Project 1 Abdullah Nuhin

May 15, 2024

Create a table showing constituent (stocks) risk analysis in the equal-weight portfolio analysis as of the current date

Column 1 – Ticker Column 2 – Portfolio Weight (equally weighted) Column 3 – Annualized Volatility (using trailing 3-months) Column 4 – Beta against SPY (using trailing 12-months) Column 5 – Beta against IWM (using trailing 12-months) Column 6 – Beta against DIA (using trailing 12-months Column 7 – Average Weekly Drawdown (52-week Low minus 52-week High) / 52-week High Column 8 – Maximum Weekly Drawdown (52-week Low minus 52-week High) / 52-week High Column 9 – Total Return (using trailing 10-years) Column 10 – Annualized Total Return (using trailing 10-years)

[68]:
```
import numpy as np
import pandas as pd
import yfinance as yf
```

[69]:
```
# Define the tickers in the portfolio
tickers = ['AAPL', 'MSFT', 'AMZN', 'GOOGL', 'GOOG', 'TSLA', 'NVDA']
etfs = ['SPY', 'IWM', 'DIA']
all_tickers = tickers + etfs

# Download historical data for the stocks and benchmarks
start_date = '2014-01-01'
end_date = '2024-02-24'
data = yf.download(all_tickers, start=start_date, end=end_date)['Close']

# Calculate the weights for an equal-weighted portfolio
weights = np.ones(len(tickers)) / len(tickers)

# Calculate the annualized volatility (standard deviation) using the trailing␣
↪3-months
volatility = data.rolling(window=63).std() * np.sqrt(252)
annualized_volatility = volatility.iloc[-1][tickers]

# Calculate the beta against SPY, IWM, and DIA using the trailing 12-months
def calculate_beta(stock_returns, benchmark_returns):
    cov_matrix = np.cov(stock_returns, benchmark_returns)
    return cov_matrix[0, 1] / cov_matrix[1, 1]
```

```
betas = {ticker: {} for ticker in tickers}
for etf in etfs:
    for ticker in tickers:
        stock_returns = data[ticker].pct_change().dropna()
        benchmark_returns = data[etf].pct_change().dropna()
        betas[ticker][etf] = calculate_beta(stock_returns[-252:],␣
 ↪benchmark_returns[-252:])

# Calculate the average weekly drawdown and maximum weekly drawdown
weekly_returns = data.resample('W').ffill().pct_change()
high_prices = data.resample('W').ffill().rolling(window=52).max()
low_prices = data.resample('W').ffill().rolling(window=52).min()
drawdowns = (high_prices - low_prices) / high_prices

average_weekly_drawdown = drawdowns[tickers].mean()
maximum_weekly_drawdown = drawdowns[tickers].min()

# Calculate the total return and annualized total return using the trailing␣
 ↪10-years
total_return = (data.iloc[-1][tickers] / data.iloc[0][tickers]) - 1
annualized_total_return = (1 + total_return) ** (252 / len(data)) - 1

# Create the risk analysis table
risk_analysis = pd.DataFrame({
    'Ticker': tickers,
    'Portfolio Weight': weights,
    'Annualized Volatility': annualized_volatility,
    'Beta against SPY': [betas[ticker]['SPY'] for ticker in tickers],
    'Beta against IWM': [betas[ticker]['IWM'] for ticker in tickers],
    'Beta against DIA': [betas[ticker]['DIA'] for ticker in tickers],
    'Average Weekly Drawdown': average_weekly_drawdown,
    'Maximum Weekly Drawdown': maximum_weekly_drawdown,
    'Total Return': total_return,
    'Annualized Total Return': annualized_total_return
})

# Display the risk analysis table
print("Constituent Risk Analysis:")
print(risk_analysis)
```

```
[**********100%%**********]  10 of 10 completed

Constituent Risk Analysis:
       Ticker  Portfolio Weight  Annualized Volatility  Beta against SPY  \
Ticker
AAPL     AAPL          0.142857              75.036247          1.058900
MSFT     MSFT          0.142857             260.415897          1.082603
AMZN     AMZN          0.142857             148.047732          1.444229
```

```
GOOGL   GOOGL          0.142857             90.982173          1.234526
GOOG     GOOG          0.142857             91.223376          1.249838
TSLA     TSLA          0.142857            407.683055          2.066520
NVDA     NVDA          0.142857           1567.899434          2.056704

        Beta against IWM  Beta against DIA  Average Weekly Drawdown  \
Ticker
AAPL            0.357478          0.936886                 0.338083
MSFT            0.231311          0.847048                 0.296833
AMZN            0.473002          1.036493                 0.370954
GOOGL           0.352973          0.833782                 0.293501
GOOG            0.358761          0.848763                 0.294750
TSLA            1.057497          1.791702                 0.514203
NVDA            0.532498          1.264386                 0.548508

        Maximum Weekly Drawdown  Total Return  Annualized Total Return
Ticker
AAPL                   0.202052      8.239347                 0.245420
MSFT                   0.141306     10.042519                 0.267531
AMZN                   0.158380      7.794130                 0.239364
GOOGL                  0.145280      4.168034                 0.176009
GOOG                   0.141924      4.240570                 0.177628
TSLA                   0.265042     18.184210                 0.338555
NVDA                   0.200285    197.781841                 0.686036
```

Create a table showing Portfolio Risk against the three ETFs: Column 1 – ETF Ticker Column 2 – Correlation against ETF Column 3 – Covariance of Portfolio against ETF Column 4 – Tracking Errors (using trailing 10-years) Column 5 – Sharpe Ratio (using current risk-free rate) Column 6 – Annualized V olatility (252 days) Spread (Portfolio V olatility – ETF V olatility)

[70]:
```python
# Define the tickers in the portfolio and the ETFs
tickers = ['AAPL', 'MSFT', 'AMZN', 'GOOGL', 'GOOG', 'TSLA', 'NVDA']
etfs = ['SPY', 'IWM', 'DIA']
all_tickers = tickers + etfs

# Download historical data for the stocks and benchmarks
start_date = '2014-01-01'
end_date = '2024-02-24'
data = yf.download(all_tickers, start=start_date, end=end_date)['Close']

# Calculate the returns for the portfolio and ETFs
portfolio_returns = data[tickers].pct_change().mean(axis=1).dropna()
etf_returns = data[etfs].pct_change().dropna()

# Calculate the risk-free rate (assuming 5% per year)
risk_free_rate = 0.05 / 252
```

```
# Calculate the correlations, covariances, tracking errors, Sharpe ratios, and␣
 ↪volatility spreads
correlations = {}
covariances = {}
tracking_errors = {}
sharpe_ratios = {}
volatility_spreads = {}

for etf in etfs:
    etf_ret = etf_returns[etf]
    correlations[etf] = portfolio_returns.corr(etf_ret)
    covariances[etf] = portfolio_returns.cov(etf_ret)
    tracking_errors[etf] = np.sqrt(((portfolio_returns - etf_ret) ** 2).mean())
    sharpe_ratios[etf] = (portfolio_returns.mean() - risk_free_rate) /␣
 ↪portfolio_returns.std()
    volatility_spreads[etf] = (portfolio_returns.std() * np.sqrt(252)) -␣
 ↪(etf_ret.std() * np.sqrt(252))

# Create the portfolio risk table
portfolio_risk_table = pd.DataFrame({
    'ETF': etfs,
    'Correlation': [correlations[etf] for etf in etfs],
    'Covariance': [covariances[etf] for etf in etfs],
    'Tracking Error': [tracking_errors[etf] for etf in etfs],
    'Sharpe Ratio': [sharpe_ratios[etf] for etf in etfs],
    'Annualized Volatility Spread': [volatility_spreads[etf] for etf in etfs]
})

# Display the portfolio risk table
print("\nPortfolio Risk Analysis:")
print(portfolio_risk_table)
```

```
[**********************100%%**********************]  10 of 10 completed


Portfolio Risk Analysis:
   ETF  Correlation  Covariance  Tracking Error  Sharpe Ratio  \
0  SPY     0.831305    0.000157        0.010089      0.063877
1  IWM     0.689484    0.000165        0.012639      0.063877
2  DIA     0.717507    0.000135        0.012053      0.063877

   Annualized Volatility Spread
0                      0.096865
1                      0.050527
2                      0.098594
```

Create correlation matrix showing the correlations between the equal-weighted portfolio, 3 ETFs, and your 7 stocks.

[71]:
```
# Define the tickers in the portfolio and the ETFs
tickers = ['AAPL', 'MSFT', 'AMZN', 'GOOGL', 'GOOG', 'TSLA', 'NVDA']
etfs = ['SPY', 'IWM', 'DIA']
all_tickers = tickers + etfs

# Download historical data for the stocks and benchmarks
start_date = '2014-01-01'
end_date = '2024-02-24'
data = yf.download(all_tickers, start=start_date, end=end_date)['Close']

# Calculate the returns for all assets
all_returns = data.pct_change().dropna()

# Calculate the returns for the equal-weighted portfolio
portfolio_returns = all_returns[tickers].mean(axis=1)

# Add the portfolio returns to the returns DataFrame
all_returns['Portfolio'] = portfolio_returns

# Calculate the correlation matrix
correlation_matrix = all_returns.corr()

# Apply color function to the correlation matrix
def color_correlation(val):
    color = 'black'
    if np.isclose(val, -1):
        color = 'red'
    elif np.isclose(val, 1):
        color = 'green'
    return f'color: {color}'

styled_correlation_matrix = correlation_matrix.style.applymap(color_correlation)

# Display the correlation matrix
print("\nCorrelation Matrix:")
display(styled_correlation_matrix)
```

```
[**********100%%**********]  10 of 10 completed


Correlation Matrix:
```

<pandas.io.formats.style.Styler at 0x7fcc8739cc50>