Intelligent Systems, Simulation, & Machine Learning -- Portfolio Optmization

1. Portfolio asset selection
2. Generate x(signifcant) # of portfolios
   1. [Design] portfolio generating function:
      1. <Params>
         1. [Req | List] Portfolio of Stock Symbols (Limit: 10)
         2. [Opt | Int] Number of Trials (Default: 1000)
         3. [Opt | Bool] Verbose (Default: False)

\*\*kwargs

* + 1. <Return>
       1. [DataFrame] Ranked Simulated Portfolio Data
          1. Indecies: Portfolio Efficiency Rank
          2. Headers: Stock Symbols
          3. Data Points:

Sharpe Ratio – Efficiency Metric

Expected Return

Expected Volatility

Asset Weights

* + - 1. Considerations
         1. Return datastructure that provides best continuity/transition to next step
         2. datas structure that best optimizes the speed / search algorthm efficiency of next step
         3. Descrcriptive summary of all intermediary transformative process
         4. Reduction of resource/time extending processes in creating portfolios.

1. Select Efficient Frontier Portfolios
   1. [Design] Optimal Portfolio Selection Algorithm
      1. Portfolio Classification Algorithm
      2. Build a Portfolio Classification Tree
         1. Objective:
            1. Sort each portfolio into a narrowly defined bin based on [ex. volatility | ex. return]. Once tree classification tree has been built, algortihm will select and return the best portfolio (sharpe ratio) in each classification bin. These portfolios will represent efficient/optimally constructed portolios. Portfolio selection will be determined by highest sharpe ratio which quantifies the efficiency of which the portfolio balances the risk/return trade off.
      3. Implement a Depth First Search Algorthm to Return the Optimal Portfolio From Each of the Child Node End Points.
      4. Sort and Index DataFrame;
         1. Identify Simulated Portfolio Range Metrics:
            1. Lower Bound
            2. Upper Bound
         2. Round Lower & UpperBound to Nearest Integer (Conservatively)
            1. Lower Bound - Round Up
            2. Upper Bound – Round Down
         3. Use Numpy “ARANGE” Function to Set Even Intervals Within Comprised Range.
         4. Select Optimal/Efficient Portfolio Within Each Classification Range
            1. If Initially Sorting by Expected Return, Optimal/Effiicient Portfolio Will Be The Portfolio With The Lowest Portfolio Variance.
            2. If Initially Sorting By Expected Volatility, Optimal/Efficient Portfolio Will Be Determined by Highest Expected Return.

\*\*Sorting First by Volatility or Expected Return Should Yield Same Results. However, Depending on Range and Intervals, Total Number of Selected Portfolios Could Vary Between Simulations as well as Within a Single Simulation Depending on Initial Sorting Method

* + - 1. Optimization:
         1. Portfolio should be indexed, and selection algorithm should truncate the input data to:

Expected Return

Expected Portfolio Variance

Search Algorithm paramters should accept a list/array and return the index numbers associated with each of the optimal portfolios.

Selected Portfolio Indecies would then be fed back into the original dataframe where the corresponding/associating portfolio data could then be retrieved

1. Interpreting & Visualizing the Results
   1. Minimal number of selected of portfolios should make interactive visualization relatively workable in JavaScript.
      1. Plotting on a scatter plot, hover-overs, etc.
      2. Creating an SVG object with a semi transparent fill in order to visualize the encapsulating area in which 99% of portfolios will fall.
      3. Linked, interactive dashboard plots that update depending on the portfolio selected on the main efficient frontier plot.
         1. Pie-Chart Breakdown (Sector)
         2. Minimal Table to Show (Symbol, Weight, ER, EV, Sector)
      4. Correlation Matrix for All Stocks Within Portfolio
         1. Apply a gradient heat map. (Red >> Green)
   2. Ability to Add Stocks or Remove Stocks From Portfolio and Dynamically Rerun Portfolio simulation and Update Dashboard
   3. Insight Metrics:
      1. Correlation Matrix
      2. Probability of Improvement (compolsition strength hueristic indicator)
      3. Scatter plot:
         1. Probability of Improvement
         2. Magnitude of Improvement
   4. Calculate user portfolio metrics in the same method and plot against efficent frontier
      1. Given portfolios current expected return and expected volatility:
         1. Output assessment metric and recommend:
            1. Portfolios that optimize expected return for current level of volatility
            2. Portfolio weights that mnimize volatility for the current level of return.
            3. Plot shift in portfolio performance on frontier plot.
            4. Output expected improvement in metrics over current portfolio.
   5. Additive/Dynamic/Generative Portfolio Inclusion Plotting
      1. Plot (Line Graph) How the stock will affect the expected return of the protfolio (Inclusive plot) as opposed to plotting just the stock itself
   6. \*\*Risk Adjusted Plot
      1. Graph Risk Adjusted Value vs Nominal Values
2. Backtesting Portfolio Performance
   1. Using optimized portfolio
      1. Calculate Portfolio Weight Average Return for Each Available Date
         1. Asset Performance Attribution – Visually show how portfolio weightings directly impact the portfolio return. Assets that are expected to do better will have be assigned higher weigtings within the portfolio which hopefully leads to better performance / holding period return. – Analagous in sports to allowing you better player play more minutes.
   2. Methodology:
      1. Input: Optimized Portfolio as Dictionary
         1. Key: Ticker Symbol
         2. Value: Portfolio Weight
      2. For Each Stock in Portfolio:
         1. Return Portfolio Weight Associated w/ Current Stock
         2. Return Closing Data (DataFrame)
         3. Convert Closing Data to Log Returns
         4. Use Apply & Lambda to Apply Weights to Stock Log Return Data
            1. End Result: Weighted Daily Return of Stock
         5. Merge Weighted Stock Return to Master DataFrame
      3. Append Column – Portfolio Daily Return to Master DataFrame
         1. Value of Column Will be Equal to Sum of Weighted Return for All Stocks w/n portfolio. \*\*Plot This Value\*\*
   3. Function Output: DataFrame w/ Weighted Returns for Each Asset in Portfolio & Column For Portfolio Weighted Average Return
   4. Plotting
      1. Plot Weighted Portfolio Return
         1. Boilinger Bands – Expected Movement (2 Standard Deviations)
         2. 14 Day Moving Average