FE-630 Portfolio Theory and Application

Guanzhuo Qiao (10442266)

You Yu (10443241)

Yu Peng (10438088)

Heng Lu (10434211)

Notations & Conclusion

Notations

We use the ETFs' data from Yahoo Finance and the factor data from Ken French's website. After simple modification, we divide this data set into 3 time periods.

Time Period:

Before the crisis: 2007-03-19 to 2008-02-29

During the crisis: 2008-03-03 to 2010-08-31

After the crisis: 2010-09-01 to 2015-01-02

The whole period: 2007-03-19 to 2017-07-24

The S&P 500 ETF is used as the benchmark and market portfolio (r_m). The risk free rate(r_f) is selected from the factor data set.

We also select several time period term for estimating expected return and covariance matrix. Short term: 60 days, Middle term: 120 days and Long term: 180 days. We denote this by using

$$RetCov_{cov\_term}^{ret\_term}$$

Say, $RetCov_{120}^{60}$ means that the expected returns are estimated by using 60 days data and covariance matrix is estimated by using 120 days data. In this report we try different combinations: $RetCov_{60}^{60},\ RetCov_{120}^{60},\ RetCov_{180}^{120}$.

Another notation we need is the β_T^m which is what we constrain the portfolios' target beta. In this report we try different number: $\beta_T^m = 0.5,\ \beta_T^m = 1,\ \beta_T^m = 2$.

Conclusion

In this section, we present the overall returns and indicators of each portfolio that in different time periods using different combinations of estimation term and portfolio's target beta.

Portfolio Indicators Table (before the crisis)

	$RetCov_{60}^{60}$			$RetCov_{120}^{60}$			$RetCov_{180}^{120}$			SPY
	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	
Annual_cumulatedRet	8.49767	7.53732	6.35036	6.64957	6.41836	5.83755	4.57705	4.29485	3.66879	0.964266
Annual_ariMeanRet	2.49593	2.37372	2.2242	2.152187	2.119774	2.049432	1.662366	1.593047	1.428803	-0.02141
Annual_geoMeanRet	2.148975	2.028048	1.855362	1.901749	1.866093	1.770551	1.525691	1.461673	1.303247	-0.03639
Annual_MinRet	-52.2832	-55.1376	-60.8486	-37.3102	-40.5511	-50.4096	-31.672	-28.7419	-25.8554	-7.40836
Max_10day_Drawdown	-0.2716	-0.25989	-0.28997	-0.25756	-0.24312	-0.27033	-0.14598	-0.127	-0.13283	-0.07481
vol	0.836022	0.833922	0.85948	0.713462	0.717184	0.748759	0.531427	0.521172	0.50886	0.172852
Annual_SharpeRatio	2.913715	2.774506	2.518033	2.932445	2.872028	2.656971	3.015213	2.941541	2.68994	-0.471
Annual_Kurtosis	1.358219	1.590121	2.068032	2.414585	2.590431	3.153887	7.15822	7.117503	6.762789	0.461433
Annual_Skew	0.081186	0.068168	-0.00783	0.417287	0.347838	0.112533	1.382195	1.435796	1.364698	-0.26888
Annual_mVaR	-117.496	-117.856	-123.909	-91.5696	-93.5563	-103.02	-46.4347	-44.875	-46.1986	-29.7037
Daily_VaR	-7.69874	-7.72577	-8.05146	-6.56124	-6.61294	-6.96955	-4.86347	-4.78451	-4.72213	-1.80673
Annual_VaR	-121.728	-122.155	-127.305	-103.742	-104.56	-110.198	-76.8982	-75.6498	-74.6635	-28.567
Annual_CVaR	-165.166	-162.083	-172.014	-147.014	-150.341	-163.532	-96.5118	-96.2142	-101.696	-40.1794

We make comparisons with portfolios under the same period of estimation between different β_T^m and portfolios under the same β_T^m between different periods.

As is shown, we conclude that *before the crisis*:

1. Under the same period of estimation, the portfolio with larger β_T^m tends to have a lower return but a larger VaR. Take $RetCov_{60}^{60}$ as example,

	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$
Annual_cumulatedRet	8.49767	7.53732	6.35036
Annual_VaR	-121.728	-122.155	-127.305

2. Under the same β_T^m, the portfolio with longer period of estimation tends to have a lower return and a smaller VaR. Take $\beta_T^m = 0.5$ as example,

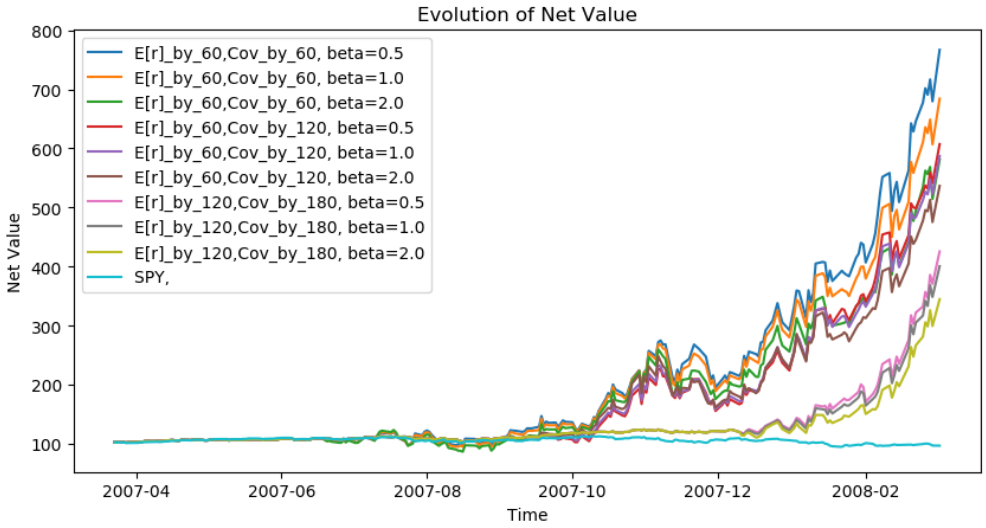
	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$
Annual_cumulatedRet	8.49767	6.64957	4.57705
Annual_VaR	-121.728	-103.742	-76.8982

3. Compared with the benchmark SPY, our portfolios infest a better result. Take $\beta_T^m = 1$ as example,

	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$	SPY
Annual_cumulatedRet	7.53732	6.41836	4.29485	0.964266
Annual_VaR	-121.728	-103.742	-76.8982	-28.567
Annual_SharpeRatio	2.913715	2.932445	3.015213	-0.471

Our portfolios have a higher return and larger risk than the benchmark. But when we compare the Sharpe Ratio, we can see that each of our portfolio has a better chance to make profit than the benchmark.

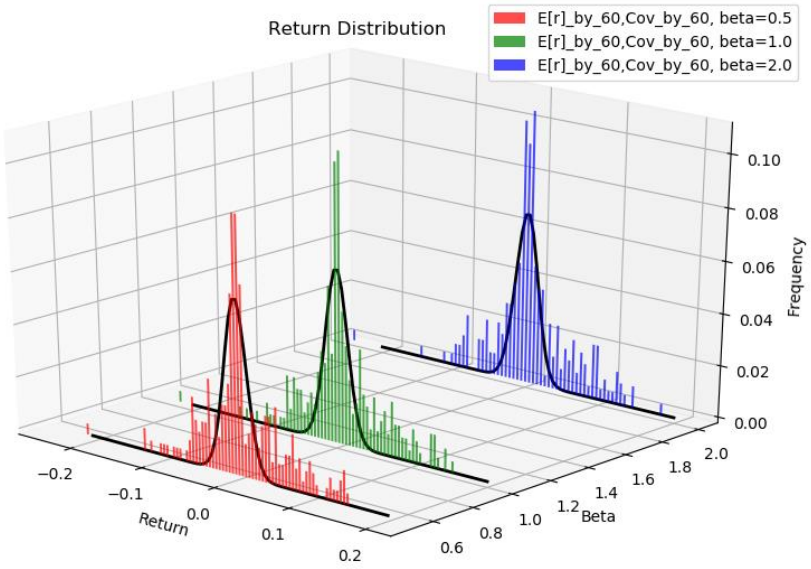
The following chart shows the curves of the PnL between different portfolios *before the crisis*.

