Managerial Report: Investment Portfolio Analysis

```
import pandas as pd
import numpy as np

import plotly.io as pio
pio.templates.default = "plotly_white"

import plotly.express as px
import plotly.graph_objects as go

import pandas_datareader.data as web
```

Portfolio Composition

```
# Ticker Selection
# 3M, Target Corporation, Abbott, Apple
stock = ['MMM', 'TGT', 'ABT', 'AAPL']
```

Justification for Ticker Selection

```
from pandas_datareader import data
from datetime import datetime
from pandas_datareader import data as pdr
import yfinance as yfin
yfin.pdr_override()
```

```
import datetime
data =pdr.get_data_yahoo(stock,start='2010-1-1',end=datetime.today().strftime('%Y-%m-%d'))['Adj Close']
print(data.round(2))
```

```
[**********100%%**********]  4 of 4 completed
              AAPL     ABT     MMM     TGT
Date
2010-01-04    6.48   19.14   54.18   33.60
2010-01-05    6.49   18.98   53.84   33.73
2010-01-06    6.39   19.09   54.60   34.37
2010-01-07    6.38   19.25   54.64   34.79
2010-01-08    6.42   19.35   55.03   34.65
...            ...     ...     ...     ...
2023-12-07  194.27  104.05  103.28  135.19
2023-12-08  195.71  104.51  103.37  135.19
2023-12-11  193.18  106.22  103.16  136.76
2023-12-12  194.71  106.68  102.56  135.66
2023-12-13  197.96  107.25  104.19  138.38

[3511 rows x 4 columns]
```

Now we perform some simple calculations using the pandas data manipulation functions.

```
stock_ret = data.pct_change()
print(stock_ret.round(4)*100)
```

```
              AAPL   ABT   MMM   TGT
Date
2010-01-04    NaN   NaN   NaN   NaN
2010-01-05   0.17 -0.81 -0.63  0.37
2010-01-06 -1.59  0.56  1.42  1.91
2010-01-07 -0.18  0.83  0.07  1.23
2010-01-08  0.66  0.51  0.70 -0.40
...          ...   ...   ...   ...
2023-12-07  1.01 -0.85  0.48  1.36
2023-12-08  0.74  0.44  0.09  0.00
2023-12-11 -1.29  1.64 -0.20  1.16
2023-12-12  0.79  0.43 -0.58 -0.80
2023-12-13  1.67  0.53  1.59  2.01

[3511 rows x 4 columns]
```

```
#Calculate mean returns and covariances of all of the stocks
mean_returns = stock_ret.mean()
cov_matrix = stock_ret.cov()
print(mean_returns)
print(cov_matrix)
```

```
AAPL    0.001133
ABT     0.000585
MMM     0.000285
TGT     0.000556
dtype: float64
          AAPL       ABT       MMM       TGT
AAPL  0.000316  0.000102  0.000105  0.000096
ABT   0.000102  0.000187  0.000090  0.000077
MMM   0.000105  0.000090  0.000197  0.000087
TGT   0.000096  0.000077  0.000087  0.000304
```

Here we define the simulation parameters.

```
#Set the number of iterations to 10000 and define an array to hold the simulation results; initially set to all zeros
num_iterations = 10000
simulation_res = np.zeros((3+len(stock),num_iterations))
```

You will notice that we allocated an empty matrix to hold all of our results. There are many cleaner ways to do this using only pandas objects but this is a numerical approach.

```
simulation_res.shape
```

```
(7, 10000)
```

```
for i in range(num_iterations):
  #Select random weights and normalize to set the sum to 1
  weights = np.array(np.random.random(len(stock)))
  weights /= np.sum(weights)
  #Calculate the return and standard deviation for every step
  portfolio_return = np.sum(mean_returns * weights)
  portfolio_std_dev = np.sqrt(np.dot(weights.T,np.dot(cov_matrix, weights)))
  #Store all the results
  simulation_res[0,i] = portfolio_return
  simulation_res[1,i] = portfolio_std_dev
  #Calculate Sharpe ratio and store it in the array
  simulation_res[2,i] = simulation_res[0,i] / simulation_res[1,i]
  #Save the weights in the array
  for j in range(len(weights)):
    simulation_res[j+3,i] = weights[j]
```

```
sim_frame = pd.DataFrame(simulation_res.T,columns=['ret','stdev','sharpe',*stock])
```

Portfolio Optimization

Now we can locate the ideal portfolios from the simulation.

```
max_sharpe = sim_frame.iloc[sim_frame['sharpe'].idxmax()]
```

```
# Two key metrics guided our approach
# Spot the position of the portfolio with minimum Standard Deviation
min_std = sim_frame.iloc[sim_frame['stdev'].idxmin()]
print("The portfolio for max Sharpe Ratio:\n",max_sharpe)
print("\nThe portfolio for min risk:\n",min_std)
```

```
The portfolio for max Sharpe Ratio:
 ret       0.000964
stdev     0.014515
sharpe    0.066403
MMM       0.699925
TGT       0.173504
ABT       0.003877
AAPL      0.122695
Name: 9351, dtype: float64

The portfolio for min risk:
```

