Bond Equity strategies

Objective: Build a strategy that outperforms the traditional 60/40 portfolio

Bond: TLT
Equity: SPY

Notes

- Due to my employer's personal dealing policy, I am restricted to trading only once every 30 days, so my strategy uses a rebalancing period of 31 days (additional day to account for approval process).

Import Packages

In [1]:
```
import openbb as obb
import pandas as pd
import numpy as np
import statsmodels.api as sm
import sklearn as sk
from typing import Union, List, Tuple, Dict, Callable
import datetime as dt
import calendar
import plotly
import plotly.express as px
import plotly.io as pio
import plotly.offline as pyo
pyo.init_notebook_mode(connected = True)
pio.renderers.default = "plotly_mimetype+jupyterlab"
import math
import os
import arch
from Backtest import Backtest
import scipy as sci
```

In [2]:
```
pio.renderers.default = "plotly_mimetype+jupyterlab"
```

Helpers

In [3]:
```
def show_full_df(data: pd.DataFrame) -> None:
    with pd.option_context('display.max_rows', None, 'display.max_columns
        display(data)
```

Benchmark - Buy SPY

In [4]:
```
# Import data from OpenBB
start: dt.date = dt.date(year= 2000,month= 1, day=1)
end: dt.date = dt.date(year= 2025,month= 1, day=1)
```

```
SPY_raw: pd.DataFrame = obb.obb.equity.price.historical(symbol= "SPY", st
SPY: pd.DataFrame = SPY_raw[["close","volume"]].rename({"close":"SPY"}, a
```

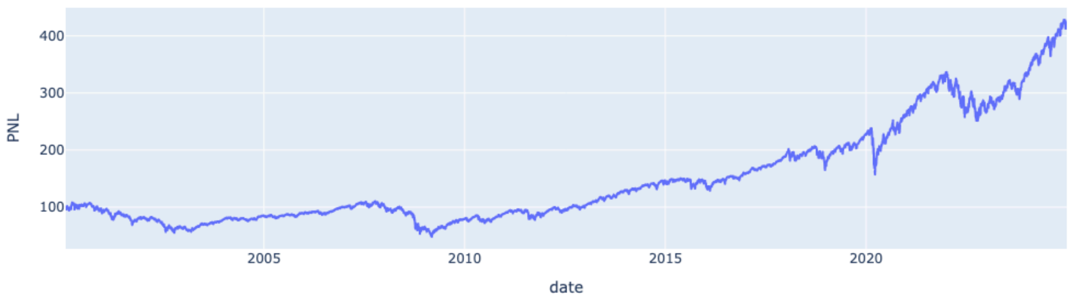
In [5]:
```
def buy_and_hold_func(tick: dt.date,securities: List = ["SPY"], data= SPY
    return (1,{"Add_data":{}})
```

In [6]:
```
buy_and_hold: Backtest = Backtest(name= "Buy and hold", data = SPY)
```

In [7]:
```
buy_and_hold.securities = ["SPY"]
buy_and_hold.assign_rebal_attr(day_of_rebal = 1, interval = "quarter")
buy_and_hold.assign_signals(trading_signal= buy_and_hold_func)
buy_and_hold.lookback_window = 15
```

In [8]:
```
benchmark: pd.DataFrame = buy_and_hold.calculate()
```

In [95]:
```
buy_and_hold.plot()
```



In [10]:
```
buy_and_hold.stats
```

Out[10]:

	Returns	Volatility	Sharpe
Results	0.085997	0.19379	0.443765

In [11]:
```
# Function for strategy comparison
def outperformance(strategy: Backtest, benchmark: Backtest = buy_and_hold
    try:
        flag: int = 1
        if measure == "Volatility":
            flag: int = -1
        out_performance = float((benchmark.stats[measure] - strategy.stat
        print(f"Outperformance ({measure}): %.3f" %(out_performance*flag)
    except:
        print("Make sure Benchmark strategy has been run")
```

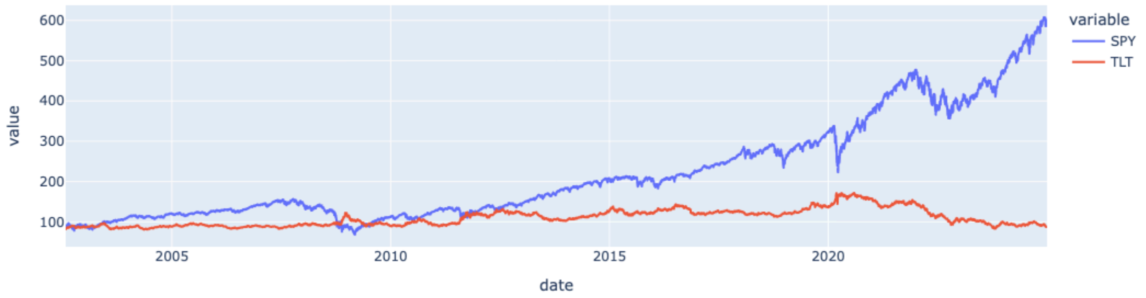
Import data

In [12]:
```
TLT_raw: pd.DataFrame = obb.obb.equity.price.historical(symbol= "TLT", st
```

In [13]:
```
TLT_SPY = SPY.merge(TLT_raw.rename({"close":"TLT"}, axis = 1).TLT, left_i
TLT_SPY.drop("volume", axis = 1, inplace = True)
```

Visualisation and initial analysis

In [96]:
```
fig1 = px.line(TLT_SPY, x = TLT_SPY.index, y = ["SPY", "TLT"])
fig1.update_layout(autosize = True)
fig1.show()
```