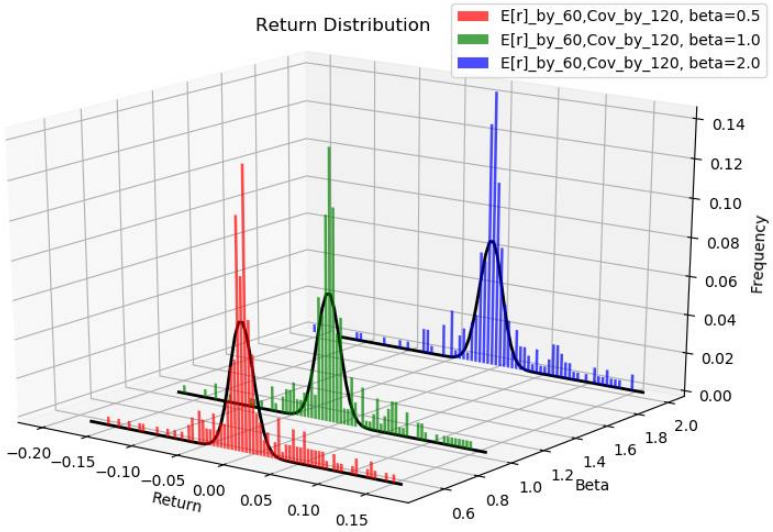


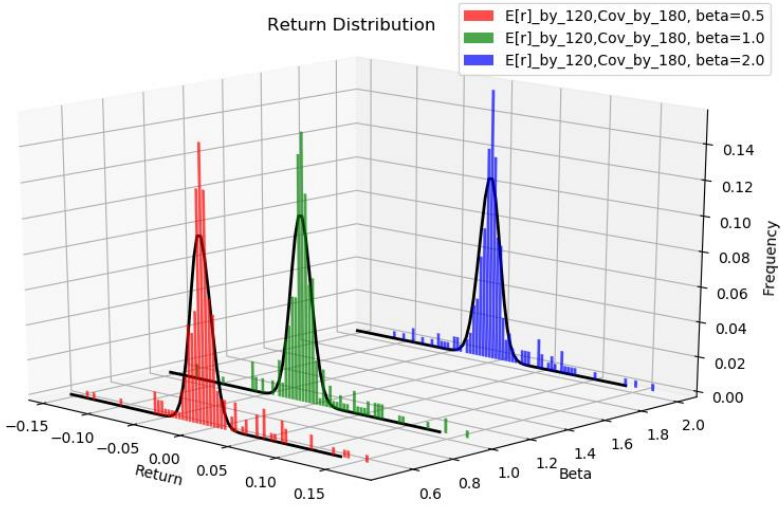
As a result, the portfolios with shorter term of estimation shows a greater profit compared to longer ones.

Note that there is a long time of horizontal lines at the beginning of the chart. This is because we started to rebalance our position after such a long time.

The following three charts are the portfolios' daily return distribution plots.







From these charts we can conclude that the longer term of the estimation is, the distribution is more likely to cluster to its mean, which means they have a lower volatility but a higher kurtosis.

Portfolio Indicators Table (during the crisis)

	$RetCov_{60}^{60}$			$RetCov_{120}^{60}$			$RetCov_{180}^{120}$			SPY
	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	
Annual_cumulatedRet	0.241091	0.241	0.24085	0.458652	0.461037	0.046352	0.446242	0.464995	0.494301	0.940682
Annual_ariMeanRet	-0.7649	-0.73785	-0.62847	-0.25461	-0.22896	-0.12961	-0.38129	-0.33435	-0.21878	-0.00939
Annual_geoMeanRet	-1.41854	-1.41892	-1.41954	-0.77825	-0.77308	-0.76773	-0.80559	-0.76456	-0.70362	-0.06114
Annual_MinRet	-76.6499	-82.5629	-93.986	-71.6003	-69.9295	-79.7328	-65.5965	-67.5284	-75.2781	-24.612
Max_10day_Drawdown	-0.60382	-0.62496	-0.66689	-0.53982	-0.53618	-0.56862	-0.5943	-0.6049	-0.62734	-0.24948
vol	1.124416	1.147473	1.236218	1.009415	1.029797	1.117068	0.906889	0.913996	0.971728	0.322237
Annual_SharpeRatio	-0.73362	-0.69531	-0.55691	-0.31167	-0.2806	-0.16974	-0.4866	-0.43146	-0.2869	-0.21535
Annual_Kurtosis	1.841495	1.779252	1.800196	2.567646	2.439359	2.564003	3.324095	3.194858	2.918766	7.822855
Annual_Skew	-0.38959	-0.38458	-0.33368	-0.36279	-0.31626	-0.18252	-0.51068	-0.46576	-0.35649	0.394265
Annual_mVaR	-197.741	-201.514	-214.29	-172.574	-174.83	-184.507	-158.218	-158.289	-165.11	-44.27
Daily_VaR	-12.0032	-12.2323	-13.1117	-10.6028	-10.8045	-11.6727	-9.58685	-9.64202	-10.1964	-3.35598
Annual_VaR	-189.788	-193.409	-207.314	-167.644	-170.835	-184.561	-151.581	-152.454	-161.219	-53.0627
Annual_CVaR	-280.681	-284.497	-298.084	-259.375	-261.532	-276.834	-232.296	-232.685	-242.04	-77.6518

We make comparisons with portfolios under the same period of estimation between different β_T^m and portfolios under the same β_T^m between different periods.

As is shown, we conclude that *during the crisis*:

1. Under the same period of estimation, the portfolios with different β_T^m tend to have a similar return expectation. Meanwhile, they share almost the same risk. Take $RetCov_{60}^{60}$ as example,

	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$
Annual_cumulatedRet	0.241091	0.241	0.24085
Annual_VaR	-189.788	-167.644	-151.581

2. Under the same β_T^m, the portfolio with medium and long period of estimation tends to have a higher return than the short one. However, although they behavior differently, they all inevitably suffer great loss during the crisis. Take $\beta_T^m = 0.5$ as example,

	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$
Annual_cumulatedRet	0.241091	0.458652	0.446242
Annual_VaR	-189.788	-167.644	-151.581

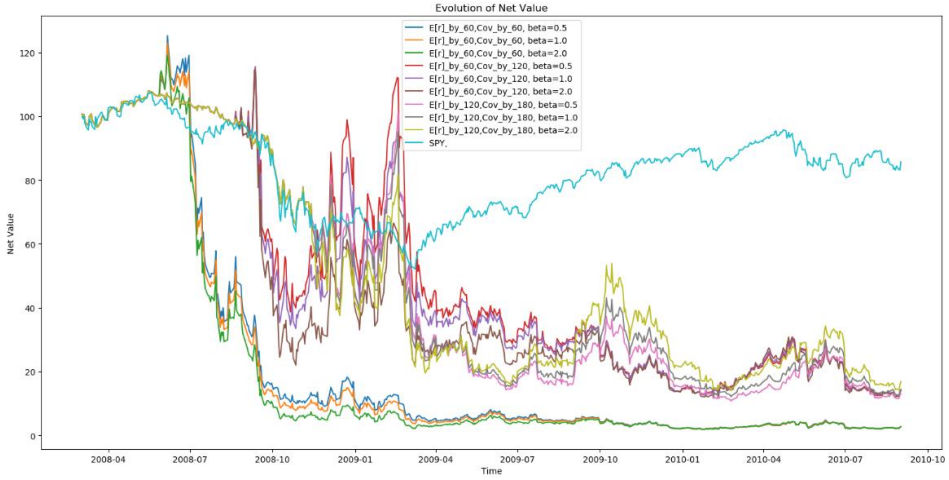
3. Compared with the benchmark SPY, our portfolios didn't infest a better result. Take $\beta_T^m = 1$ as example,

