SmartTrader

ISS SLS PROJECT REPORT

ONN WEI CHENG A0092201X

YANG JIESHEN A0003901Y

NIRAV JANAK PARIKH A0213573J

ISA-PM-SLS-2021-01-09-IS02PT

TABLE OF CONTENTS

APPENDICES

1. PROBLEM STATEMENT

1.1. PROBLEM DESCRIPTION

Large volumes of financial instruments are traded daily on exchanges and a growing majority of these are traded using automated trading algorithms. An automated trading strategy that maximizes profit is highly desirable for investors, mutual funds and hedge funds.

These trading algos can be broadly categorized into:

a) Passive / Fundamental Trading which is primarily a long-term strategy based on market and security fundamentals or passively benchmark index trading.

b) Active / Price Action Trading which is based on buying and selling securities based on short-term movements to profit from the price movements and relies on historical prices (open, high, low, and close) to make trading decisions.

This project will focus on the Price Action Trading which is driven on the characteristics of a security's price movements. Since it ignores the more subjective fundamental factors and focuses solely on recent and past price data, the Price Action Trading Strategy is more conducive to Machine Learning.

There are essentially two contrasting approaches to developing price trading strategies:

a) Model based: This approach attempts to create a mathematical model of the market thru representative variables, such as mean price and corelations, to create a simplified representation of the true complex market model. Some examples of this are trend-following, mean reversion and arbitrage strategies.

b) Model free: Here there is no attempt to model the market, rather it looks at price patterns and attempt to fit an algorithm to it. There is no attempt to produce a causal analysis or explanation – just an attempt to identify patterns that will repeat in the future. Some examples will include technical analysis, charting and candle patterns.

This project will focus on the Model free approach and leverage Reinforcement Learning (RL) to develop an automated trading agent (Smart Trader) driven solely on historical data and derived technical features.

1.2. APPROACH TO SOLUTION

While RL has traditionally proved its capabilities in learning to play games, RL presents a unique opportunity to model the complexities of the trading strategy as a game in which an agent can be trained to maximize its total reward from the market.

Trading is a continuous strategy in which decisions are taken sequentially and which cumulatively result in a profit or loss. As such, there are no specific label associated with any historical data and this is a major problem for traditional Supervised Learning (SL) approaches. On the other hand, RL can use feedback from its own action and experiences, such as the rewards returned by the stock market to learn an optimal trading strategy.

RL also has different learning goals from SL. While SL learns to make the best predictions (Classification or Regression), RL learns a policy for actions that would maximize its long-term cumulative reward, which is the goal of trading.

Trading is a continuous activity and has no defined endpoint. Also, trading is a partially observable environment as we do not have complete information about the traders in the market. Since we have a Partially Observable Markov Decision Processes (POMDP) and we don't know the reward function and transition probability, we use Q-Learning as our approach.

Q-learning is a model-free reinforcement learning algorithm. It is a value-based, off policy method that supplies information to an agent for the next trading action (BUY, HOLD, SELL). The Q-learning function learns through the epsilon greedy algorithm by exploiting known knowledge and occasionally exploring random actions that are outside the current policy.

Q-value refers to the action quality and is stored in Q-table with dimensions [state, action]. The Q-value is the maximum discounted future reward when an action, a, is performed in a state, s. The objective of Q-learning is to learn the optimal value of the Q-table that will provide the maximum reward at the end of the n number of training cycles or iterations. It functions well without the reward functions and state transition probabilities.

Deep Q-Learning replaces the Q-table and approximates the Q-value with a neural network as a function approximator. At run-time, the agent will evaluate every action from the current state and select the one with the maximum Q-value.

2. MODEL OVERVIEW AND IMPLEMENTATION

The analysis and work done in this project improves on an existing Deep Q-Network reinforcement learning model available here.

The model divides the data set into the training dataset and the test dataset. The agent learns using the training dataset and is used to predict buy/sell/hold actions to maximize profits.