In [15]:
```
# Calculate log returns for TLT and SPY
TLT_SPY_logret = (np.log(TLT_SPY) - np.log(TLT_SPY.shift(1))).dropna(how
```

In [16]:
```
# Calculate correlation of TLT and SPY log returns
TLT_SPY_logret.corr()
```

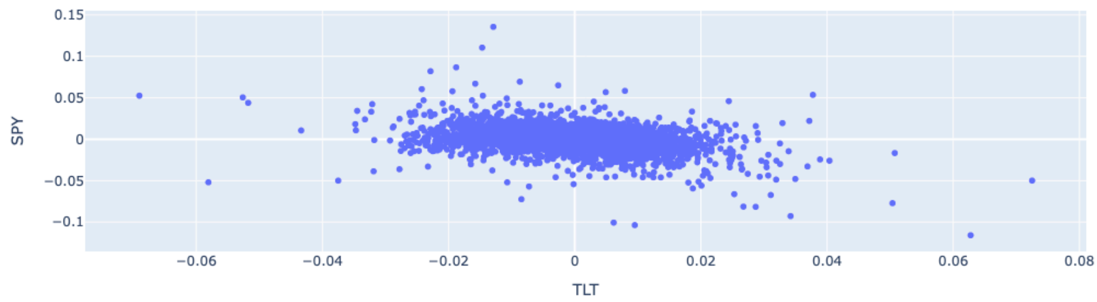
Out[16]:

	SPY	TLT
SPY	1.00000	-0.32641
TLT	-0.32641	1.00000

In [17]:
```
pio.renderers
```

Out[17]:
```
Renderers configuration
-----------------------
    Default renderer: 'plotly_mimetype+jupyterlab'
    Available renderers:
        ['plotly_mimetype', 'jupyterlab', 'nteract', 'vscode',
         'notebook', 'notebook_connected', 'kaggle', 'azure', 'colab',
         'cocalc', 'databricks', 'json', 'png', 'jpeg', 'jpg', 'svg',
         'pdf', 'browser', 'firefox', 'chrome', 'chromium', 'iframe',
         'iframe_connected', 'sphinx_gallery', 'sphinx_gallery_png']
```

In [97]:
```
# Plot scatter of log returns
fig2 = px.scatter(TLT_SPY_logret, x = "TLT", y = "SPY")
fig2.update_layout(autosize = True)
fig2.show()
```



In [98]:
```
# Simple regression model of log returns
reg0_model = sm.regression.linear_model.OLS(TLT_SPY_logret.SPY, TLT_SPY_l
reg0_results = reg0_model.fit()
reg0_results.summary()
```

Out[98]:

OLS Regression Results

Dep. Variable:	SPY	**R-squared (uncentered):**	0.106
Model:	OLS	**Adj. R-squared (uncentered):**	0.106
Method:	Least Squares	**F-statistic:**	672.2
Date:	Wed, 02 Apr 2025	**Prob (F-statistic):**	4.08e-140
Time:	20:46:19	**Log-Likelihood:**	17300.
No. Observations:	5644	**AIC:**	-3.460e+04
Df Residuals:	5643	**BIC:**	-3.459e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
TLT	-0.4260	0.016	-25.926	0.000	-0.458	-0.394

Omnibus:	1152.497	**Durbin-Watson:**	2.147
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	35251.501
Skew:	-0.225	**Prob(JB):**	0.00
Kurtosis:	15.235	**Cond. No.**	1.00

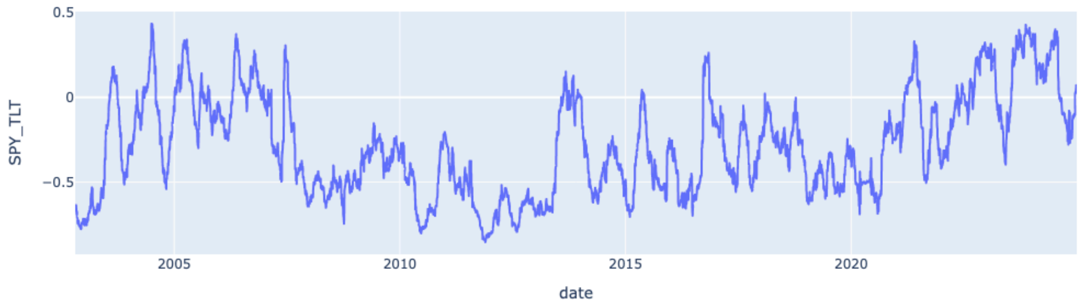
Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Comment

- Whilst the T-statistic is very high, the R^2 is very low (would expect this given how naive it is)

In [99]:
```
# Rolling correlation between log returns
rolling_corr = TLT_SPY_logret[["SPY","TLT"]].rolling(60).corr().unstack()
rolling_corr.name = "SPY_TLT"

fig3 = px.line(rolling_corr, x = rolling_corr.index, y = "SPY_TLT")
fig3.update_layout(autosize = True)
fig3.show()
```



Strategies

In [21]:
```
# Split data in half for training
train_TLT_SPY = TLT_SPY.iloc[:int(len(TLT_SPY)/2)]
```

Simple 60/40 portfolio

In [22]:
```
def TLT_SPY_0(tick, securities: List = ["SPY","TLT"], data = train_TLT_SP
    index_loc_tic: int = data.index.get_loc(tick)
    lb_window = data.iloc[index_loc_tic-lookback:index_loc_tic]

    sig: tuple = (0.6,0.4)


    return (sig, {"add_data":{}})
```

In [23]:
```
TLT_SPY0: Backtest = Backtest(name= "Basic 60/40", data = train_TLT_SPY)
```