	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$	SPY
Annual_cumulatedRet	0.241	0.461037	0.464995	0.940682
Annual_VaR	-193.409	-170.835	-152.454	-53.0627
Annual_SharpeRatio	-0.69531	-0.2806	-0.43146	-0.21535

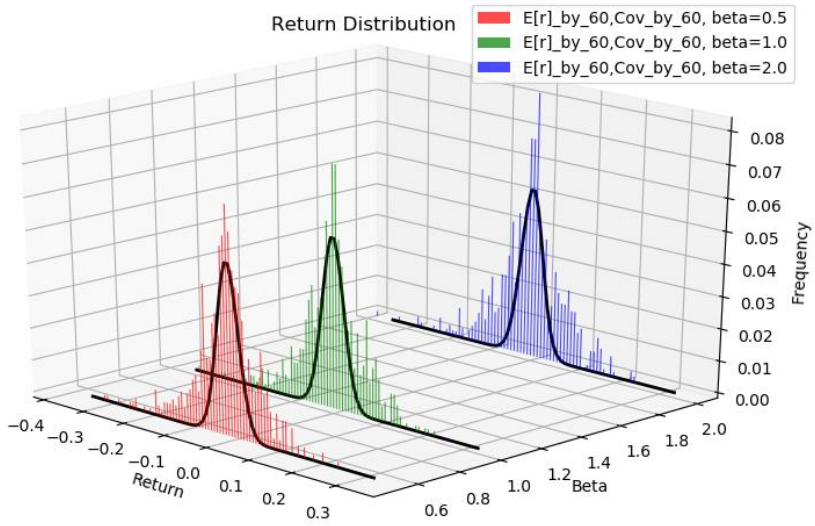
4. To be specific, the max-ten-day drawdown reaches an incredibly high level up to 50% and the Skewness of distribution is negative which shows that all the portfolios suffer large amounts of great loss

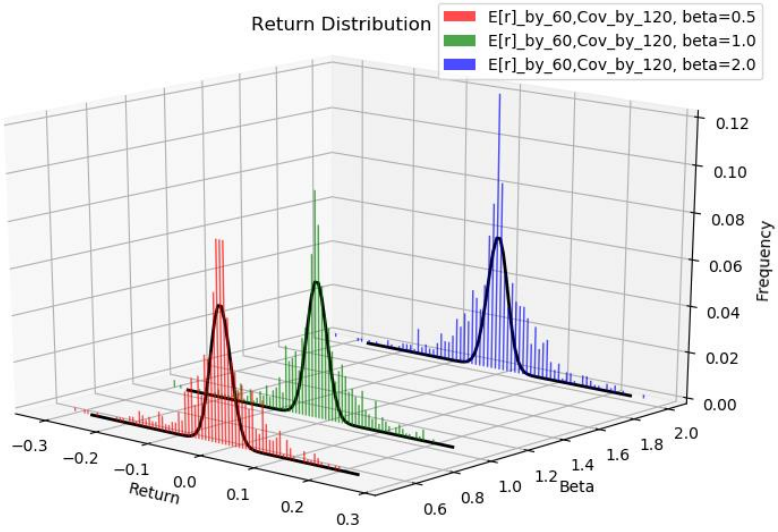
Our portfolios didn't obviously behavior better than the benchmark. So we can conclude that, during the doom days of the whole market, nearly none of the optimized strategy could guarantee a stable profit.

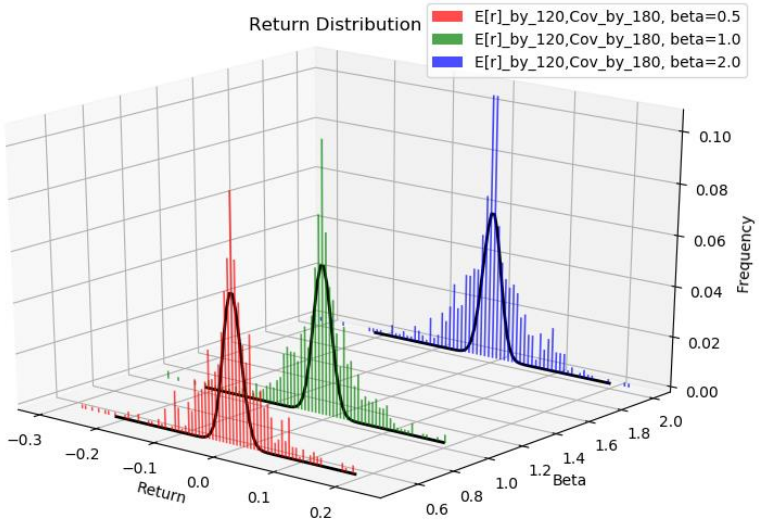
The following chart shows the curves of the PnL between different portfolios *during the crisis*.



The above chart validates our conclusions on the data table. Through it, we can see that almost all the portfolios end with a great dump during a long period. Note that, the short term period portfolios dived quickly at the beginning and because they lost too much at the early time, these short term period portfolios continued to behavior weakly even the market rebounded heavily.







From these charts we can conclude that the longer term of the estimation is, the distribution is more likely to cluster to its mean, which means they have a lower volatility but a higher kurtosis.

Compared with charts before crisis, the tails of charts here are fatter. That meant there were more extreme values of return occurred.

Portfolio Indicators Table (After the crisis)

	$RetCov_{60}^{60}$			$RetCov_{120}^{60}$			$RetCov_{180}^{120}$			SPY
	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	
Annual_cumulatedRet	0.88751	0.942598	1.03138	0.806154	0.855236	0.907485	0.609327	0.633822	0.673824	1.19027
Annual_ariMeanRet	0.212363	0.280909	0.401854	0.108576	0.172969	0.255503	-0.20018	-0.14944	-0.05949	0.185637
Annual_geoMeanRet	-0.11931	-0.05911	0.030903	-0.21539	-0.15633	-0.09706	-0.49491	-0.45557	-0.39448	0.174241
Annual_MinRet	-76.9208	-82.0847	-85.7576	-67.9049	-60.6967	-59.9987	-53.4519	-59.4353	-66.8809	-16.2808
Max_10day_Drawdown	-0.46789	-0.4952	-0.53176	-0.46978	-0.46752	-0.51094	-0.46363	-0.49245	-0.54726	-0.15803
vol	0.809279	0.818994	0.855808	0.798417	0.805973	0.835068	0.766773	0.780919	0.816332	0.150784
Annual_SharpeRatio	0.18827	0.269733	0.399452	0.06084	0.140164	0.234116	-0.33932	-0.2682	-0.14638	0.833225
Annual_Kurtosis	1.9383	2.064514	2.067169	1.943001	1.709268	1.59852	2.245322	2.090621	2.023986	5.103773
Annual_Skew	-0.2143	-0.23489	-0.1962	-0.3184	-0.25808	-0.1774	0.114424	0.06771	0.030708	-0.45269
Annual_mVaR	-133.466	-134.907	-139.367	-134.585	-134.509	-137.208	-121.402	-124.59	-130.603	-23.9568
Daily_VaR	-8.33396	-8.4076	-8.74221	-8.26247	-8.31532	-8.58498	-8.05679	-8.18365	-8.51607	-1.49434
Annual_VaR	-131.771	-132.936	-138.226	-130.641	-131.477	-135.741	-127.389	-129.395	-134.651	-23.6276
Annual_CVaR	-180.151	-179.963	-185.177	-189.062	-189.191	-191.505	-174.209	-176.593	-183.163	-36.0845

We make comparisons with portfolios under the same period of estimation between different β_T^m and portfolios under the same β_T^m between different periods.