A deep neural network is used to predict the action value based on input states. Due to the nature of stock markets, prices are somewhat correlated sequentially. Hence, an experience relay is deployed to stabilize the network. The agent uses a memory to store batches of historical data which is parsed in randomly to train the neural network. The target Q network is updated periodically to further reduce correlation.

The neural network structure is displayed in Appendix I. The general design is to have a deeper network with more neurons to handle the complexity of a trading scenario.

In this project, we are focused more on designing the environment and evaluating the agent for better representation and performance comparison.

2.1. STATE

A **state** is the description on the current environment that the agent finds itself. The details on this state are provided to the agent by the environment. It is important that this state has the Markov Property. This means that the current state information has sufficient and necessary information for the agent to accurately predict expected next rewards and next states given an action, without the need for any additional information.

In our trading model, our OHLC daily price data is a time series and is not stationary. In other words, there is a trend embedded within the time-series. We make the daily OHLC price data by calculating the daily returns (today's price – yesterday's price). This makes the returns data stationary, and this is what we use in our model to describe current state.

Furthermore, given the inherent trends in the price data, a single day of price returns is not sufficient to describe the complete state of the market. Hence, we use a sliding window of n days of historical returns data (defined by parameter historical_n = 90) in the state description to provide the agent the full view of market movements in the recent past.

We also explored the usage of various technical analysis indicators to ensure the state fulfils the Markov property. A brief explanation of the technical analysis used is as followed:

- **Trade Volume**: This is the total volume over the entire trading day
- **SMA-14**: This is the simple moving averages over a 14-day rolling window
- **EMA-14**: This is the exponential moving average over a 14-day rolling window
- **Bollinger Bands**: Bollinger band is used to analyse if stock it is overbought or oversold. They consist of three bands:
  - 20-day moving average
  - Upper Bollinger band - 2 standard deviations above
  - Lower Bollinger band - 2 standard deviations below
- **Stochastic Oscillator**: This is a momentum indicator which compares the recent closing price of an asset to a range of its prices over a specific period of time. It is an

indicator which does not follow price or volume but signifies the speed, or momentum of a price. Helps to identify a bullish or bearish trend in the market.

In final model, the full state returned by the environment on every step is the summation of the mark-to-market returns of the strategy's position, daily returns for the past 90 inclusive of today and the list of technical analysis indicators. This state is input into the NN network to model the Q-value for the state-action value.

2.2. **ACTIONS**

Actions is the set of all possible moves the agent can make from a given state. For our trading agent, there are only three permissible action: BUY, HOLD or SELL.

In order to affect the off-policy learning the model employs an Epsilon Greedy Policy to find the balance between the model's Explore vs Exploit behaviour. The model defines a starting epsilon value (epsilon = 0.9) which starts decreasing by a pre-defined value (epsilon_decrease = 1e-3) after a predefined number of step (start_reduce_epsilon = 200) till it reaches the specified minimum value (epsilon_min = 0.1). This allows the model to start with a high exploration rate and hence learn quickly during the initial steps and gradually reduce exploration and increase exploitation as the model starts learning the optimal policy.

In the current model, if the agent signals the SELL action when there is no open position, the environment just ignores the action and returns zero rewards. This is because the current environment only allows a sell of an existing long position. However, the SELL signal from the agent is still a valid signal which indicates that the best action from this state is to sell. In future enhancement the environment will be enhanced to support short-selling. Short-selling refers to the selling the stock without a position (essentially borrowing stock and selling it with the expectation to buy it back when the market drops). With this enhancement the environment will act on the agent's SELL signal by essentially shorting the stock and return a corresponding reward.

For each of the actions, the environment takes a distinct set of steps to calculate the current rewards. These reward calculations will be described in the next section.

2.3. **REWARDS**

A **reward** is the feedback provided by the environment that measure the success or failure of an agent's actions in a given state. For our environment, the rewards are calculated distinctly for each trading action (BUY, HOLD, SELL).

