2023 Mar 15

Major Strategy Restructuring

Artemis was originally designed to be a high frequency trading (HFT) bot. However, due to certain FINRA and SEC account requirements for pattern day trading (Artemis will inevitably be considered a pattern day trader), the strategy for Artemis must change to a higher time frame so trades are less frequent and more calculated. (See issue #004)

This can be a very positive change, as Artemis will now have to consider a larger range of data and possibly move into swing trading.

Whereas before, when Artemis could focus on the price movement of one stock, it now should look at multiple potential target stocks at the same time and scan for entry points in each to avoid idleness.

This is where the change in strategy comes in, *since a strategy at its core really only has one job, no matter how complicated: give an entry point and an exit point*.

Now, rather than the entry point being based on one indicator, it should have multiple indicators acting as confirmations for price movement before entering a position.

For example, if price is moving sideways without much change but RSI is decreasing, there is a strong possibility that the price will breakout upwards. However, this should be confirmed with volume analysis and a look at previous support and resistance lines. Note Pfizer (PFE) at around 3:30pm on 2023 Mar 14 – had sideways movement with dropping RSI and still broke below previous resistance before breaking upwards until market close.

Going Forward:

When looking at candlestick charts, set the bar frequency to 1 hour. Add RSI indicator.

Look for ranges (a high price and a low price) which the price oscillates between.

ex.

Issues Log

#001: Slippage in time delay for data updates

Description:

Data is pulled from yfinance every 60 seconds, this constitutes the candlestick data that is stored in a csv file and used to inform trades. Initially, the driver code used a built in time.sleep(60) to deactivate the Artemis core for 60 seconds and wait for the next data update. (Data is pulled at 1 second past the start of the new minute: i.e. at 09:31:01)

However, the execution of code afterwards resulted in a roughly 4 millisecond delay, causing an offset in the number of milliseconds into the 1st second between each call of the update function. This resulted in a build up of slippage until the 60 second wait pushed into the 2nd second, causing the driver loop to miss an iteration.

Status:

RESOLVED; set time delay to 57 seconds so loop runs until the 1st second is reached and condition is met to execute strategy. 2 seconds are built in to buy function if called, this allows time for order to be executed on the market and pinged back to post_analytics() which updates self.last_buy_price. Now function is run within 1/10 of a millisecond difference of 4 milliseconds after 1 second.

#002: Dependence on TA-Lib module

Description:

Artemis' strategies rely on the TA-Lib module for technical analysis. This is only installable in an Anaconda environment due to the necessity of C-compilation (TL;DR it cannot be pip installed). This will need to be reflected in the Heroku requirements file.

Status:

IDEA; use heroku conda buildpack and conda reqs file

#003: MACD crossover delayed data results in missed swings

Description:

Most clear in SPY index chart with 1-minute candlesticks (though it should be the same for any scale). By the time the MACD histogram crosses the signal line, the price has already moved halfway through its movement. This, coupled with the reliance on price variable being set to the closing price of the last candlestick, results in entries that are much too late.

Status:

IDEA; possible but not sure if feasible: the price movement upwards usually starts when the MACD histogram is most negative, basically the minimum. If program can find when derivative of MACD histogram is 0 at local minimum, this may indicate when momentum will shift upwards to bull.

#004: Day trading requirements

Description:

As per https://www.sec.gov/files/daytrading.pdf, a pattern day trader executes 4 or more day trades within 5 business days. With data set to 1-minute candlesticks, this is guaranteed to trigger more

than 4 trades within 5 days. If flagged as a pattern day trader, the account must have a minimum equity of $25,000.

Status:

IDEA; Long term: funds will come later and will have to set risk coefficient to accommodate maximum trade value tolerance. $25,000 can sit in account with only 10% of that being used actively to trade, and risk coefficient would only be applied to that $2,500. Program would be set to not spend more than that, probably via restriction on principle, i.e. the program's internal account balance is set to 10% real account balance, or set to $2500 (or other value).

Short term: set scale of data higher and change strategy to swing trade. Make sure less than 4 trades are made per week, do not want to have assets stuck. Paper trading can still test high frequency strategies.

#005: Database operations are too computationally expensive

Description:

Originally, all stock data was being stored into a database file and rewritten every minute, all for data persistence. This led to very slow and inefficient operations just handling data, since it would need to be read and converted into a pandas dataframe for ta-lib anyways.

Status:

RESOLVED; database files eliminated. At start of driver in Artemis core, a new manager object is created with historic data in self.data attribute. This is the most recent data and is updated every self.period (the scale of data, right now probably 1 hour candlesticks). This ensures that the manager always has access to historic data up until the most recent candlestick.

#006: End of trading day program does not stop

Description:

Driver code currently checks time to see if the trading day has started and the time is between 9:30 am and 4:00pm. However, after 4:00pm, the program will continue running empty loops until the next day at 9:30am. This is computationally inefficient.

Status:

IDEA; use Heroku scheduler or have time.sleep() through the night when the time is 4:00pm, and time.sleep() through the weekend when it is Friday. Can also program in the holidays when the market is closed. This is not too serious, the dyno hours are probably being used on Heroku anyways.