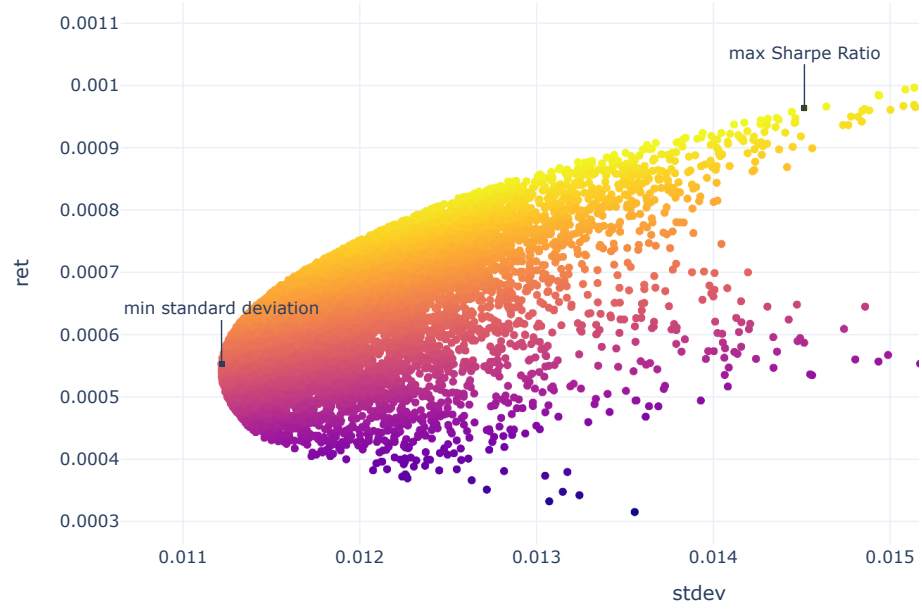
```
ret       0.000553
stdev     0.011218
sharpe    0.049303
MMM       0.124255
TGT       0.373525
ABT       0.315709
AAPL      0.186511
Name: 1032, dtype: float64
```

```
max_sharpe.stdev
```

```
0.014514986165118338
```

```
import plotly.express as px
fig = px.scatter(sim_frame, x = 'stdev', y='ret', color = 'sharpe')

fig.add_annotation(
            x=max_sharpe.stdev,
            y=max_sharpe.ret,
            text="max Sharpe Ratio")
fig.add_annotation(
            x=min_std.stdev,
            y=min_std.ret,
            text="min standard deviation")
fig.update_annotations(dict(
            xref="x",
            yref="y",
            showarrow=True,
            arrowhead=7,
            ax=0,
            ay=-40
))
fig.show()
```



Investment Strategy

```
# Cumulative Returns Analysis
from dateutil.relativedelta import relativedelta

todays_date = datetime.today().strftime('%Y-%m-%d')
two_years_ago_date = (datetime.today() - relativedelta(years=2)).strftime('%Y-%m-%d')
price_data = data.loc[two_years_ago_date:todays_date]
ret_data = price_data.pct_change()[1:]
```

```
# Asset weights for max sharpe ratio portfolio
sr_wts = [max_sharpe[x] for x in stock]
sr_weighted_returns = (sr_wts * ret_data)
sr_port_ret = sr_weighted_returns.sum(axis=1)
sr_cumulative_ret = (sr_port_ret + 1).cumprod()
```

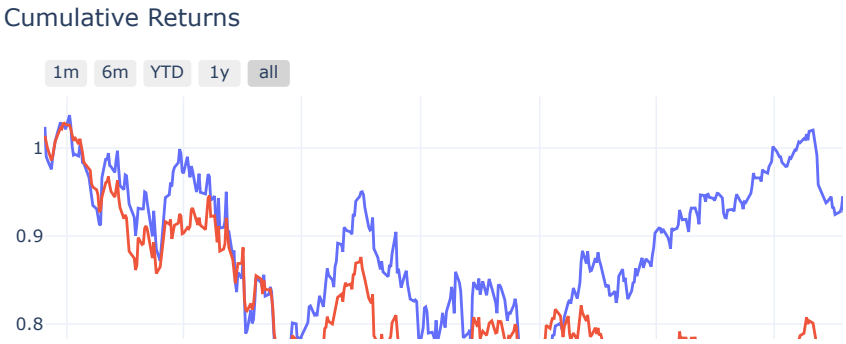
```
# Asset weights for max sharpe ratio portfolio
mr_wts = [min_std[x] for x in stock]
mr_weighted_returns = (mr_wts * ret_data)
mr_port_ret = mr_weighted_returns.sum(axis=1)
mr_cumulative_ret = (mr_port_ret + 1).cumprod()
```

```
# Create the cumulative return plots
fig = go.Figure()
fig.add_trace(go.Scatter(x=sr_cumulative_ret.index, y=sr_cumulative_ret,
                    mode='lines',
                    name="Sharpe Ratio Cumulative Return"))
fig.add_trace(go.Scatter(x=mr_cumulative_ret.index, y=mr_cumulative_ret,
                    mode='lines',
                    name="Min. Risk Cumulative Return"))

# Set title
fig.update_layout(
    title_text="Cumulative Returns"
)

# Add range slider
fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                     label="1m",
                     step="month",
                     stepmode="backward"),
                dict(count=6,
                     label="6m",
                     step="month",
                     stepmode="backward"),
                dict(count=1,
                     label="YTD",
                     step="year",
                     stepmode="todate"),
                dict(count=1,
                     label="1y",
                     step="year",
                     stepmode="backward"),
                dict(step="all")
            ])
        ),
        rangeslider=dict(
            visible=True
        ),
        type="date"
    )
)


fig.show()
```



Conclusion and Recommendations

The portfolio analysis demonstrates a strategic approach to investment, balancing returns and risk. The selected tickers, 3M, Target Corporation, Abbott, and Apple, offer a diversified mix with promising future returns. Investors seeking a balanced approach can consider the Max Sharpe Ratio Portfolio, while those prioritizing risk mitigation may opt for the Min Risk Portfolio.

Next Steps

Continued monitoring of market conditions and periodic reassessment of the portfolio composition are recommended. Additionally, exploring advanced predictive analytics methods can enhance future decision-making processes.

This investment portfolio is positioned to deliver strong returns while effectively managing risk, making it a compelling choice for investors seeking a well-rounded and strategic investment strategy.