In [23]:
```
# Importing modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt

# import the matplotlib_converters from pandas,
# to plot the columns of a dataframe using a for loop
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# load all the stocks of the S&P500 for 1998-2013
df = pd.read_csv("SP500.csv", index_col=[0])

# Convert the index to datetime format
df.index=pd.to_datetime(df.index,format='%Y-%m-%d').date

df.tail()
```

Out[23]:

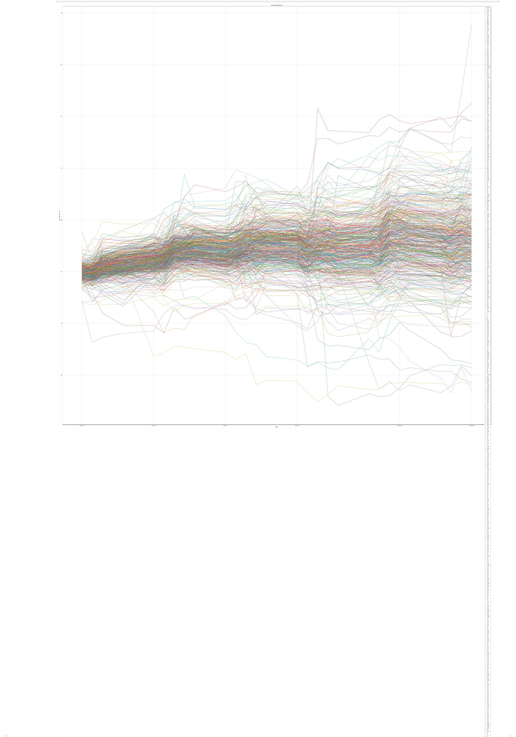
	dlph	cait	coh	mcd	ca	mwv	twc	lsi	life	pnr	...	bbby	chrw	mxla	biib	dri	schw	cl	te	vz	hrs
2013-08-05	55.90	83.54	53.79	99.25	30.56	37.31	116.37	7.83	74.80	63.15	...	78.14	59.60	14.81	226.95	49.58	22.6193	60.96	17.65	50.15	57.74
2013-08-06	55.58	82.47	53.93	98.64	30.71	37.14	114.24	7.67	74.82	63.20	...	77.54	59.58	14.61	221.93	49.20	22.3400	60.94	17.59	50.07	57.25
2013-08-07	54.94	82.42	53.82	98.33	30.61	37.07	114.75	7.62	74.75	63.34	...	76.23	58.38	14.60	217.98	49.70	22.5600	60.95	17.69	49.96	56.97
2013-08-08	55.56	83.98	53.86	98.06	30.88	37.04	114.79	7.66	74.71	63.65	...	76.79	56.72	14.68	217.17	49.50	22.3100	60.99	17.89	49.61	57.61
2013-08-09	55.56	84.49	53.33	97.56	30.97	36.87	114.89	7.65	74.80	63.43	...	78.01	56.77	14.49	212.12	49.54	22.2800	60.66	17.82	49.32	57.65

5 rows × 500 columns

In [83]:
```
# compute the returns of the stocks
df_change = df.pct_change().shift(-1)

# Drop null values
df_change.dropna(inplace=True)
```

In [25]:
```
# Plot percentage returns of all the stocks
plt.figure(figsize=(100, 100))
for column in df_change.columns.values:
    plt.plot(df_change.index, (df_change + 1).cumprod()[column] * 100,
             label=column)
plt.legend(loc='best',fontsize=12)
plt.ylabel('Stock Returns in %',fontsize=14)
plt.xlabel('Year',fontsize=14)
plt.title('Stock Performance',fontsize=16)
plt.grid()
plt.show()
```



In [84]:
```
############################
# MOMENTUM FACTOR          #
############################
# Determine the lookback period to calculate the average price of the stock.
# You can consider a period of 21-working days per month
# and 12 months in a year to generate the yearly average price
yearly_working_days = 21 * 12

# Calculate the momentum values for each stock subtracting the price form its mean
momentum_factor = df - df.rolling(yearly_working_days).mean()

# Determine the buy-sell signals from the momentum values of stocks
# momentum >0 implies bullishness, thus buy the stock.
# momentum <0 implies bearishness, thus sell the stock.
# The momentum_buy_stocks dataframe contains values of the stocks that are more than 0
momentum_buy_stocks = momentum_factor[momentum_factor > 0]
# The momentum_sell_stocks dataframe contains values of the stocks that are less than 0
momentum_sell_stocks = momentum_factor[momentum_factor < 0]

# Calculate the factor performance using the returns and signals
momentum_performance = (df_change * -momentum_buy_stocks.isnull() - df_change
                        * -momentum_sell_stocks.isnull()).mean(axis=1)
```

