Clustered Reinforcement Learning for Trading (RL4T)

Packages Used:

Pandas, Numpy, TA, Sklearn, Matplotlib

Methods Useds:

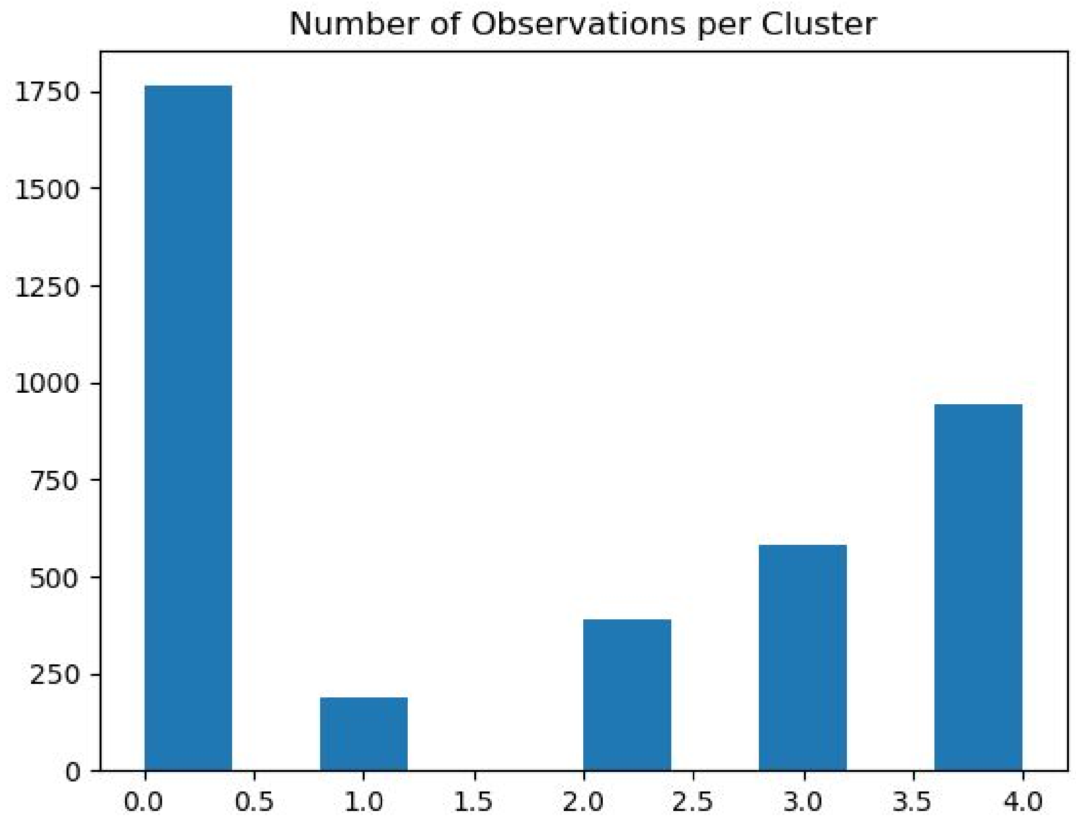
Q Learning

Description:

Within this project I am clustering a set of metrics (technical indicators only) to minimize the number of possible states to n_clusters, and for each time step (in the given publicly traded stock data) I generate its “state” via KMeans and determine whether or not an individual should buy/don’t buy based on the greedy method.

Question #1:

What is the distributions of observations for 5 clusters?



Not uniform which means the Q Learning model will be exposed to different state-reward pairs differently.

Question #2:

Settings for basic Q Learning model?

- Description
    - For each time step, determine to buy/don't buy then adjust Q table (for each state aka "cluster") based on percentage change of next 5 day high.
    - If buy is the suggested action, then only increase by 1.01.
    - If trigger isn't actually reached, then only decrease by -0.05.
- Details
    - Learning Rate: 0.01
    - Epsilon: 1 / Number of actions (aka ½)
    - Number of Clusters: 5
    - Iterations: 10

General Observation

- The total reward for each cluster depends on how many actual observations were seen in that cluster.
- The reward for buy/don't buy depends on whether or not the random number hits that action.
- So, in general, this model is good for learning through trial and error but its actions are highly dependent on rewards that are only given by each observation of a particular state. The learning rate helps with this but if the state is only seen a few times then its actions will be greedy towards the few it has seen vs adjusting all possible actions through what actually occurred and the estimated opportunity cost respectfully.
- Separately the Q model doesn't incorporate a decay in memory so that the agent can stay adaptive in an evolving environment.
- Next Steps
    - Incorporate an opportunity cost reward policy.
    - Incorporate decay in memory
    - Incorporate true positive percentage against total reward
    - See sensitivity in total reward given a set of clusters (5, 10, 25, 50, etc)