In [24]:
```
TLT_SPY0.securities = ["SPY", "TLT"]
TLT_SPY0.assign_rebal_attr(fixed_ticks=31)
TLT_SPY0.assign_signals(trading_signal= TLT_SPY_0)
TLT_SPY0.lookback_window = 60
```

In [25]:
```
TLT_SPY0.calculate()
```

Out[25]:

date	SPY	TLT	PNL	SPY holdings	TLT holdings	SPY weights	TLT weights
2002-10-23	90.199997	84.330002	100.000000	0.665188	0.474327	0.6	0.4
2002-10-24	88.360001	85.070000	99.127057	0.665188	0.474327	0.6	0.4
2002-10-25	90.199997	85.379997	100.498041	0.665188	0.474327	0.6	0.4
2002-10-28	89.610001	85.260002	100.048666	0.665188	0.474327	0.6	0.4
2002-10-29	88.570000	86.370003	99.883372	0.665188	0.474327	0.6	0.4
...	...	...	...	...	...	...	..
2013-10-07	167.429993	106.139999	172.603651	0.617619	0.652131	0.6	0.4
2013-10-08	165.479996	106.169998	171.418859	0.617619	0.652131	0.6	0.4
2013-10-09	165.600006	105.320000	170.938669	0.621533	0.645828	0.6	0.4
2013-10-10	169.169998	105.489998	173.267327	0.621533	0.645828	0.6	0.4
2013-10-11	170.259995	105.459999	173.925422	0.621533	0.645828	0.6	0.4

2762 rows × 9 columns

In [100…

```
TLT_SPY0.plot()
```

In [27]:

```
TLT_SPY0.stats
```

Out[27]:

	Returns	Volatility	Sharpe
Results	0.075881	0.108423	0.699859

In [28]:
```
outperformance(TLT_SPY0)
```

```
Outperformance (Sharpe): 0.256
```

In [29]:
```
outperformance(TLT_SPY0, measure = "Returns")
```

```
Outperformance (Returns): -0.010
```

In [30]:
```
outperformance(TLT_SPY0, measure = "Volatility")
```

```
Outperformance (Volatility): 0.085
```

Comments

- Large drawdown in 2008 down to below initial investment amount
  - I suspect this is driven by equity underperformance, coupled with positive correlation between SPY and TLT in that period
- Adding TLT has certainly reduced annualised returns but reduced volatility by a greater amount

Dynamic weights - Simple regression

Idea: Use a simple regression model to inform the weights

In [31]:
```
sm.regression.linear_model.RegressionModel
```

Out[31]:
```
statsmodels.regression.linear_model.RegressionModel
```

In [32]:
```
def TLT_SPY_1(tick, securities: List = ["SPY","TLT"], data = train_TLT_SP
    index_loc_tic: int = data.index.get_loc(tick)
    lb_window = data.iloc[index_loc_tic-lookback:index_loc_tic]

    lb_log_ret = (np.log(lb_window)-np.log(lb_window.shift(1))).dropna(ho

    reg_model = sm.regression.linear_model.OLS(lb_log_ret.SPY, lb_log_ret
    reg_results = reg_model.fit()

    TLT_coef: float = reg_results.params.iloc[0]

    SPY_w = 1/(1+abs(TLT_coef))
    TLT_w = 1 - SPY_w

    sig: tuple = (SPY_w,TLT_w)

    return (sig, {"add_data":{"beta": TLT_coef}})
```

In [33]:
```
TLT_SPY1: Backtest = Backtest(name= "SPY TLT dynamic weights", data = tra

TLT_SPY1.securities = ["SPY", "TLT"]
TLT_SPY1.assign_rebal_attr(fixed_ticks=31)
TLT_SPY1.assign_signals(trading_signal= TLT_SPY_1)
TLT_SPY1.lookback_window = 60
```

In [34]:
```
TLT_SPY1.calculate()
```

Out[34]:

date	SPY	TLT	PNL	SPY holdings	TLT holdings	SPY weights	weig
2002-10-23	90.199997	84.330002	100.000000	0.406302	0.751234	0.366484	0.633
2002-10-24	88.360001	85.070000	99.808317	0.406302	0.751234	0.366484	0.633
2002-10-25	90.199997	85.379997	100.788792	0.406302	0.751234	0.366484	0.633
2002-10-28	89.610001	85.260002	100.458931	0.406302	0.751234	0.366484	0.633
2002-10-29	88.570000	86.370003	100.870247	0.406302	0.751234	0.366484	0.633
...	...	...	...	...	...	...	
2013-10-07	167.429993	106.139999	216.137248	1.200765	0.141439	0.930777	0.069
2013-10-08	165.479996	106.169998	213.800003	1.200765	0.141439	0.930777	0.069
2013-10-09	165.600006	105.320000	213.823885	1.232830	0.092223	0.954203	0.045
2013-10-10	169.169998	105.489998	218.240756	1.232830	0.092223	0.954203	0.045
2013-10-11	170.259995	105.459999	219.581769	1.232830	0.092223	0.954203	0.045

2762 rows × 9 columns

In [35]:
```
TLT_SPY1.output.describe()
```

Out[35]:

	SPY	TLT	PNL	SPY holdings	TLT holdings	SPY w
count	2762.000000	2762.000000	2762.000000	2762.000000	2762.000000	2762.0
mean	122.957346	96.676846	150.588186	0.840132	0.466456	0.6
std	20.538104	12.043123	31.747038	0.179354	0.246811	0.1
min	68.110001	80.650002	98.577690	0.406302	0.019623	0.3
25%	110.202497	88.412502	126.983783	0.719588	0.219667	0.5
50%	123.155003	92.115002	146.033318	0.861508	0.507014	0.6
75%	137.047501	101.902502	172.232852	0.967784	0.690264	0.8
max	173.050003	132.160004	222.722903	1.232830	0.851442	0.9

