***“Back\_Testor.py*” – Documentation**

# Description:

This file contains a Backtestor class which serves the purpose of backtesting trading strategies based on the signals generated from the Generate\_Signal class. Backtestor is connected to Risk Advisor and Performance Advisor to assess risk and performance metrics of the strategy. Backtestor runs on data stored in the database.

# Dependencies:

* Pandas
* numpy
* matplotlib
* copy
* Risk\_Advisor
* Performance\_Advisor
* DB\_Operator
* Any further dependencies of Risk and Performance advisors or DB\_Operator

# Class Function Descriptions:

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Inputs | Output | Description |
| Constructor | Asset (list),  Frequency (str),  Signal\_use (list/str),  start (str),  end (str),  Starting\_Equity | None | User can choose which asset to trade on, what period to trade on, and the frequency of trading.  Upon its construction, the Backtestor will go read the recorded price/ volume/ signal information from the historical database. Then it sets out several adjustable parameters including primary\_signal, weighting method, Risk cap etc. All of which can be modified in following functions. |
| Set\_Signal\_Rule | Primary (str),  Agree (int) | None | This function sets the rule of combining signal. It allows to specify which signal is primary and how many non-primary signal needs to agree before we take a trade.  Primary signal has same importance as non-primary signals combined. i.e. if Primary signals says to long we’d go long unless non-primary signals all agree to short. |
| Set\_Weight\_Method | Method (str) | None | This function sets rules about how to calculate values to trade when we made a trade decision. |
| Set\_Margin\_Rate | Margin (dict) | None | This function informs Back\_Testor the margin rate of all tradables through a dictionary. This is used later on to calculate the margin required to ensure we don’t run out of free cash. |
| Set\_Risk\_Cap | Value\_Cap (double),  Indiv\_Value\_Cap(double),  Volatility\_Cap (double),  Indiv\_Vol\_Cap (double) | None | This function allows users to set the value cap of investing in individual asset level and portfolio level. It also allows set the volatility cap on individual level and portfolio level |
| \_Signal\_Combine | None | None | This function combines signals based on the rule specified by user through above function. By default there’s no primary signal and no agree required. |
| \_Calculate\_Weight | Trade\_Info (pd.df),  Time (str),  Count (str),  Original\_Holding | None | This is an internal function that calculates today’s trading amount based on last day’s position holding and today’s new signal.  For example if we have long position and have a short signal then we close long position. Otherwise we add to the short position using the weight method named above, e.g. ‘Equal\_Weight’ that spends 1/20 of portfolio equity to trade.  After new positions calculated we run the new position through exposure control to ensure it ticks all risk criteria |
| \_Exposure\_Control | Holding (list),  Time (str),  Count (int),  Period (int) | Holding (list) | This is also an internal function that ensures all the risk caps set by user are met.  The calculation is based on last day’s equity and the variance covariance matrix of the past (Period) amount of days.  If any cap is exceeded, this function will downscale today’s trade to ensure it falls under risk radar again. Then it returns the new position to be executed. |
| run | None | None | This is the main user function that iterates through time and calculate every day’s holding, run them through exposure check, calculate returns and move on to the next day.  After portfolio holding is built for the whole period, internal risk advisor and performance advisor are constructed to assess the performance and risk of the named strategy. |
| Risk\_Demos | None | Pd.df  and  Bar plot  and  Line Plot | This is a wrapper function for the user to quickly see all the risk metrics of the tested strategy |
| Performance\_Demos | None | Pd.df  and  Bar plot  and  Line Plot | This is a wrapper function for the user to quickly see all the performance metrics of the tested strategy |