model, Journal of the Operations Research Society of Japan, 38 (1995), 173-187.
- Best, M.J., and Grauer, R.R. (1991). "On the Sensitivity of Mean-Variance-Efficient Portfolios to Changes in Asset Means: Some Analytical and Computational Results." The Review of Financial Studies, January, 315-342.
- Bevan, A., and Winkelmann, K. (1998). "Using the Black-Litterman Global Asset Allocation Model: Three Years of Practical Experience." Fixed Income Research, Goldman, Sachs & Company, December.