As is shown, we conclude that *after the crisis*:

1. Under the same period of estimation, the portfolio with larger β_T^m tends to have a higher return. And the portfolios they share almost the same risk. So the portfolio can be leveraged that we can make more profit with the same risk level. Take $RetCov_{60}^{60}$ as example,

	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$
Annual_cumulatedRet	0.88751	0.942598	1.03138
Annual_VaR	-131.771	-132.936	-138.226

2. Under the same β_T^m, the portfolio with short and medium period of estimation tends to have a higher return than the long one. However, although they behavior differently, they all continue to suffer loss after the crisis. Take $\beta_T^m = 0.5$ as example,

	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$
Annual_cumulatedRet	0.88751	0.806154	0.609327
Annual_VaR	-131.771	-130.641	-127.389

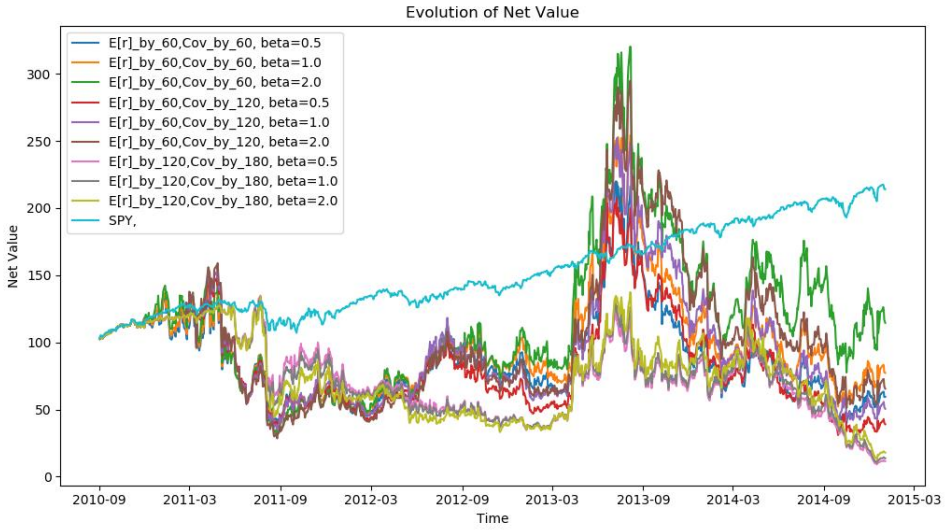
3. Compared with the benchmark SPY, our portfolios didn't infest a better result. Take $\beta_T^m = 1$ as example,

	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$	SPY
Annual_cumulatedRet	0.942598	0.855236	0.633822	1.19027
Annual_VaR	-132.936	-131.477	-129.395	-23.6276
Annual_SharpeRatio	0.269733	0.140164	-0.2682	0.833225

4. To be specific, the max-ten-day drawdown reaches an incredibly high level up to 50% and the skewness of distribution under short and medium term is negative, while the skewness of the long term is positive, which shows that portfolios of longer term may have more days of surge than shorter ones and have more days of larger slide than shorter ones.

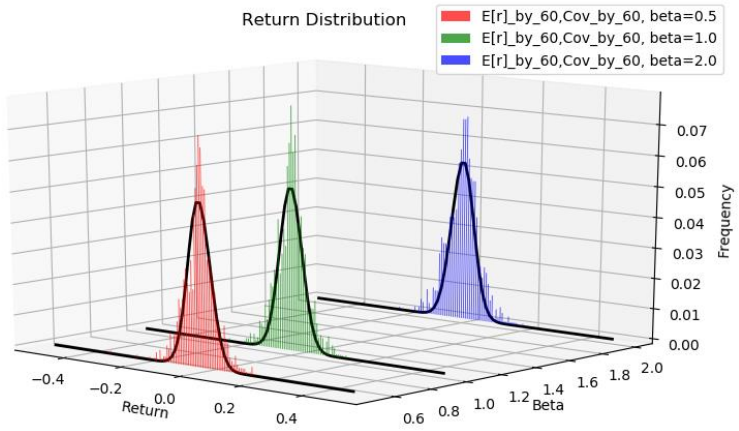
Our portfolios didn't obviously behavior better than the benchmark. So we can conclude that, even after the doom days of the whole market, this optimized strategy might be not able to guarantee a stable profit.

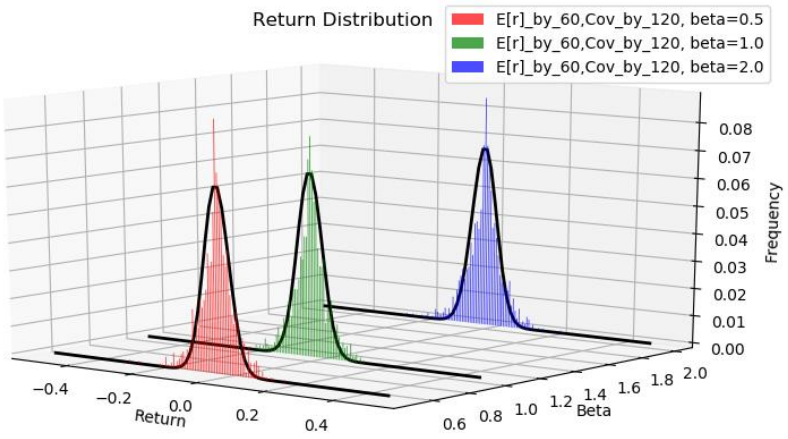
The following chart shows the curves of the PnL between different portfolios *after the crisis*.

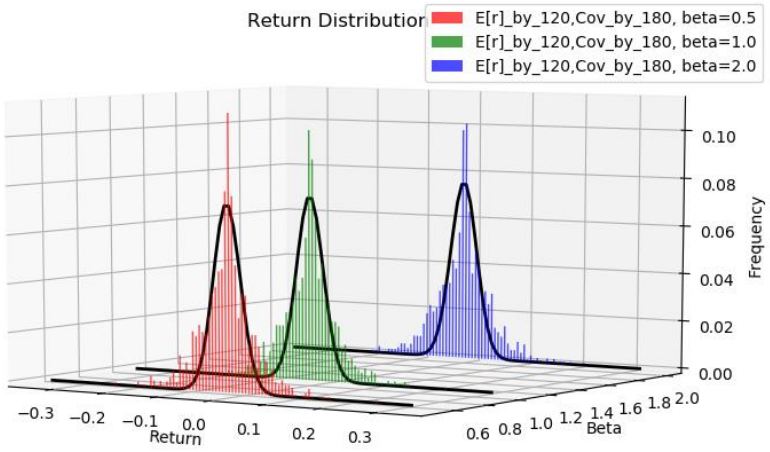


The above chart validates our conclusions on the data table. We find that during the first half year in 2013, our portfolios ascended dramatically. However, they began to decrease from the last quarter of 2013. Although the benchmark continued to rise in the long term, our portfolios unfortunately dumped finally.

Note that, the short term period portfolios dived quickly at the beginning and because they lost too much at the early time, these short term period portfolios continued to behavior weakly even the market rebounced heavily.







These three charts show smoother than those during the crisis. And we can get the similar conclusions of the period during the crisis.

