Anastasia Kashkinova

3 January 2021

Object Oriented Programming Coursework for Midterm: MerklerexBot

- R1: Market analysis:

  - R1A: To retrieve the live OrderBook, inside its `getStats()` my bot uses the modified method of orderBook called `getLiveOrders()` which is based off of the `getOrders()` method. Instead of the current time it passes it a `deque` data structure which has up to 10 past timestamps. The function then operates on a larger dataset of asks to get more accurate stats.
  - R1B: The getStats() function gets both `currentEntries` and `liveEntires`(consisting of up to 10 past timeframes), and returns 5 float values (using tuple and tie in order to retrieve them) including the current timestamp `mean`, `minPrice` and `maxPrice` for the product, as well as `meanChange` and predicted `futurePrice` that's based off of the difference between current mean price and the historicalMean evaluated from previous timeframes. The future predicted price come from subtracting historical mean value from the current value, and using that difference divided by the number of analyzed timestamps to predict the likely future mean price change.

- R2: Bidding and buying functionality:

  - R2A: The bot bases its bidding decision off of the getStats() return values, if the predicted price is likely to grow, then the Bot places a bid for the given product, it uses the current minimumPrice to do that.
  - R2B: The bids are passed to the exchange's matching engine when the bot goes to the next timeframe.
  - R2C: The exchange matching engine returns accepted sales, if there's been no sales, none of our bids were processed. The bot uses a modified getOrders() function to get all the bids for the current timeframe that it placed with its username.
  - R2D: If none of our bids have been accepted by the matching engine, we withdraw our bids and remove them from the orders using removeOrder() method on the orderBook, that way they also do not skew market stats.

- R3: Offering and selling

  - R3A: The bot bases its asking decision off of the getStats() return values, if the predicted price is likely to go down, the Bot places an ask for the given product, using the current maximumPrice.
  - R3B: The asks are passed to the exchange's matching engine when the bot goes to the next timeframe.
  - R3C: The exchange matching engine returns accepted sales, if there's been no sales, none of our asks were processed. The bot uses a modified getOrders() function to get all the asks for the current timeframe that it placed with its username.
  - R3D: If none of our asks have been accepted by the matching engine, we withdraw our asks and remove them from the orders using removeOrder() method on the orderBook, that way they also do not skew market stats.

- R4: Logging

  - R4A: The wallet of assets is being recorded every timeframe and stored in assets.txt file
  - R4B: The bids and offers with the details including withdrawals are being stored in the ask-bid.txt file
  - R4C: The record of all successful sales is being stored in purchase-sale.txt and added on every timeframe.
  - There's also a general log.txt file that stores all of the above including some extra information like the stats for the given order on every timeframe, the files are cleared in the beginning of every simulation