The reward for each action is primarily the returns generated for each action. Furthermore, we have introduced the concepts of Transaction Cost and Holding Cost to account for the real-life trading costs.

- **Transaction cost** = the cost of executing a trade and is deducted from the rewards of the BUY action. This is a percentage of the cost of the stock bought at the BUY action.

- **Holding Cost** = Financing cost of holding a position over a period. This is deducted at every time step when an action is taken. This is a percentage of the cost of the current position bought at the BUY action.

Hence the rewards from the environment for each action are as follows:

- Reward for BUY action = Holding cost + Transaction cost of the stock bought
- Reward for HOLD action = Holding cost of the current position
- Reward for SELL action = Holding cost + Profit or Loss from selling the open position.

The environment was also constructed with the possibility of clipping the rewards at +/- 1 which we attempt to vary in our iterative improvements. In the final model, the clipping of rewards is set to false, which means that profit or loss from sale of a stock is immediate equated as the rewards without clipping. In addition, the agent (not the environment) also incorporates a discounting function (gamma = 0.97) to recognize the diminishing value of future rewards and adjust for this in the learning model.

2.4. **ITERATIVE IMPROVEMENTS**

This section illustrates the different improvements made iteratively to make the model more realistic to achieve the final deep Q learning model described above.

2.4.1.**PHASE 1 - IMPROVED BASE MODEL**

The improved base model includes the various improvements and optimisation that was made, including changes to rewards functions and training logic, among others.

Initially, the rewards per time step was clipped, with any positive profits resulting in a +1 reward, while any negative profits resulting in a -1 reward. This was not indicative of the returns made by the strategy and therefore we equated the profits/loss as the rewards. In other words, clipping of rewards was set to false.

The states of the model consist of previous prices of the stock (history) and initialised with zeros. In the initial model, the prices of the stock are added at each time step during training, resulting in many states in the memory without the full history of prices, especially at the start of each episodic run. We felt that this was not indicative of the real environment, as most stock will have their prices available and can be used for prediction. Therefore, we ensure that the previous prices are fully initialised before adding the transition to the memory buffer and used for training.

In the improved base model, we only made use of the closing price of the profit/loss of the portfolio and the closing price of the past 90 days as the states for model training. Transaction cost and holding cost is set at 0.

2.4.2. **PHASE 2 – ADDING TECHNICAL ANALYSIS**

In Phase 2, we seek to improve on the reinforcement learning model, by adding technical analysis variables as state variables. It includes indicators such as trade volume, moving averages, bollinger bands and stochastic oscillator. As this are typical technical analysis variables used by traders in the financial markets, the intuition that the addition of such variables allows for the better derivation of the action values and consequently the best actions. Transaction cost and holding cost is also set at 0.

2.4.3. **PHASE 3 – ADDING TRANSACTION AND HOLDING COST**

In Phase 3, we seek to make the model realistic. To prevent the constantly buying and selling stocks without the consideration of transaction cost, a 1% transaction cost is added to the model. Similarly, to consider the impact of opportunity cost of holding a portfolio, a daily holding cost of 0.1% is added to discourage excessive holdings and promote efficient cash management strategies.

3. MODEL EVALUATION

Is having a strategy that can provide us with a profit good enough for us to say that the model has been successful? If not, what metrics and benchmarks do we have to use to compare the performance of the strategy and consequently the performance of the reinforcement learning model? This section will explore the metrics to be used and use those metrics to compare the performance of our strategy.

3.1. METRICS USED FOR EVALUATION

"The riskier the investment the higher the returns" is an adage that still holds true very much today. Similarly, our metrics used to evaluate our strategy has to consider the balance between the risks and rewards. To this, we propose 2 different benchmarks that we can use to compare the returns of our strategy.