Portfolio Indicators Table (the whole period)

	$RetCov_{60}^{60}$			$RetCov_{120}^{60}$			$RetCov_{180}^{120}$			SPY
	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$	
Annual_cumulatedRet	0.974866	1.007877	1.002563	0.839091	0.863545	0.889286	0.813562	0.845447	0.842016	1.075386
Annual_ariMeanRet	0.523982	0.564669	0.602798	0.366191	0.396586	0.454471	0.290148	0.336409	0.372601	0.093462
Annual_geoMeanRet	-0.02545	0.007846	0.00256	-0.17537	-0.14667	-0.11731	-0.20625	-0.16783	-0.1719	0.07269
Annual_MinRet	-90.7584	-89.5219	-85.9976	-87.5247	-86.7449	-90.1911	-87.5766	-89.6682	-103.513	-24.612
Max_10day_Drawdown	-0.6338	-0.6557	-0.68543	-0.62223	-0.64395	-0.67646	-0.59312	-0.59459	-0.64185	-0.24948
vol	1.040742	1.048299	1.089641	1.030584	1.032457	1.060122	0.989518	0.996305	1.032883	0.203909
Annual_SharpeRatio	0.445819	0.481417	0.498144	0.297105	0.326005	0.3721	0.232586	0.277434	0.302649	0.164104
Annual_Kurtosis	2.140554	2.049294	2.090989	2.593418	2.66358	3.07723	2.02244	2.092054	2.49478	14.75758
Annual_Skew	-0.13189	-0.1119	-0.04542	-0.21506	-0.20256	-0.13512	-0.13611	-0.17681	-0.23488	0.213277
Annual_mVaR	-167.245	-167.833	-172.222	-168.017	-167.631	-168.952	-160.682	-162.493	-169.126	-25.6226
Daily_VaR	-10.6172	-10.6795	-11.0944	-10.5747	-10.582	-10.8466	-10.1779	-10.23	-10.596	-2.08388
Annual_VaR	-167.873	-168.858	-175.418	-167.2	-167.316	-171.5	-160.926	-161.75	-167.538	-32.949
Annual_CVaR	-233.955	-234.616	-242.517	-237.726	-237.521	-241.763	-228.557	-229.73	-237.245	-49.7102

We make comparisons with portfolios under the same period of estimation between different β_T^m and portfolios under the same β_T^m between different periods.

As is shown, we conclude that *during the whole period*:

1. During the whole period of estimation, the portfolio with different β_T^m tend to have a similar return. And the portfolios they share almost the same risk. Take $RetCov_{60}^{60}$ as example,

	$\beta_T^m = 0.5$	$\beta_T^m = 1$	$\beta_T^m = 2$
Annual_cumulatedRet	0.974866	1.00788	1.00256
Annual_VaR	-167.873	-168.858	-175.418

2. Under the same β_T^m, the portfolio with short period of estimation tends to have a higher return than the other two portfolios. And the portfolio with longer term period has a slightly smaller risk exposure. Take $\beta_T^m = 0.5$ as example,

	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$
Annual_cumulatedRet	0.974866	0.839091	0.813562
Annual_VaR	-167.873	-167.2	-160.926

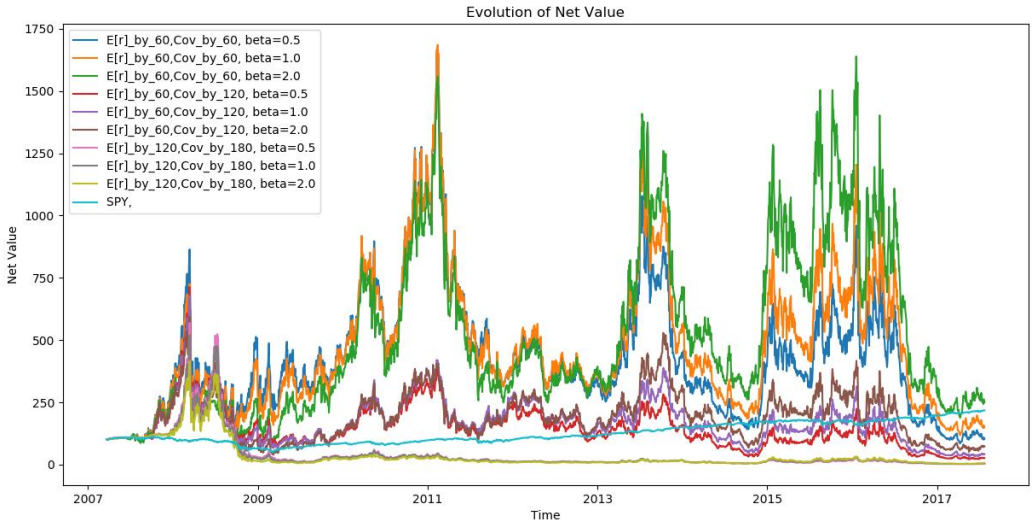
3. Compared with the benchmark SPY, our portfolios didn't infest a better result. Take $\beta_T^m = 1$ as example,

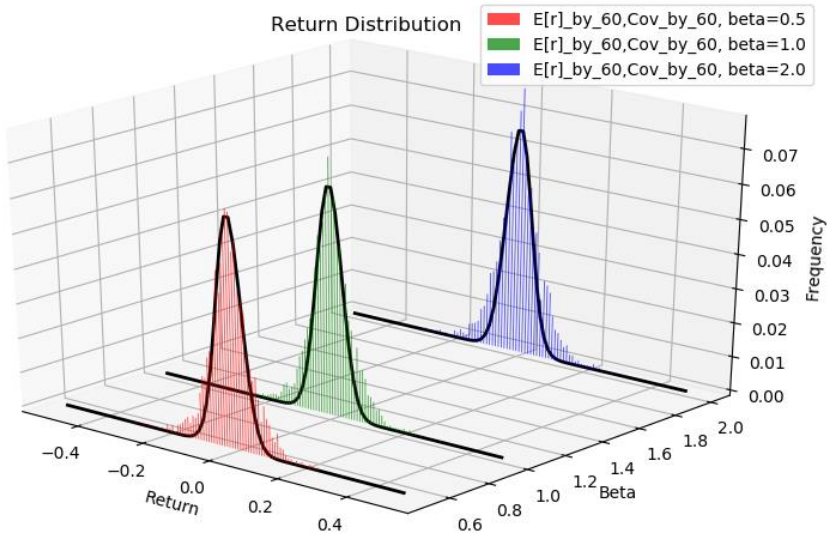
	$RetCov_{60}^{60}$	$RetCov_{120}^{60}$	$RetCov_{180}^{120}$	SPY
Annual_cumulatedRet	1.00788	0.863545	0.845447	1.07539
Annual_VaR	-168.858	-167.316	-161.75	-32.949
Annual_SharpeRatio	0.481417	0.326005	0.277434	0.164104

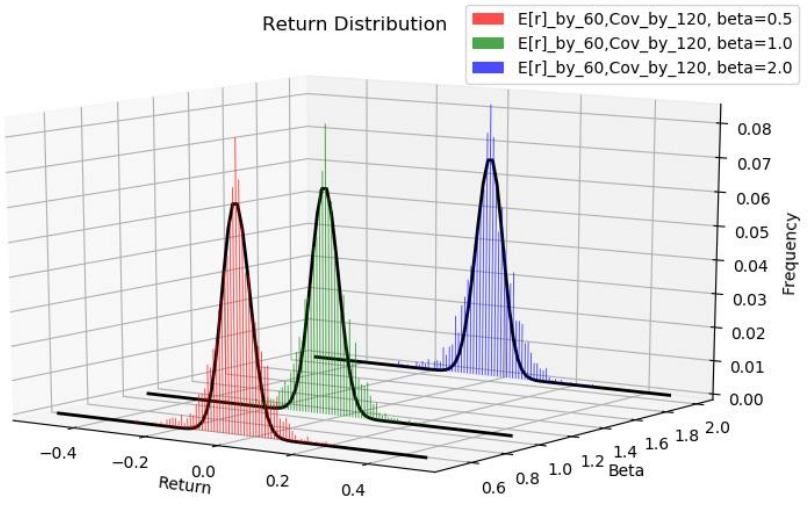
4. To be specific, the max-ten-day drawdown reaches an incredibly high level up to 50% and the skewness of distribution is negative.