In [85]:
```
plt.figure(figsize=(10, 6))
plt.plot((momentum_performance + 1).cumprod() * 100,
         label='Momentum Factor Performance')
plt.legend(loc='best', fontsize=12)
plt.ylabel('Returns in %',fontsize=14)
plt.xlabel('Year',fontsize=14)
plt.title('Momentum Factor Returns',fontsize=16)
plt.grid()
plt.show()
```

In [86]:
```
momentum_performance=momentum_performance.dropna().rename('returns')
momentum_performance=pd.DataFrame(momentum_performance)
# Convert the index to datetime format
#momentum_performance.index = pd.to_datetime(momentum_performance.index, format='%Y-%m-%d').date
```

In [87]:
```
############################
# SHORT-TERM REVERSAL      #
############################
# Here the value of 1 would be 5 as you are considering only the past one week's  data
weekly_working_days = 5

# Calculate the rolling mean of the prices and subtract it from prices
mean_reversion_factor = df - df.rolling(weekly_working_days).mean()

# Create two dataframes to hold the buy-sell signals for the rolling mean difference values of stocks
# The reversal_sell_stocks dataframe contains values of stocks that are above the rolling mean
reversal_sell_stocks = mean_reversion_factor[mean_reversion_factor > 0]

# The reversal_buy_stocks dataframe contains values of stocks that are below the rolling mean
reversal_buy_stocks = mean_reversion_factor[mean_reversion_factor < 0]

# Here you assume that the price will revert back to the rolling mean, so you sell the stock when the market goes
# above the moving average and buy the stock when the price is below the moving average
```

In [88]:
```
# Calculate the factor performance using the returns and signals
mean_reversion_performance = (df_change * -reversal_buy_stocks.isnull() - df_change
                              * -reversal_sell_stocks.isnull()).mean(axis=1)

# Convert the mean reversion performance to a dataframe
# with the column name as return and the index as a datatime format.
# When you create a multi-factor model by combining the two factors,
# their indexes should match, so we make the indexes of both momentum
# and mean_reversion factors as datetime.
mean_reversion_performance = pd.DataFrame(
    mean_reversion_performance.dropna(), columns=['return'])

# Convert the mean reversion performance index to a datetime
mean_reversion_performance.index = pd.to_datetime(
    mean_reversion_performance.index, format='%Y-%m-%d')
```

In [89]:
```
# Plot the performance of the factor
plt.figure(figsize=(10, 6))
plt.plot((mean_reversion_performance + 1).cumprod()
         * 100, label='Reversal Factor Performance')
plt.legend(loc='best', fontsize=12)
plt.title('  Short-term Mean Reversion Factor Returns', fontsize=16)
plt.ylabel('Returns in %', fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.grid()
plt.show()
```

In [90]:
```
# Correlation of the two factors
print('Correlation Coeff for the two factors: ', ((momentum_performance['returns']
                                                   + 1).cumprod() * 100).corr((mean_reversion_performance['return']
                                                                               + 1).cumprod() * 100))
```

```
Correlation Coeff for the two factors: -0.9000231718352343
```

In [94]:
```
# Give 50% of the capital to both the factors and calulate the returns of the final combined model
plt.figure(figsize=(10, 7))

# We use the index of mean_reversion_performance as the X-axis to represent the dates and the
# combined factor performance on the Y-axis
plt.plot(mean_reversion_performance.index.date, ((mean_reversion_performance.values + mean_reversion_performance.values) / 2
                                                 + 1).cumprod() * 100, label='Multi Factor Performance')
plt.title('Equal Weighted Portfolio Returns', fontsize=16)
plt.legend(loc='best', fontsize=12)
plt.ylabel('Returns in %', fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.grid()
plt.show()
```

In [ ]:

In [ ]:

In [ ]: