Time Series Momentum

In [ ]:
```
import pandas as pd
import numpy as np
import pickle


import multi_factor_util
from multi_factor_util import get_performance_metrics, get_data_from_dict
```

Data Extraction

In [ ]:
```
file_name = "multifactor_data_2017_2022.bz2"
with open (file_name ,"rb")as file:
    data = pickle.load(file)
```

In [ ]:
```
close_price = get_data_from_dict(data, 'Close')
close_price.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...
2017-10-02	36.327709	69.086456	123.357384	147.940002	52.756409	157.477875	119.987679	47.959499	178.421280	139.316406	...
2017-10-03	36.485950	68.559082	122.273506	148.600006	52.560257	157.528351	120.694221	47.855000	180.400482	139.335342	...
2017-10-04	36.249767	68.711945	123.102348	147.949997	52.517624	158.672592	122.619507	48.272499	180.577911	138.711075	...
2017-10-05	36.700871	68.925949	124.067749	150.250000	52.858742	157.208633	122.981590	49.042500	180.111099	138.701599	...
2017-10-06	36.679630	69.162910	124.095093	151.119995	53.097553	156.342072	122.857933	49.479000	180.391205	138.890808	...

5 rows × 100 columns

Calculation of the Return

In [ ]:
```
rolling_returns = close_price.pct_change(252)
rolling_returns.dropna(inplace=True)
rolling_returns.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	U
2018-10-02	0.513209	0.079147	0.302198	0.838583	-0.121256	0.141963	0.083719	1.055182	0.462743	0.075165	...	0.445948	-0.0007
2018-10-03	0.524980	0.107990	0.321253	0.820390	-0.115486	0.137364	0.075730	1.040288	0.447323	0.079422	...	0.449805	0.0150
2018-10-04	0.507931	0.080538	0.295174	0.782426	-0.104299	0.110443	0.062739	0.977751	0.439358	0.087943	...	0.476468	0.0170
2018-10-05	0.465225	0.083381	0.280385	0.751880	-0.115032	0.123813	0.061884	0.926543	0.447970	0.074885	...	0.467229	0.0233
2018-10-08	0.462675	0.084589	0.270160	0.685349	-0.111616	0.138627	0.070590	0.884052	0.453423	0.078184	...	0.475212	0.0411

5 rows × 100 columns

In [ ]:
```
rolling_returns.iloc[0]
```

Out[ ]:
```
AAPL     0.513209
ABBV     0.079147
ACN      0.302198
ADBE     0.838583
AIG     -0.121256
           ...
VRTX     0.260790
VZ       0.155893
WBA     -0.017212
WFC     -0.033816
XOM      0.102101
Name: 2018-10-02 00:00:00, Length: 100, dtype: float64
```

Rebalance the Portfolio

In [ ]:
```
rebalancing = pd.DataFrame (index=rolling_returns.index)
rebalancing["is_start_month"] = rebalancing.index.to_series().dt.month != rebalancing.index.to_series().shift(1).dt.r
rebalancing.head()
```

Out[ ]:

	is_start_month
2018-10-02	True
2018-10-03	False
2018-10-04	False
2018-10-05	False
2018-10-08	False

In [ ]:
```
start_month_data = rolling_returns [rolling_returns.index.isin(rebalancing[rebalancing["is_start_month"]].index)]
start_month_data.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	
2018-10-02	0.513209	0.079147	0.302198	0.838583	-0.121256	0.141963	0.083719	1.055182	0.462743	0.075165	...	0.445948	-0.0
2018-11-01	0.351669	-0.098466	0.129278	0.391660	-0.317572	0.131794	0.110117	0.509070	0.284916	0.104770	...	0.285012	-0.0
2018-12-03	0.096724	0.010302	0.158697	0.421903	-0.262070	0.178854	0.160031	0.524807	0.290886	0.175482	...	0.287568	-0.0
2019-01-02	-0.052822	-0.041365	-0.065080	0.281500	-0.317304	0.135934	0.122845	0.316092	0.146876	0.089460	...	0.049195	-0.1
2019-02-01	0.009496	-0.250918	-0.019712	0.238386	-0.296200	0.035028	0.179338	0.120850	0.238264	0.193872	...	0.221352	-0.1

5 rows × 100 columns

Signal Generation

In [ ]:
```
monthly_trading_signal = start_month_data.applymap(lambda x: 1 if x > 0.05 else 0)
```

In [ ]:
```
monthly_trading_signal.fillna(0,inplace=True)
daily_trading_signals = monthly_trading_signal.reindex (rebalancing.index)
```

```
daily_trading_signals = daily_trading_signals.ffill()
daily_trading_signals.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WI
2018-10-02	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0
2018-10-03	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0
2018-10-04	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0
2018-10-05	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0
2018-10-08	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0

5 rows × 100 columns

Daily Return Calculation

In [ ]:
```
daily_returns = close_price.pct_change(axis=0)
portfolio_returns = daily_trading_signals.shift(1) * daily_returns
portfolio_returns.dropna(inplace=True)
portfolio_returns['Mean_Returns'] = portfolio_returns.apply(
    lambda row: row[row != 0].mean(), axis=1)

portfolio_returns.head()
```

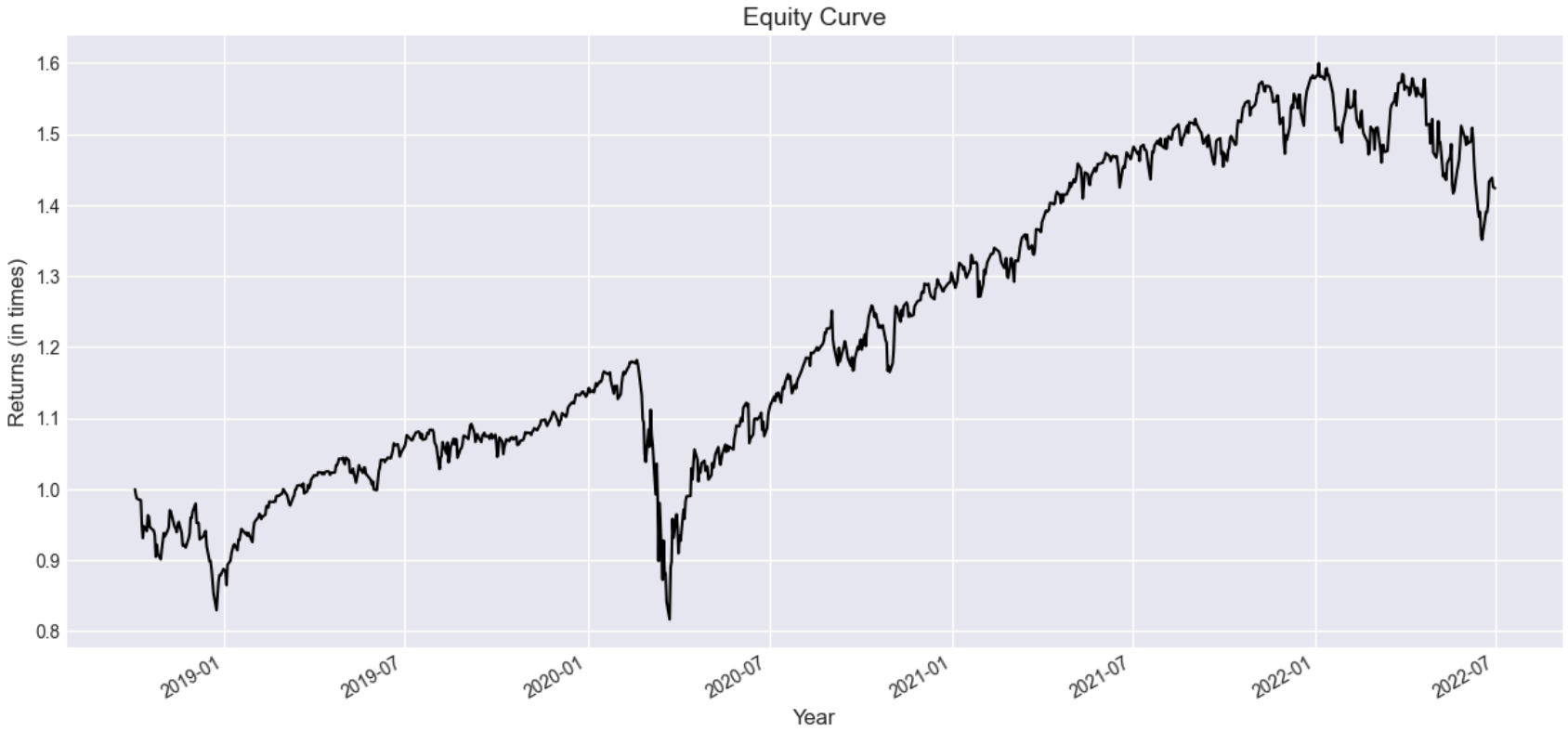
Out[ ]:

