**Stockie Feature List:**

Date Modified: 24 February 2020

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Action** |
| 1 | Make period an argument to backtester(). Minor problem. | Leave for now. |
| 2 | Build a model to predict the expected gain of the trade (NN) at the start. That way we can determine whether it is expected to do better than the stocks we are currently invested in. We can use the data from possible\_trades and PnL data frames as input for training the NN. This is a follow-on project of the final project project. | Follow-up project (IMPORTANT) |
| 3 | Merge backtester() into PnL class. Not a trivial change and should be done with care. Not urgent at this stage. | Leave for now. |
| 4 | Document code in backtest.py and make a call tree so we can see what is calling what. With that we can see if some functions naturally fit into classes. | Finish as part of final project. |
| 5 | Add code to backtester() to determine where it is spending most of its time. That will give us ideas where we can improve performance. | Finish as part of final project. |
| 6 | Backtester() sells and buys the same stock multiple times. It may improve performance if we save the data in a CSV file and then read it from disk as and when needed. That should reduce network load and waiting times. Will also make system more robust. | Leave for now. |
| 7 | The backtester() currently only considers “complete” trades (i.e. buy of ticker followed by a sell of ticker). Instead of dropping the buy signals at the end, keep them and put them in. a data frame. This needs to be incorporated in extract\_trades() function. | Finish as part of final project |
| 8 | Measure how well model does predicting into the future. To do that we need to address (7) | Follow-on project |
| 9 | To assist analysis, add buy\_date to entries in PnL.df. It currently only has one date per row and we need two. This will allow us to measure the number of days per actual trade and hence the average over all actual trades. | Finish as part of final project |
| 10 | The backtester() currently does not re-balance the stocks invested in. To manage the risk, it should rebalance things periodically. Does not seem a major problem looking at performance of trading system. | Follow-on project |
| 11 | Before building the model (balanced scorecard), check that the local minima are indeed the local minima and make sure the Close price does not go down the next day. This needs to be added to the determine\_minima\_n\_maxima(). Idem ditto for local maxima. | Follow-on project |
| 12 | Automatically sell stock if the gain is above a certain threshold (say 50% or 100%). That avoids losing money. It also quickly frees up money for new investments. This needs to be added to day\_close() function. | Follow-on project |
| 13 | Instead of just using print() statements, use logging and write the information to a log file. That way we can go back and see what happened during a particular run. | Follow-on project |
| 14 | Schedule job to run every night after stock market closes and every noon when the stock market is open. Start using good ol’ cron. Need to check how to do that as it has been a while. Needs (13) in place to ensure we can trace what happened while job ran. | Follow-on project |
| 15 | Add regression testing functionality. The easiest way to do that is to check whether backtester() generates the same data frame contents for a fixed period. That allows us to make sure changes don’t break the code. | Follow-on project |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |

NB: highlighted items to be completed as part of final project. Other items will be considered after putting presentation together and planning out feature (2) on the list.