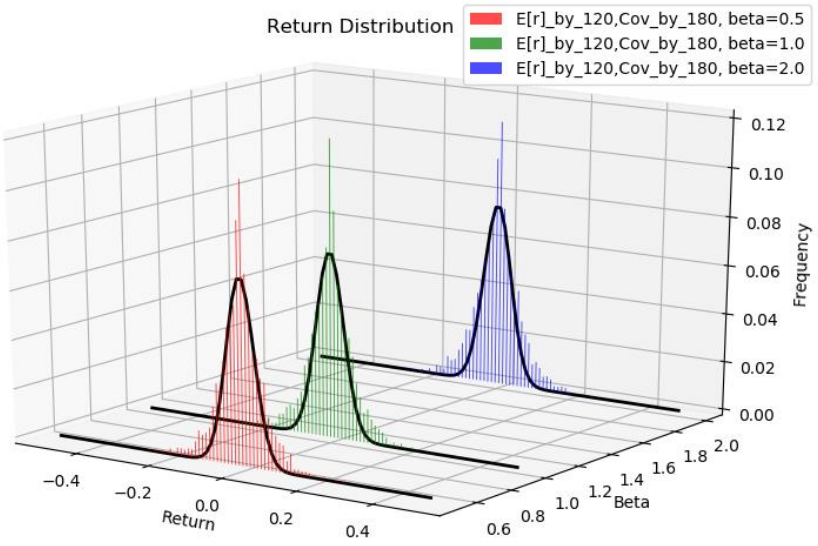
Our portfolios didn't obviously behavior better than the benchmark. So we can conclude that, even during the whole market, this optimized strategy might be not able to guarantee a stable profit.

The following chart shows the curves of the PnL between different portfolios *during the whole time*.









Due to the large quantity of our data, the result we get shows a normal distribution. Our data includes all the period of the market so we have got the full information of all assets.

Overall Conclusion

As a normal sense, portfolios with higher beta should tend to have a higher return expectation and risk exposure. However, according to our calculation between different periods and different expected beta, optimized portfolios didn't show a definite good behavior. We have examined our program hundreds of time and we can confidently tell that we executed the portfolio accurately. As the result isn't satisfying, we assume that this model might not fit the market.

If we are asked to recommend the strategy to the investors for different periods, we would like to recommend them that:

Before crisis, use the portfolio with $\mathbf{RetCov}_{\mathbf{180}}^{\mathbf{120}}$ and $\boldsymbol{\beta}_{\mathbf{T}}^{\mathbf{m}} = \mathbf{0.5}$ because it has a larger Shape Ratio and relatively lower drawdown.

During crisis, use the portfolio with $\mathbf{RetCov}_{\mathbf{180}}^{\mathbf{120}}$ and $\boldsymbol{\beta}_{\mathbf{T}}^{\mathbf{m}} = \mathbf{2.0}$ because during the doom days it performed the best return and it has a smaller volatility.

After crisis, use the portfolio with $\mathbf{RetCov}_{\mathbf{60}}^{\mathbf{60}}$ and $\boldsymbol{\beta}_{\mathbf{T}}^{\mathbf{m}} = \mathbf{2.0}$ because it has a higher cumulative return and it has a larger Shape Ratio.

When we analyze the portfolio sensitivity to the term and beta, we find that its sensitivity to beta is not obvious but its return has a great relationship with estimation term. What's more, the return of this strategy has a great sensitivity to the starting point of the back test.

Code Instruction

1.Preview:

We separate our program into four parts: main.py, omega_w.py, backtest.py, performance.py

(a) download.py: the function that downloads and processes the data.

(b) main.py: the main function that passes the initial attributes. This function calls other sub-functions to get the final results and displays plots.

(c) omega_w.py: the target function that calculates the optimized weights of each asset. The return of it is the adjusted ω.

(d) backtest.py: the back-test function that does the back-test so that we can get an updated weight of the assets.

(e) performance.py: the function that calculates and displays the indicators.

2.Code explanation:

(a) performance.py:

```
# We download the needed data and save it into an csv file after processing it.
#-----------------------------------------------------------------------------
import pandas as pd
from pandas_datareader import data as pdrd
import numpy as np
import os
import datetime as dt

path = r"D:\Grad 2\630\630assignment\Final"
os.chdir(path)

# function of downlaoding the data
def download(stocklist):
    startTime = dt.datetime(2007,1,1)
    endTime=dt.datetime(2019,4,1)
    for stock in stocklist:
        equity_data = pdrd.DataReader(stock, data_source='yahoo', start=startTime, end=endTime)
        equity_data.to_csv('{}.csv'.format(stock))

# read in the data and process it
datalist=[]
```

```
def processdata(datalist):
    d=pd.DataFrame()
    for name in datalist:
        file='{}.csv'.format(name)
        data=pd.read_csv(file).set_index(['Date'])['Adj Close']
        data=data.loc['2007-03-01':'2019-04-01']
        d=pd.concat([d,data],axis=1,sort=False)
    #d.dropna()
    return d

datalist=['SPY','FXE','EWJ','GLD','QQQ','DBA','SHV','GAF','USO','XBI','ILF','FEZ','EPP']

data=processdata(datalist)
data.columns=datalist
data=data[data['GAF'].notnull()]
F=pd.read_csv('F1.csv')
F=F[(F['Unnamed: 0']>='20070301')&(F['Unnamed: 0']<='20190401')]
F=F.rename(columns={'Unnamed: 0':'Date'}).set_index('Date')
F.index = pd.to_datetime(F.index.astype(str), format='%Y%m%d')
data=data.join(F)
data.to_csv(r'630data.csv')

data.head()

logdata=data.T.iloc[0:-4].T
for col in logdata.columns:
    logdata[col]=np.log(logdata[col])-np.log(logdata[col].shift(1))
logret=logdata
logret.to_csv(r'logret.csv')
simpledata=data.T.iloc[0:-4].T
for col in simpledata.columns:
    simpledata[col]=(simpledata[col]-simpledata[col].shift(1))/simpledata[col].shift(1)
simpleret=simpledata
simpleret.to_csv(r'simpleret.csv')
```

(b) main.py:

```
# We import the processed data and save it into an object
#-------------------------------------------------------------------------------
# libs needed:
import os
os.chdir(r"D:\Grad 2\630\630assignment\Final")
import pandas as pd
import backtest
import performance
import numpy as np
```

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.mlab as mlab

all_data = pd.read_csv("630data.csv",index_col=0)
ff_factor = all_data.loc[:,"Mkt-RF":r"RF"]
etf_data = all_data.loc[:,:"EPP"]

etf_return = (etf_data/etf_data.shift(1)-1).dropna(axis = 0)
ff_data = ff_factor.iloc[1:,:]