Risk-free returns

Investment returns other than rewarding the undertaking of additional risk, also has a function of rewarding the opportunity cost of investment. Risk-free returns in way signifies the minimum returns that strategy needs to provide to cover the opportunity cost of making that investment, notwithstanding the risk that is undertaken by the strategy. Risk-free returns are typically estimate by using the treasury bonds issued by countries with highly rated credit ratings. In our case, we will be using the US 10-year treasury bond rates.[1]

Expected returns

To understand the expected returns to take on a particular amount of risk, we refer to capital asset pricing model, more commonly known as the CAP-M model.

$$Expected\ returns = Risk\ free\ returns + Beta(Market\ returns - Risk\ free\ rate)$$

In the CAP-M Model, other than risk free returns, expected returns also depend on market returns and beta. Market returns measures how well the market is performing while, beta measures how much additional volatility does the stock have against the market, which is also the systemic risk of the stock.

The intuition behind the CAP-M model, is that the investors are rewarded based on the additional systematic risk that taken vis-à-vis the market. This also suggests that unsystematic risk should not be rewarded as the risk can be mitigated by diversification. Our reference for the performance of the market will be the Vanguard S&P 500 Exchange Traded Fund, which mimics the S&P 500 with an accuracy of 99.7%.[2] A 5-year monthly beta is used in this case.[3]

[1] https://www.macrotrends.net/2016/10-year-treasury-bond-rate-yield-chart
[2] https://personal.vanguard.com/pub/Pdf/sp968.pdf?2210142649
[3] https://sg.finance.yahoo.com/quote/GOOG/key-statistics?p=GOOG

Strategy returns

The returns from the strategy will be the pure profit or loss derived by the strategy excluding transaction and holding cost. Although, transaction and holding cost are part of the rewards of the reinforcement learning model, it is not included in the returns of the strategy. This was done to ensure that the return of the strategy is comparable to the two benchmarks that we have proposed, as transaction and holding cost are also not factored into the benchmarks.

Maximum drawdown

Although, risk-free and expected returns are good benchmarks to compare the returns of the strategy, it will also be important to understand the maximum risk that is undertaken by the strategy. This is especially important as the principal in the investment in stocks is limited and not protected. Therefore, in addition to the returns benchmark, we propose an additional metric of maximum drawdown. Maximum drawdown refers to the maximum loss (mark-to-market) that is incurred by our strategy based on the current portfolio holdings.

3.2. TRAINING RESULTS BETWEEN ITERATIVE IMPROVEMENTS

All training results from following phases are analysed by changing parameters to the environment in ***SmartTrader_Analysis.ipynb***. Utility functions and classes are found in ***SmartTraderLibrary.ipynb***

3.2.1. PHASE 1 - IMPROVED BASE MODEL

The results of the improved base model are as shown in **Figure 1** and **Figure 2**. The cumulative profit of the strategy is almost equal to the expected stocks returns calculated from CAP-M for most of the period with some out performance in Q2 2017. However, it does outperform the risk-free return. This translates to 34% profits versus the average portfolio size.

The strategy also had a maximum drawdown of USD $623 with an average portfolio size of USD $3365, signifying a 18% dip in average portfolio size in its lowest performing times. **Figure 3** shows the suggested actions by the phase 1 model for both training and testing data.