In [101…
```
TLT_SPY1.plot()
```

In [37]:
```
TLT_SPY1.stats
```

Out[37]:

	Returns	Volatility	Sharpe
Results	0.109538	0.104603	1.047181

In [38]:
```
outperformance(TLT_SPY1)
```

```
Outperformance (Sharpe): 0.603
```

In [39]:
```
outperformance(TLT_SPY1, benchmark = TLT_SPY0)
```

```
Outperformance (Sharpe): 0.347
```

In [40]:
```
outperformance(TLT_SPY1, measure = "Returns")
```

```
Outperformance (Returns): 0.024
```

Comments

- Significant improvement in Sharpe vs 60/40
- Still seeing sluggish performance between 2007 and 2009
- Interestingly, returns went up vs Buy-and-hold strategy

Controlling for Volatility - adding VIX

Idea: Volatility is a key variable that dictates the relationship between bonds and stocks - stocks and bonds move together (usually down) during periods of high volatility. The most naive way to do this is to incorporate VIX into the simple regression approach above.

In [41]:
```
# Import data from OpenBB
VIX_raw: pd.DataFrame = obb.obb.equity.price.historical(symbol= "^VIX", s
VIX = VIX_raw[["close"]].rename({"close":"VIX"}, axis = 1)
```

In [42]:
```
# TSV for "TLT, SPY, VIX"
TSV = TLT_SPY.merge(VIX, left_index= True, right_index = True)
```

In [43]:
```
# Calculate log returns
TSV_log_ret = (np.log(TSV) - np.log(TSV.shift(1))).dropna(how = "any")
```

In [44]:
```
reg2_model = sm.regression.linear_model.OLS(TSV_log_ret.SPY, TSV_log_ret[
reg2_results = reg2_model.fit()
```

```
reg2_results.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	SPY	**R-squared (uncentered):**	0.548
Model:	OLS	**Adj. R-squared (uncentered):**	0.548
Method:	Least Squares	**F-statistic:**	3422.
Date:	Wed, 02 Apr 2025	**Prob (F-statistic):**	0.00
Time:	20:43:07	**Log-Likelihood:**	19224.
No. Observations:	5644	**AIC:**	-3.844e+04
Df Residuals:	5642	**BIC:**	-3.843e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
TLT	-0.2033	0.012	-16.848	0.000	-0.227	-0.180
VIX	-0.1128	0.002	-74.260	0.000	-0.116	-0.110

Omnibus:	1516.638	**Durbin-Watson:**	2.147
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	92246.409
Skew:	0.393	**Prob(JB):**	0.00
Kurtosis:	22.790	**Cond. No.**	8.21

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Comment

- The stand-out observation here is the improvement in R^2

In [102…

```
fig4 = px.scatter_3d(TSV_log_ret, x = "SPY", y = "VIX", z = "TLT").update
fig4.update_layout(autosize = True, height = 450)
fig4.show()
```

Comments

- Clear negative relationship between VIX log returns and SPY
- Conscious that the absolute level of the VIX is important but changes in the VIX are also a key thing to consider

In [46]:
```
# Split data in half for training
train_TSV = TSV.iloc[:int(len(TSV)/2)]
```

In [47]:
```
def TLT_SPY_2(tick, securities: List = ["SPY","TLT"], data = train_TSV, l
    index_loc_tic: int = data.index.get_loc(tick)
    lb_window = data.iloc[index_loc_tic-lookback:index_loc_tic]

    lb_log_ret = (np.log(lb_window)-np.log(lb_window.shift(1))).dropna(ho

    reg_model = sm.regression.linear_model.OLS(lb_log_ret.SPY, lb_log_ret
    reg_results = reg_model.fit()

    TLT_coef: float = reg_results.params.loc["TLT"]

    SPY_w = 1/(1+abs(TLT_coef))
    TLT_w = 1 - SPY_w

    sig: tuple = (SPY_w,TLT_w)

    return (sig, {"add_data":{"beta": TLT_coef}})
```

In [48]:
```
TLT_SPY2: Backtest = Backtest(name= "SPY TLT VIX dynamic weights", data =
TLT_SPY2.securities = ["SPY", "TLT"]
TLT_SPY2.assign_rebal_attr(fixed_ticks=31)
TLT_SPY2.assign_signals(trading_signal= TLT_SPY_2)
TLT_SPY2.lookback_window = 60
```

In [49]:
```
TLT_SPY2.calculate()
```

Out[49]:

date	SPY	TLT	VIX	PNL	SPY holdings	TLT holdings	wei
2002-10-23	90.199997	84.330002	33.200001	100.000000	0.673078	0.465888	0.60
2002-10-24	88.360001	85.070000	34.029999	99.106295	0.673078	0.465888	0.60
2002-10-25	90.199997	85.379997	30.000000	100.489180	0.673078	0.465888	0.60
2002-10-28	89.610001	85.260002	31.070000	100.036162	0.673078	0.465888	0.60
2002-10-29	88.570000	86.370003	32.270000	99.853296	0.673078	0.465888	0.60
...	...	...	...	...	...	...	
2013-10-07	167.429993	106.139999	19.410000	201.114835	1.178912	0.035127	0.98
2013-10-08	165.479996	106.169998	20.340000	198.817015	1.178912	0.035127	0.98
2013-10-09	165.600006	105.320000	19.600000	198.928639	1.103492	0.152690	0.91
2013-10-10	169.169998	105.489998	16.480000	202.894054	1.103492	0.152690	0.91
2013-10-11	170.259995	105.459999	15.720000	204.092276	1.103492	0.152690	0.91

2762 rows × 10 columns

In [50]:
```
TLT_SPY2.output.describe()
```

Out[50]:

	SPY	TLT	VIX	PNL	SPY holdings	TLT ho
count	2762.000000	2762.000000	2762.000000	2762.000000	2762.000000	2762.0
mean	122.957346	96.676846	20.612252	143.172725	0.961823	0.2
std	20.538104	12.043123	9.611184	25.760202	0.130203	0.1
min	68.110001	80.650002	9.890000	92.349993	0.517940	0.0
25%	110.202497	88.412502	14.280000	123.023692	0.900262	0.1
50%	123.155003	92.115002	17.945001	139.980005	0.968337	0.2
75%	137.047501	101.902502	23.885000	160.298494	1.029907	0.3
max	173.050003	132.160004	80.860001	207.699934	1.242650	0.6

In [103…
```
TLT_SPY2.plot()
```

In [52]:
```
TLT_SPY2.stats
```

Out[52]:

	Returns	Volatility	Sharpe
Results	0.098864	0.148952	0.663728

In [53]:
```
outperformance(TLT_SPY2)
```