#  We slice out the data according to the dates before crisis. After we searched information
online, # we finally defined 2008-03-03 as the Eve of the crisis. The following the main function of
getting # the result of back test returns.
#-------------------------------------------------------------------------------
# before crisis
bc_test_etf_return = etf_return.loc[:"2008-03-03",:]
bc_test_ff_data = ff_data.loc[:"2008-03-03",:]
bc_lookback_list = [[60,60],[60,120],[120,180]]
bc_beta_list = [0.5,1,2]
bc_performance_result = pd.DataFrame([])
bc_return_result = pd.DataFrame([])
omega_list = []
for lb in bc_lookback_list:
   for bt in bc_beta_list:
      res = backtest.back_test(etf_return = bc_test_etf_return,
                        ff_data = bc_test_ff_data,
                        return_period = lb[0],
                        variance_period = lb[1],
                        lamb = 10,
                        beta_tm = bt)
      omega_list.append(res[1])
      res = pd.DataFrame(res[0],index = pd.to_datetime(bc_test_etf_return.index))
      res_perf = performance.indicator(X = res,rf = 0.06, confidenceLevel = 0.95, position = 100)
      bc_return_result = pd.concat([bc_return_result,res],axis = 1)
      bc_performance_result = pd.concat([bc_performance_result,res_perf],axis = 1)

bc_return_result = pd.concat([bc_return_result,bc_test_etf_return['SPY']],axis = 1)

bc_spy_performance = performance.indicator(X = pd.DataFrame(bc_test_etf_return.loc[:,'SPY']),rf
= 0.06, confidenceLevel = 0.95, position = 100)
bc_performance_result = pd.concat([bc_performance_result,bc_spy_performance],axis = 1)
bc_performance_result.columns =
[['E[r]_by_60,Cov_by_60','E[r]_by_60,Cov_by_60','E[r]_by_60,Cov_by_60',
                'E[r]_by_60,Cov_by_120','E[r]_by_60,Cov_by_120','E[r]_by_60,Cov_by_120',

'E[r]_by_120,Cov_by_180','E[r]_by_120,Cov_by_180','E[r]_by_120,Cov_by_180','SPY'],
```

```
                ['beta=0.5','beta=1.0','beta=2.0',
                 'beta=0.5','beta=1.0','beta=2.0',
                 'beta=0.5','beta=1.0','beta=2.0','']]
bc_return_result.columns = bc_performance_result.columns

# We plot the histogram plot with the kernel density of calculated returns.
# ------------------------------------------------------------------------
# plot
for i in range(10):
   plt.plot(100*(np.cumprod(bc_return_result.iloc[:,i]+1)), label = bc_return_result.columns[i][0] +
bc_return_result.columns[i][1])
   plt.legend()

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for i in range(3):
   c =  ['r', 'g', 'b'][i]
   z =  [0.5,1,2][i]
   x,y = np.histogram(bc_return_result.iloc[:,i],bins = 80)
   y = (y[:-1]+y[1:])/2
   cs = [c] * len(x)
   ax.bar(y, x, zs=z, zdir='y', color=cs, alpha=0.7,width = 0.006,label =
bc_return_result.columns[i][0]+bc_return_result.columns[i][1])
   ax.legend()
   l = mlab.normpdf(y,np.mean(bc_return_result.iloc[:,i]),np.std(bc_return_result.iloc[:,i]))
   ax.plot(y,[z]*len(y),l,color ='black',linewidth = 2.0)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for i in range(3):
   c =  ['r', 'g', 'b'][i]
   z =  [0.5,1,2][i]
   x,y = np.histogram(bc_return_result.iloc[:,i+3],bins = 80)
   y = (y[:-1]+y[1:])/2
   cs = [c] * len(x)
   ax.bar(y, x, zs=z, zdir='y', color=cs, alpha=0.7,width = 0.006,label =
bc_return_result.columns[i+3][0]+bc_return_result.columns[i+3][1])
   ax.legend()
   l = mlab.normpdf(y,np.mean(bc_return_result.iloc[:,i+3]),np.std(bc_return_result.iloc[:,i+3]))
   ax.plot(y,[z]*len(y),l,color ='black',linewidth = 2.0)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for i in range(3):
   c =  ['r', 'g', 'b'][i]
   z =  [0.5,1,2][i]
   x,y = np.histogram(bc_return_result.iloc[:,i+6],bins = 80)
   y = (y[:-1]+y[1:])/2
```

```
    cs = [c] * len(x)
    ax.bar(y, x, zs=z, zdir='y', color=cs, alpha=0.7,width = 0.006,label =
bc_return_result.columns[i+6][0]+bc_return_result.columns[i+6][1])
    ax.legend()
    l = mlab.normpdf(y,np.mean(bc_return_result.iloc[:,i+6]),np.std(bc_return_result.iloc[:,i+6]))
    ax.plot(y,[z]*len(y),l,color ='black',linewidth = 2.0)
#  The following is the code for plots of calculated returns during crisis and after crisis. As the
# following code almost has the same format as that before crisis, here we jump to the next code
# file omega_w.py.
# ------------------------------------------------------------------------
```

(c) omega_w.py:

```
#  This is the core function that deals with our target formula and returns the optimized assets
# weight. In this file, we define two functions 'arg_w(...)' and 'get_omega(...)'.
# The arg_w(...) is the implementation of optimization problem
```

$$\begin{cases} \max\limits_{\omega\in\mathbb{R}^n} \rho^T\omega - \lambda(\omega-\omega_p)^T Q(\omega-\omega_p) \\ \sum\limits_{i=1}^{n} \beta_i^m \omega_i = \beta_T^m \\ \sum\limits_{i=1}^{n} \omega_i = 1, \qquad -2 \le \omega_i \le 2, \end{cases}$$

```
# The get_omega(...) is the implementation of getting the attributes:
```

$$\rho_i = r_f + \beta_i^3(\rho_M - r_f) + b_i^s \rho_{SMB} + b_i^v \rho_{HML} + \alpha_i$$

$$R_t = \begin{bmatrix} r_{1t}^e \\ r_{2t}^e \\ \dots \\ r_{nt}^e \end{bmatrix}, B = \begin{bmatrix} \beta_1' \\ \beta_2' \\ \dots \\ \beta_n' \end{bmatrix} = \begin{bmatrix} \beta_1^3 & b_1^s & b_1^v \\ \vdots & \ddots & \vdots \\ \beta_n^3 & b_n^s & b_n^v \end{bmatrix}$$

$$R_t = \alpha + \mathbf{B}\mathbf{f}_t + \varepsilon_t \text{ for } i = 1,2,\dots,\mathrm{T}$$

$$cov(R_t) = \mathbf{B}\Omega_f\mathbf{B}' + D = \mathbf{B}\Omega_f\mathbf{B}^T + diag(\sigma_1^2,\dots,\sigma_n^2)$$

```
# ------------------------------------------------------------------------
import scipy
from scipy.optimize import minimize
from sklearn import linear_model
import numpy as np

def arg_w(rho, lamb, Q, wp, beta_im_ ,beta_T):
    def constrain1(w):
        return np.dot(beta_im_,w)-beta_T
    def constrain2(w):
        return np.sum(w)-1
    cons = [{'type':'eq', 'fun': constrain1},
            {'type':'eq', 'fun': constrain2}]
    bnds = scipy.optimize.Bounds(-2.0, 2.0, keep_feasible = True)
```

```
    def f(w):
        return -rho.dot(w) + lamb*(w-wp).dot(Q.dot(w-wp))
    w0 = np.array([1/13]*13)
    res = minimize(f, w0, method='SLSQP', bounds=bnds, constraints=cons,
                   tol=1e-9)
    return res.x

def get_omega(return_r, factor_r, return_v, factor_v, lamb_, beta_tm_, wp_):
    rf = np.asarray(factor_r['RF'])
    rM_rf = np.asarray(factor_r['Mkt-RF'])
    rSMB = np.asarray(factor_r['SMB'])
    rHML = np.asarray(factor_r['HML'])
    SPY = np.asarray(return_r['SPY'])

    ri = np.asarray(return_r)

    var_market = np.var(SPY,ddof=1)
    beta_im = np.array([0.0]*13)
    for i in range (13):
        temp = np.cov(ri[:,i],SPY,ddof=1)
        beta_im[i] = temp[0,1] / var_market

    Ri = ri - rf.reshape(-1,1)
    f = np.array([rM_rf, rSMB, rHML])
    F = f.T

    lr = linear_model.LinearRegression().fit(F, Ri)
    alpha = lr.intercept_
    B = lr.coef_
    # get rho of short period
    ft = f[:,-1]
    rho_r = alpha + B.dot(ft) + rf[-1]