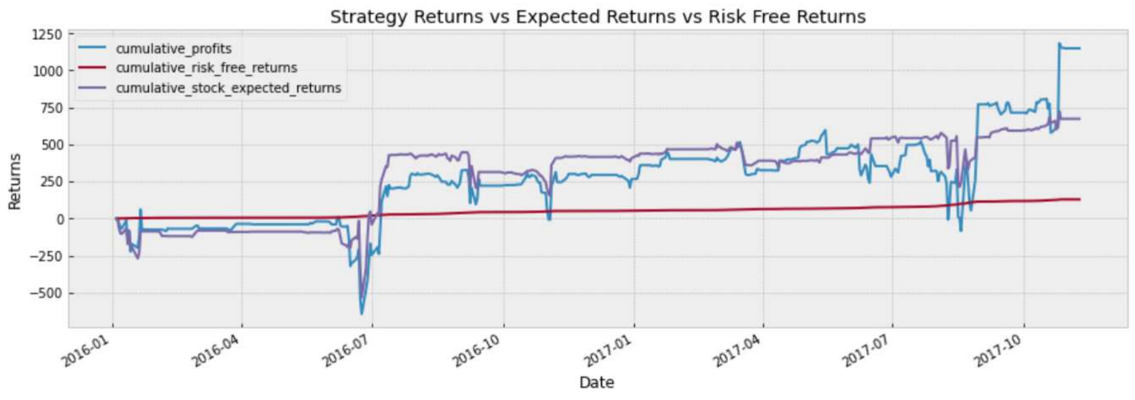
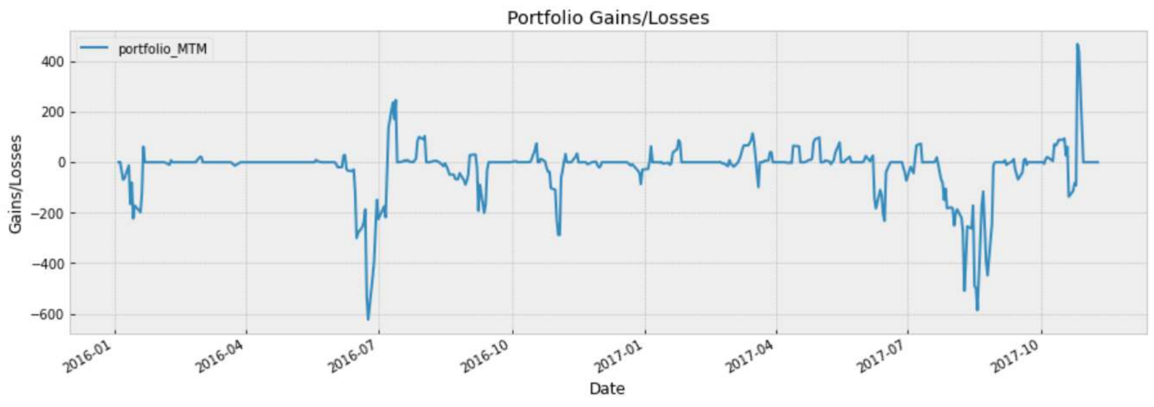


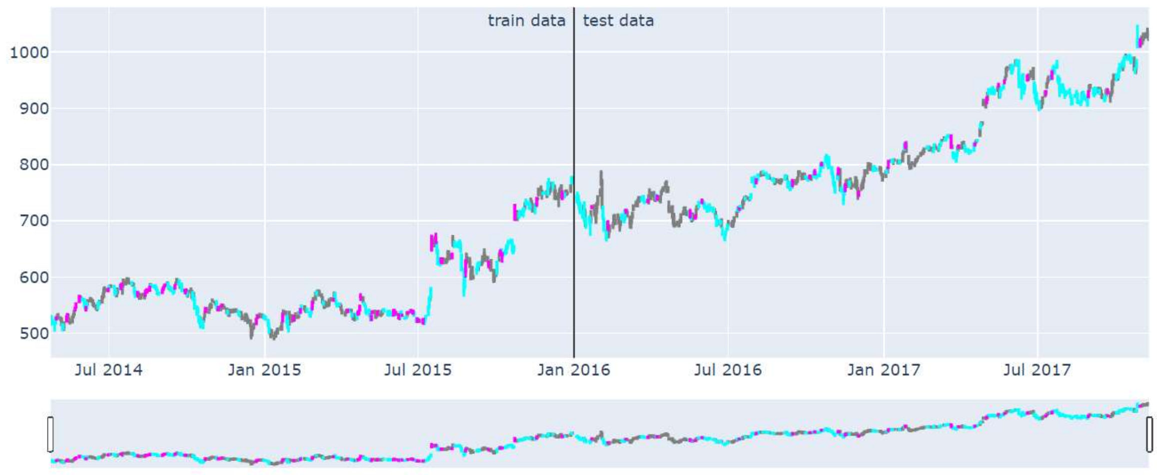
Figure 1: Strategy Return vs Benchmarks (Phase 1)



```
Average Portfolio Size: 3365.2386993603413, Maximum drawdown = -623.62
```