```
Outperformance (Sharpe): 0.220
```

In [54]:
```
outperformance(TLT_SPY2, benchmark = TLT_SPY1, measure = "Volatility")
```

```
Outperformance (Volatility): -0.044
```

Comments

- The underperformance between 2007 and 2009 has returned, bringing pnl down below the inital investment amount
- Although annualised returns are much better (than buy-and-hold and 60/40) - it's interesting to note the significant increase in volatility vs the simple regression approach above

Using T-statistic instead of regression coefficients

Idea: Shifitng from regression coefficients to T-statistics should structurally increase my TLT allocation where the relationship is statistically stronger. I still think controlling for VIX is appropriate given the rational above.

In [55]:
```
def TLT_SPY_3(tick, securities: List = ["SPY","TLT"], data = train_TSV, l
    index_loc_tic: int = data.index.get_loc(tick)
    lb_window = data.iloc[index_loc_tic-lookback:index_loc_tic]

    lb_log_ret = (np.log(lb_window)-np.log(lb_window.shift(1))).dropna(ho

    reg_model = sm.regression.linear_model.OLS(lb_log_ret.SPY, lb_log_ret
    reg_results = reg_model.fit()

    TLT_coef: float = reg_results.tvalues.loc["TLT"]

    SPY_w = 1/(1+abs(TLT_coef))
    TLT_w = 1 - SPY_w

    sig: tuple = (SPY_w,TLT_w)
```

```
    return (sig, {"add_data":{"beta": TLT_coef}})
```

In [56]:
```
TLT_SPY3: Backtest = Backtest(name= "SPY TLT VIX dynamic weights + T stat
TLT_SPY3.securities = ["SPY", "TLT"]
TLT_SPY3.assign_rebal_attr(fixed_ticks=31)
TLT_SPY3.assign_signals(trading_signal= TLT_SPY_3)
TLT_SPY3.lookback_window = 60
```

In [57]:
```
TLT_SPY3.calculate()
```

Out[57]:

	SPY	TLT	VIX	PNL	SPY holdings	TLT holdings	we
date							
2002-10-23	90.199997	84.330002	33.200001	100.000000	0.271440	0.895484	0.24
2002-10-24	88.360001	85.070000	34.029999	100.163208	0.271440	0.895484	0.24
2002-10-25	90.199997	85.379997	30.000000	100.940254	0.271440	0.895484	0.24
2002-10-28	89.610001	85.260002	31.070000	100.672652	0.271440	0.895484	0.24
2002-10-29	88.570000	86.370003	32.270000	101.384341	0.271440	0.895484	0.24
...	...	...	...	...	...	...	
2013-10-07	167.429993	106.139999	19.410000	202.422185	0.905975	0.479018	0.74
2013-10-08	165.479996	106.169998	20.340000	200.669907	0.905975	0.479018	0.74
2013-10-09	165.600006	105.320000	19.600000	200.371469	0.502791	1.106415	0.41
2013-10-10	169.169998	105.489998	16.480000	202.354517	0.502791	1.106415	0.41
2013-10-11	170.259995	105.459999	15.720000	202.869366	0.502791	1.106415	0.41

2762 rows × 10 columns

In [104…
```
TLT_SPY3.plot()
```

In [59]:
```
TLT_SPY3.stats
```

Out[59]:

	Returns	Volatility	Sharpe
Results	0.097991	0.10574	0.926717

In [60]:
```
outperformance(TLT_SPY3)
```

```
Outperformance (Sharpe): 0.483
```

In [61]:
```
TLT_SPY3.output.describe()
```

Out[61]:

	SPY	TLT	VIX	PNL	SPY holdings	TLT ho
count	2762.000000	2762.000000	2762.000000	2762.000000	2762.000000	2762.0
mean	122.957346	96.676846	20.612252	146.385920	0.447726	0.9
std	20.538104	12.043123	9.611184	36.464030	0.236360	0.3
min	68.110001	80.650002	9.890000	100.000000	0.130056	0.0
25%	110.202497	88.412502	14.280000	119.826836	0.285172	0.6
50%	123.155003	92.115002	17.945001	133.278059	0.368613	0.9
75%	137.047501	101.902502	23.885000	170.171382	0.492595	1.1
max	173.050003	132.160004	80.860001	226.487050	1.225345	1.5

In [62]:
```
outperformance(strategy= TLT_SPY3, benchmark=TLT_SPY0)
```

```
Outperformance (Sharpe): 0.227
```

Comment

- This approach manages the drawdown between 2007 and 2009 much better
- Slightly lower returns, but I expect this comes from the higher allocation to TLT

Out of sample performance

In [63]:
```
TLT_SPY3_OOS: Backtest = Backtest(name= "SPY TLT VIX dynamic weights + T
TLT_SPY3_OOS.securities = ["SPY", "TLT"]
TLT_SPY3_OOS.assign_rebal_attr(fixed_ticks=31)
```

```
TLT_SPY3_OOS.assign_signals(trading_signal= TLT_SPY_3)
TLT_SPY3_OOS.lookback_window = 60
```

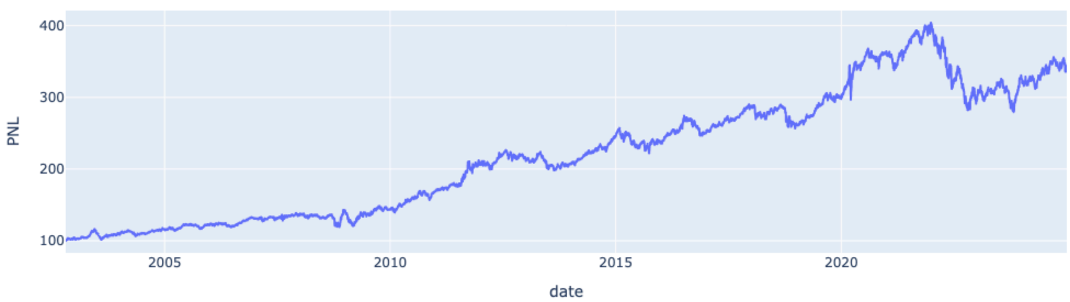
In [64]:
```
TLT_SPY3_OOS.calculate()
```

Out[64]:

date	SPY	TLT	VIX	PNL	SPY holdings	TLT holdings	weig
2002-10-23	90.199997	84.330002	33.200001	100.000000	0.271440	0.895484	0.244
2002-10-24	88.360001	85.070000	34.029999	100.163208	0.271440	0.895484	0.244
2002-10-25	90.199997	85.379997	30.000000	100.940254	0.271440	0.895484	0.244
2002-10-28	89.610001	85.260002	31.070000	100.672652	0.271440	0.895484	0.244
2002-10-29	88.570000	86.370003	32.270000	101.384341	0.271440	0.895484	0.244
...	...	...	...	...	...	...	
2024-12-24	601.299988	87.870003	14.270000	343.959453	0.529801	0.303974	0.920
2024-12-26	601.340027	87.820000	14.730000	343.965466	0.529801	0.303974	0.920
2024-12-27	595.010010	87.099998	15.950000	340.392955	0.529801	0.303974	0.920
2024-12-30	588.219971	87.800003	17.400000	337.008370	0.529801	0.303974	0.920
2024-12-31	586.080017	87.330002	17.350000	335.731752	0.529801	0.303974	0.920

5585 rows × 10 columns

In [105…
```
TLT_SPY3_OOS.plot()
```



In [66]:
```
TLT_SPY3_OOS.stats
```

Out[66]:

	Returns	Volatility	Sharpe
Results	0.082369	0.107901	0.763383

In [67]:
```
outperformance(TLT_SPY3_OOS)
```

```
Outperformance (Sharpe): 0.320
```

In [68]:
```
outperformance(TLT_SPY3_OOS, benchmark= TLT_SPY0)
```

```
Outperformance (Sharpe): 0.064
```

Comment

- The Sharpe here is clearly less attractive out of sample which is cause for concern.
- By virtue of being long only, this strategy will always underperform in periods of market distress as both stocks and bonds will go down

Analysis - PCA

In [69]:
```
TSV_log_ret_demeaned = (TSV_log_ret - TSV_log_ret.mean())
PCA0 = sk.decomposition.PCA(n_components= 3)
PCA0.fit(TSV_log_ret_demeaned)
PCA0.set_output(transform = "pandas")
res0 = PCA0.transform(TSV_log_ret_demeaned)
```

In [70]:
```
explained_var0 = pd.DataFrame(PCA0.explained_variance_ratio_)
explained_var0.index = res0.columns
explained_var0.columns = ["Explained_Variance"]
explained_var0
```

Out[70]:

	Explained_Variance
pca0	0.973652
pca1	0.016199
pca2	0.010149

In [71]:
```
eigen_vectors0 = pd.DataFrame(PCA0.components_)
eigen_vectors0.columns = res0.columns
eigen_vectors0
```

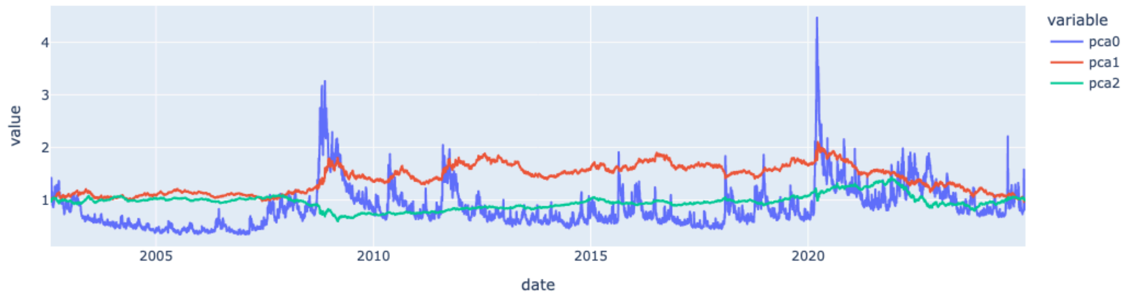
Out[71]:

	pca0	pca1	pca2
0	-0.119881	0.031893	0.992276
1	-0.562822	0.821171	-0.094390
2	0.817838	0.569791	0.080493

In [72]:
```
PCA_Factors0 = np.exp(np.cumsum(res0))
PCA_Factors0 = PCA_Factors0 / PCA_Factors0.iloc[0,:]
PCA_Factors0.index = TSV_log_ret_demeaned.index
```

In [106…