    # -----------------------------------------------------------------------
    rf_v = np.asarray(factor_v['RF'])
    rM_rf_v = np.asarray(factor_v['Mkt-RF'])
    rSMB_v = np.asarray(factor_v['SMB'])
    rHML_v = np.asarray(factor_v['HML'])
    SPY_v = np.asarray(return_v['SPY'])
    ri_v = np.asarray(return_v)

    var_market_v = np.var(SPY_v,ddof=1)
    beta_im_v = np.array([0.0]*13)
    for i in range (13):
        temp_v = np.cov(ri_v[:,i],SPY_v,ddof=1)
        beta_im_v[i] = temp_v[0,1] / var_market_v

    Ri_v = ri_v - rf_v.reshape(-1,1)
```

```
    f_v = np.array([rM_rf_v, rSMB_v, rHML_v])
    F_v = f_v.T

    lr_v = linear_model.LinearRegression().fit(F_v, Ri_v)
    alpha_v = lr_v.intercept_
    B_v = lr_v.coef_

    eph_v = Ri_v.T - (alpha_v.reshape(-1,1) + B_v.dot(f_v))
    eph2_v = np.cov(eph_v,ddof=1)
    eph2_diag_v = np.diag(eph2_v)
    D_v = np.diag(eph2_diag_v)

    omega_f_v = np.cov(f_v,ddof=1)
    cov_Rt_v = B_v.dot(omega_f_v).dot(B_v.T) + D_v

    result = arg_w(rho_r, lamb_, cov_Rt_v, wp_, beta_im_v ,beta_tm_)

    return result
```

(d) backtest.py:

```
# This function is used to implement the strategy on the historical data to get the daily portfolio #
return. Inside the function, we reset our portfolio weight every five days. Finally, the function
# returns a list of portfolio returns and a list of every asset weight everyday.
# ------------------------------------------------------------------------
import os
import pandas as pd
os.chdir(r"D:\Grad 2\630\630assignment\Final")
import numpy as np
import omega_w

def back_test(etf_return, ff_data, return_period, variance_period, lamb, beta_tm):
    port_return_list = []
    omega_list = []
    omega_p = np.array([1/13]*13)
    look_back =  max(return_period,variance_period)
    next_chang_date = look_back-1
    for i in range(len(etf_return)):
        omega_list.append(omega_p)
        today_return = np.asarray(etf_return.iloc[i,:])
        pr = np.dot(omega_p,today_return)
        port_return_list.append(pr)
        if i == next_chang_date:
            omega_p = omega_w.get_omega(return_r = etf_return.iloc[i+1-return_period:i+1],
                        factor_r =ff_data.iloc[i+1-return_period:i+1],
```

```
                    return_v = etf_return.iloc[i+1-variance_period:i+1],
                    factor_v = ff_data.iloc[i+1-variance_period:i+1],
                    lamb_ = lamb,
                    beta_tm_ = beta_tm,
                    wp_ = omega_p)
        next_chang_date += 5
    else:
        continue
  return port_return_list,omega_list
```

(e) performance.py:

```
# This function is used to implement the table of indicators required by final's PDF. It returns a
# dataframe of all the results we need which shows like:
```

	$MaxRet_{60}^{60}$	$MaxRet_{120}^{60}$	$MaxRet_{180}^{120}$	SPY
	Target $\beta^m = 0.5$	Target $\beta^m = 1$	Target $\beta^m = 2$	
Mean Return				
…				
Max DD				

```
# ------------------------------------------------------------------------

import pandas as pd
import numpy as np
from scipy.stats import kurtosis,skew,norm

def indicator(X,rf,confidenceLevel,position):
    #PnL:
    daily_cum_return=np.cumprod((X+1))
    annual_cum_return = daily_cum_return.iloc[-1,0]/len(X)*250

    #Daily Mean return(%):
    annual_arith_MeanRet=np.mean(X)*250

    #Geomean is zero if anyone of return rate is zero
    annual_geo_MeanRet=(np.power(daily_cum_return.iloc[-1,0],1/len(X))-1)*250
    #or
    #geoMeanRet=scipy.stats.mstats.gmean(X['simple ret rate'])
    # Min Return
    annual_MinRet = np.min(X)*250

    #MDD
    p_v =np.cumprod((X+1))*100
    p_v_extend = pd.DataFrame(np.append([p_v.iloc[0,0]]*9,p_v))
```

```
    Roll_Max = p_v_extend.rolling(window=10).max()

    tenday_Drawdown = float(np.min(p_v_extend/Roll_Max-1)[0])

    #volatility
    annual_vol=np.std(X)*np.sqrt(250)

    #Sharp ratio
    annualRatio=(annual_arith_MeanRet-rf)/annual_vol

    #Skewness, Kurtosis
    annual_Kurt=kurtosis(X*250)
    annual_sk=skew(X*250)

    #MVaR VaR
    daily_Kurt=kurtosis(X)
    daily_sk=skew(X)
    z=norm.ppf(1-confidenceLevel)
    t=z+((1/6)*(z**2-1)*daily_sk)+((1/24)*(z**3-3*z))*daily_Kurt-((1/36)*(2*z**3-5*z)*(daily_sk**2))
    mVaR= position*(np.mean(X)+t*np.std(X))*np.sqrt(250)
    alpha=norm.ppf(1-confidenceLevel, np.mean(X), np.std(X))
    VaR= position*(alpha)
    annualVaR=VaR*np.sqrt(250)
    CVaR = position*np.mean(X[X<=np.quantile(X,1-confidenceLevel)])[0]*np.sqrt(250)

df=pd.DataFrame([annual_cum_return,annual_arith_MeanRet[0],annual_geo_MeanRet,annual_MinRet[0],tenday_Drawdown,annual_vol[0],annualRatio[0],annual_Kurt[0],annual_sk[0],mVaR[0],VaR[0],annualVaR[0],CVaR],

index=['Annual_cumulatedRet','Annual_ariMeanRet','Annual_geoMeanRet','Annual_MinRet','Max_10day_Drawdown','vol',

'Annual_SharpeRatio','Annual_Kurtosis','Annual_Skew','Annual_mVaR','Daily_VaR','Annual_VaR','Annual_CVaR'],
                columns=['result'])
    return df
```

Datasets Cleaning:

1. Define a ‘processdata’ function, and set the parameter as a equity basket list, so that we could simply append datasets to this list.
2. Keep ‘Adj Close’ column and remove the rest columns.
3. Slice the data by date from 2007-03-01 to 2019-04-01.
4. Combine these columns from different equities into one dataframe.
5. Drop all data with NaNs.
6. Calculate simple returns for all equities based on dataframe.

Comments:

After cleaning the datasets, we have removed trading dates with no information for any equity, and our dataframe ends at 2017-07-24

Performance Calculating:

1. Arithmetic mean and geometric mean of simple return:
   From the mean return, we can learn the average performance of the portfolio.

2. Maximun 10day drawdown:
   To calculate MDD(10D), we used rolling_max_min function to find the maximum and the following minimum in 10day-period. From MDD, we can learn the stability of portfolio

3. Annual volatility:
   Calculate the standard deviation of simple return. Volatility will tell us the level of risk.

4. Sharp ratio:
   Calculate the difference of annual arithmetic mean of simple return and risk free rate, and then divided by standard deviation of simple return. Sharp ratio helps us understand the return of an investment compared to its risk. The higher sharp ratio yields the better performance of portfolio.

5. Skewness and Kurtosis:
   kurtosis is a statistical measure that is used to describe the distribution. Whereas skewness differentiates extreme values in one versus the other tail, kurtosis measures extreme values in either tail.

6. Value at Risk and conditional Value at Risk:
   VaR tells the potential loss of portfolio at a certain time frame on certain confidence level. Well, conditional value at risk tells the expected potential loss conditioning on the certain case that happens, which is also known as expected shortfall.