Figure 2: Maximum drawdown (Phase 1)

DQN:train s-reward 1821, profits 1821 test s-reward 1149, profits 1149



```
Gray: HOLD, Cyan: BUY, Magenta: SELL
```

Figure 3:Actions Suggested by Model (Phase 1)

3.2.2. **PHASE 2 – ADDING TECHNICAL ANALYSIS**

The results of phase 2 with the additional of technical analysis indicators are as shown in **Figure 4** and **Figure 5** The cumulative profit of the strategy outperformed both expected stocks return calculated from CAP-M and risk-free return. This translates to 72% profits versus the average portfolio size.

The strategy also only a maximum drawdown of USD $222 with an average portfolio size of USD $1284, signifying a 17% dip in average portfolio size in its lowest performing times. **Figure 6** shows the suggested actions by the phase 2 model for both training and testing data.

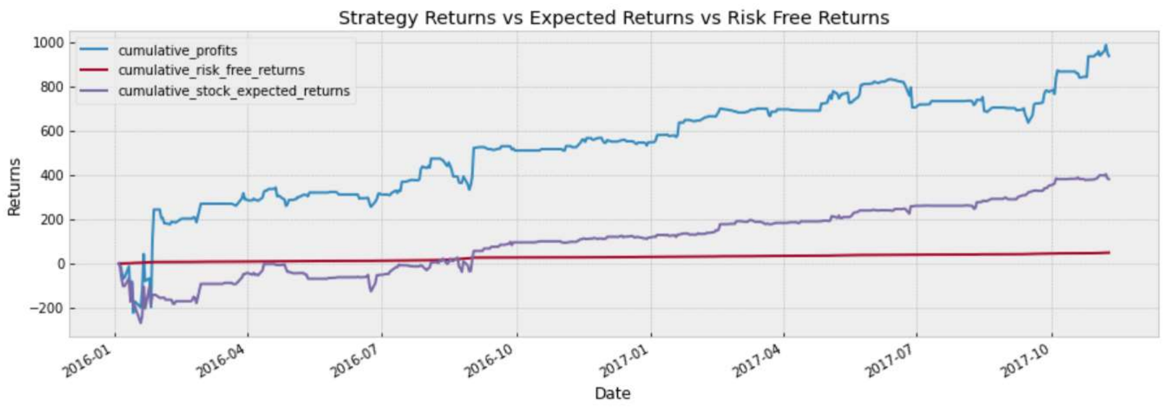
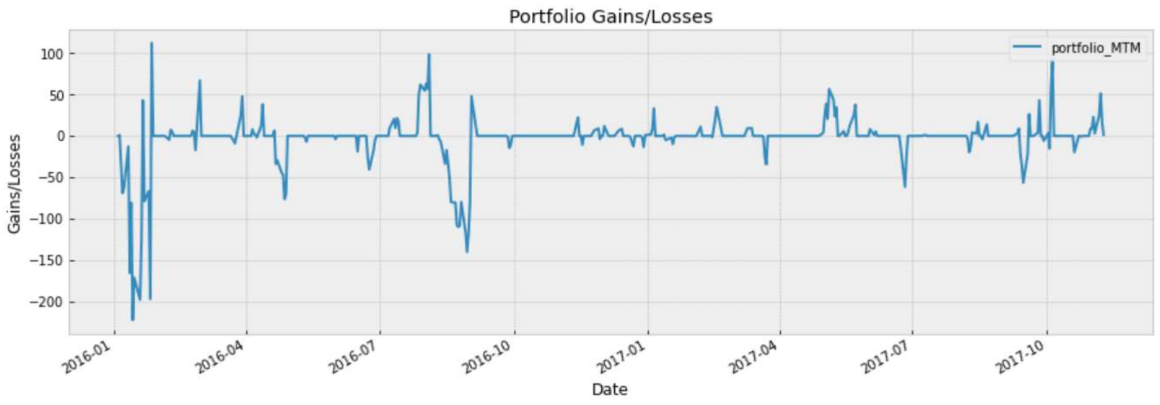