```
fig6 = px.line(PCA_Factors0, x = PCA_Factors0.index, y = PCA_Factors0.col
fig6.update_layout(autosize = True)
fig6.show()
```



In [74]:

```
adf_results_list: List = []
for x in range(len(res0.columns)):
    adf_results_list.append(sm.tsa.adfuller(res0.iloc[:,x]))

adf_results = pd.DataFrame(adf_results_list, index = res0.columns )
adf_results = adf_results[[0,2]]
adf_results.columns = ["T-statistic", "lag used"]
adf_results
```

Out[74]:

	T-statistic	lag used
pca0	-29.841186	8
pca1	-26.073277	8
pca2	-77.891742	0

Comment

- Looking at the PCA factors it seems pretty that pca0 is simply capturing the variance (or volaility) from the VIX

In [75]:

```
PCA1 = sk.decomposition.PCA(n_components= 2)
PCA1.fit(TSV_log_ret_demeaned.iloc[:,:2])
PCA1.set_output(transform = "pandas")
res1 = PCA1.transform(TSV_log_ret_demeaned.iloc[:,:2])
```

In [76]:

```
explained_var1 = pd.DataFrame(PCA1.explained_variance_ratio_)
explained_var1.index = res1.columns
explained_var1.columns = ["Explained_Variance"]
explained_var1
```

Out[76]:

	Explained_Variance
pca0	0.704377
pca1	0.295623

In [77]:

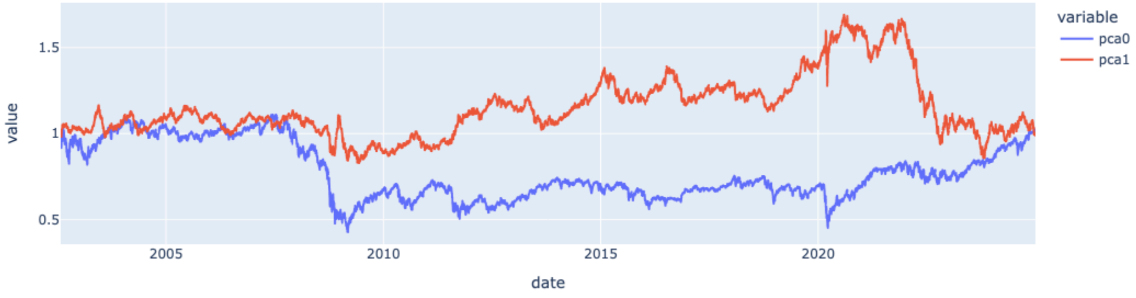
```
eigen_vectors1 = pd.DataFrame(PCA1.components_)
eigen_vectors1.columns = res1.columns
eigen_vectors1
```

Out[77]:

	pca0	pca1
0	0.904657	-0.426140
1	0.426140	0.904657

In [78]:
```
PCA_Factors1 = np.exp(np.cumsum(res1))
PCA_Factors1 = PCA_Factors1 / PCA_Factors1.iloc[0,:]
PCA_Factors1.index = TSV_log_ret_demeaned.index
```

In [107…
```
fig7 = px.line(PCA_Factors1, x = PCA_Factors1.index, y = PCA_Factors1.col
fig7.update_layout(autosize = True)
fig7.show()
```



In [80]:
```
adf_results_list: List = []
for x in range(len(res1.columns)):
    adf_results_list.append(sm.tsa.adfuller(res1.iloc[:,x]))


adf_results = pd.DataFrame(adf_results_list, index = res1.columns )
adf_results = adf_results[[0,2]]
adf_results.columns = ["T-statistic", "lag used"]
adf_results
```

Out[80]:

	T-statistic	lag used
pca0	-19.444369	15
pca1	-55.774543	1

Long short strategy - PCA mean reversion + T-statistic weightings

In [81]:
```
TSVF0 = TSV.merge(PCA_Factors0, left_index= True, right_index = True)
TSVF1 = TSV.merge(PCA_Factors1, left_index= True, right_index = True)
```

In [82]:
```
train_TSVF0 = TSVF0.iloc[:int(len(TSVF0)/2)]
train_TSVF1 = TSVF0.iloc[:int(len(TSVF0)/2)]
```

In [83]:
```
def TLT_SPY_4(tick, securities: List = ["SPY","TLT"], data = train_TSVF0,
    index_loc_tic: int = data.index.get_loc(tick)
    lb_window = data.iloc[index_loc_tic-lookback:index_loc_tic]
```

```
    lb_window1 = lb_window[securities + ["VIX"]]
    rolling_mean = lb_window.mean()

    lb_log_ret = (np.log(lb_window1)-np.log(lb_window1.shift(1))).dropna(

    reg_model = sm.regression.linear_model.OLS(lb_log_ret.SPY, lb_log_ret
    reg_results = reg_model.fit()

    TLT_coef: float = reg_results.tvalues.loc["TLT"]