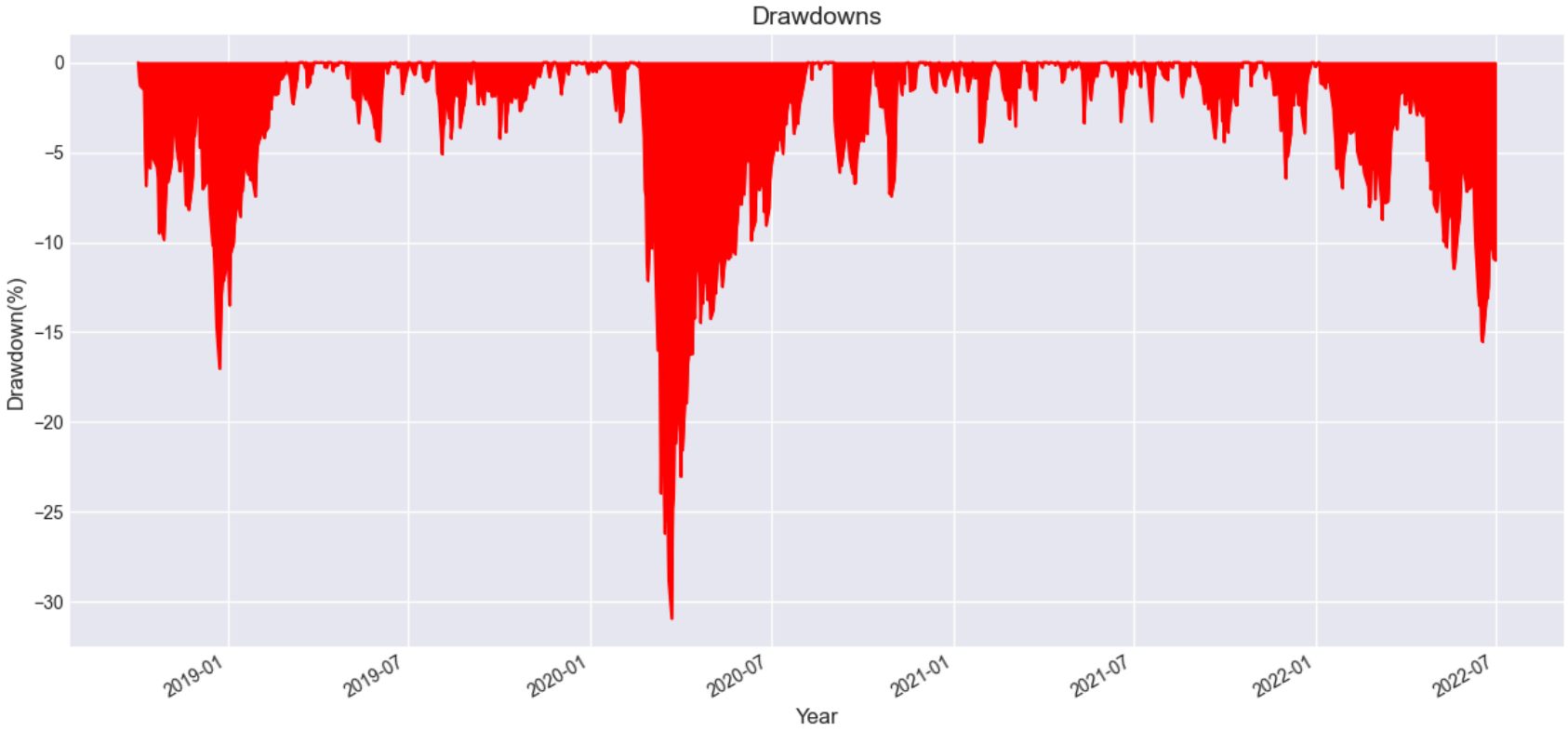
	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UPS	USB	
2018-10-03	0.012168	0.018890	0.005718	-0.005478	0.0	-0.003708	-0.001527	-0.009410	0.000434	0.004096	...	0.0	0.0	-0.0
2018-10-04	-0.017581	-0.022602	-0.013093	-0.025138	0.0	-0.016578	0.003683	-0.022194	-0.004525	0.003378	...	-0.0	0.0	-0.0
2018-10-05	-0.016229	0.005754	-0.003666	-0.001858	-0.0	0.002703	0.002146	-0.010354	0.003382	-0.012070	...	-0.0	-0.0	-0.0
2018-10-08	-0.002318	0.004556	-0.007767	-0.032406	0.0	0.007597	0.007185	-0.013352	0.005328	0.004437	...	0.0	0.0	0.0
2018-10-09	0.013854	-0.002531	-0.006474	-0.002081	-0.0	0.002092	0.012758	0.003165	0.006453	0.005825	...	-0.0	0.0	-0.0

5 rows × 101 columns

In [ ]:
```
portfolio_returns['Mean_Returns'].to_csv('raw_returns.csv')
```

In [ ]:
```
get_performance_metrics(portfolio_returns)
```


Equity Curve
Returns (in times)
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
2019-01
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
Year



	Strategy
CAGR	9.91%
Sharpe Ratio	0.55
Maximum Drawdown	-30.91%

Out[ ]:

	Strategy
CAGR	9.91%
Sharpe Ratio	0.55
Maximum Drawdown	-30.91%