    SPY_w = 1/(1+abs(TLT_coef))
    TLT_w = 1 - SPY_w


    if lb_window.pca2.iloc[-1] >= rolling_mean.pca2:
        sig: tuple = (-SPY_w,-TLT_w)
    elif lb_window.pca2.iloc[-1] < rolling_mean.pca2:
        sig: tuple = (SPY_w,TLT_w)
    else:
        sig = (0,0)


    return (sig, {"add_data":{"rolling_mean": rolling_mean}})
```

In [84]:
```
TLT_SPY4: Backtest = Backtest(name= "PCA long-short - mean-reversion", da
TLT_SPY4.securities = ["SPY", "TLT"]
TLT_SPY4.assign_rebal_attr(fixed_ticks=31)
TLT_SPY4.assign_signals(trading_signal= TLT_SPY_4)
TLT_SPY4.lookback_window = 60
```

In [85]:
```
TLT_SPY4.calculate()
```

Out[85]:

date	SPY	TLT	VIX	pca0	pca1	pca2	
2002-10-24	88.360001	85.070000	34.029999	1.076366	1.047775	0.980726	100.00
2002-10-25	90.199997	85.379997	30.000000	0.947726	1.051396	0.989112	100.765
2002-10-28	89.610001	85.260002	31.070000	0.982135	1.050761	0.985542	100.50
2002-10-29	88.570000	86.370003	32.270000	1.021768	1.065342	0.986139	101.226
2002-10-30	89.430000	86.339996	31.230000	0.988077	1.062718	0.990883	101.42
...	...	...	...	...	...	...	
2013-10-08	165.479996	106.169998	20.340000	0.903301	1.463159	0.851235	190.54
2013-10-09	165.600006	105.320000	19.600000	0.870515	1.458272	0.845096	190.26
2013-10-10	169.169998	105.489998	16.480000	0.731197	1.466830	0.848600	188.37
2013-10-11	170.259995	105.459999	15.720000	0.697290	1.467966	0.849468	187.88
2013-10-14	170.940002	104.610001	16.070000	0.712274	1.452194	0.849598	188.45

2762 rows × 13 columns

In [108…

```
TLT_SPY4.plot()
```

In [87]:

```
TLT_SPY4.stats
```

Out[87]:

	Returns	Volatility	Sharpe
Results	0.08735	0.105777	0.825798

In [88]:
```
outperformance(TLT_SPY4)
```

```
Outperformance (Sharpe): 0.382
```

Long short strategy - PCA mean reversion (bands)

Calculate confidence interval and trade if PCA is outside of bollinger band - towards mean

In [89]:
```
def TLT_SPY_5(tick, securities: List = ["SPY","TLT"], data = train_TSVF0,
    index_loc_tic: int = data.index.get_loc(tick)
    lb_window = data.iloc[index_loc_tic-lookback:index_loc_tic]

    lb_window1 = lb_window[securities + ["VIX"]]
    rolling_mean = lb_window.mean()

    lb_log_ret = (np.log(lb_window1)-np.log(lb_window1.shift(1))).dropna(

    reg_model = sm.regression.linear_model.OLS(lb_log_ret.SPY, lb_log_ret
    reg_results = reg_model.fit()

    TLT_coef: float = reg_results.tvalues.loc["TLT"]

    SPY_w = 1/(1+abs(TLT_coef))
    TLT_w = 1 - SPY_w

    k = sci.stats.t.ppf(0.975, lookback-1)


    upper_bound =  rolling_mean.pca2 + (k * lb_window.pca2.std() / np.sqr
    lower_bound = rolling_mean.pca2 - (k * (lb_window.pca2.std() / np.sqr


    if lb_window.pca2.iloc[-1]  >= upper_bound:
        sig: tuple = (-SPY_w,-TLT_w)
    elif lb_window.pca2.iloc[-1] < lower_bound:
        sig: tuple = (SPY_w,TLT_w)
    else:
        sig = (0,0)


    return (sig, {"add_data":{"rolling_mean": rolling_mean}})
```

In [90]:
```
TLT_SPY5: Backtest = Backtest(name= "PCA long-short - Mean-reversion", da
TLT_SPY5.securities = ["SPY", "TLT"]
TLT_SPY5.assign_rebal_attr(fixed_ticks=31)
TLT_SPY5.assign_signals(trading_signal= TLT_SPY_5)
TLT_SPY5.lookback_window = 60
```

In [91]:
```
TLT_SPY5.calculate()
```

Out[91]:

date	SPY	TLT	VIX	pca0	pca1	pca2	
2002-10-24	88.360001	85.070000	34.029999	1.076366	1.047775	0.980726	100.00
2002-10-25	90.199997	85.379997	30.000000	0.947726	1.051396	0.989112	100.765
2002-10-28	89.610001	85.260002	31.070000	0.982135	1.050761	0.985542	100.50
2002-10-29	88.570000	86.370003	32.270000	1.021768	1.065342	0.986139	101.226
2002-10-30	89.430000	86.339996	31.230000	0.988077	1.062718	0.990883	101.42
...	...	...	...	...	...	...	
2013-10-08	165.479996	106.169998	20.340000	0.903301	1.463159	0.851235	186.37
2013-10-09	165.600006	105.320000	19.600000	0.870515	1.458272	0.845096	186.09
2013-10-10	169.169998	105.489998	16.480000	0.731197	1.466830	0.848600	184.25
2013-10-11	170.259995	105.459999	15.720000	0.697290	1.467966	0.849468	183.77
2013-10-14	170.940002	104.610001	16.070000	0.712274	1.452194	0.849598	184.335

2762 rows × 13 columns

In [109…

```
TLT_SPY5.plot()
```

In [93]:

```
TLT_SPY5.stats
```

Out[93]:

	Returns	Volatility	Sharpe
Results	0.084177	0.103212	0.815581

In [94]:
```
outperformance(TLT_SPY5)
```

```
Outperformance (Sharpe): 0.372
```

Next steps

- Look at PNL attributions between the two assets to get a better idea of when each strategy performs well
- The relationship between bonds and equities is attenuated by many other factors (e.g Inflation, interest rates, Economic growth), so looking at macro data or approaches to identify regimes (Hidden Markov Models) may be interesting
- Investigate pairs trading between TLT and SPY (co-integration tests)
- Look at properly evaluating the role of volatility (both change and level)