Figure 4: Strategy Return vs Benchmarks (Phase 2)



```
Average Portfolio Size: 1284.869882729211, Maximum drawdown = -222.7299999999998
```

Figure 5: Maximum drawdown (Phase 2)

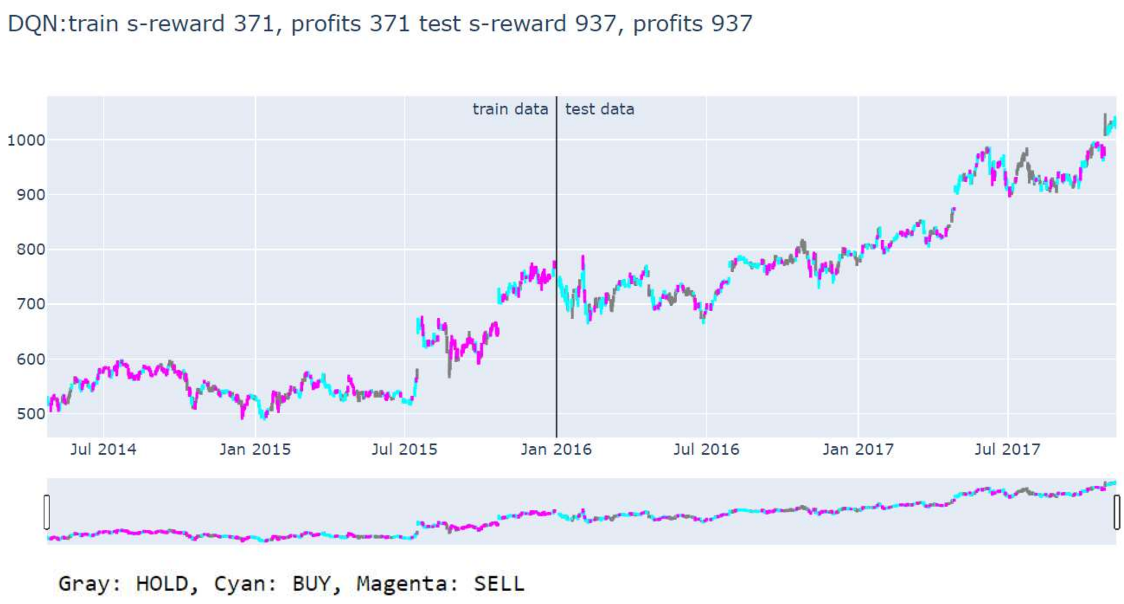


Figure 6: Actions Suggested by Model (Phase 2)

3.2.3. **PHASE 3 – ADDING TRANSACTION AND HOLDING COST**

The results of phase 3 with the addition of transaction and holding cost are as shown in **Figure 7** and **Figure 8**. The cumulative profit of the strategy outperformed both expected stocks return calculated from CAP-M and risk-free interest rates, albeit just slightly for against expected returns. This translates to 89% profits versus the average portfolio size.

The strategy had a maximum drawdown of USD $129 with an average portfolio size of USD $1120, signifying a 11% dip in average portfolio size in its lowest performing times. **Figure 9** shows the suggested actions by the phase 3 model for both training and testing data.

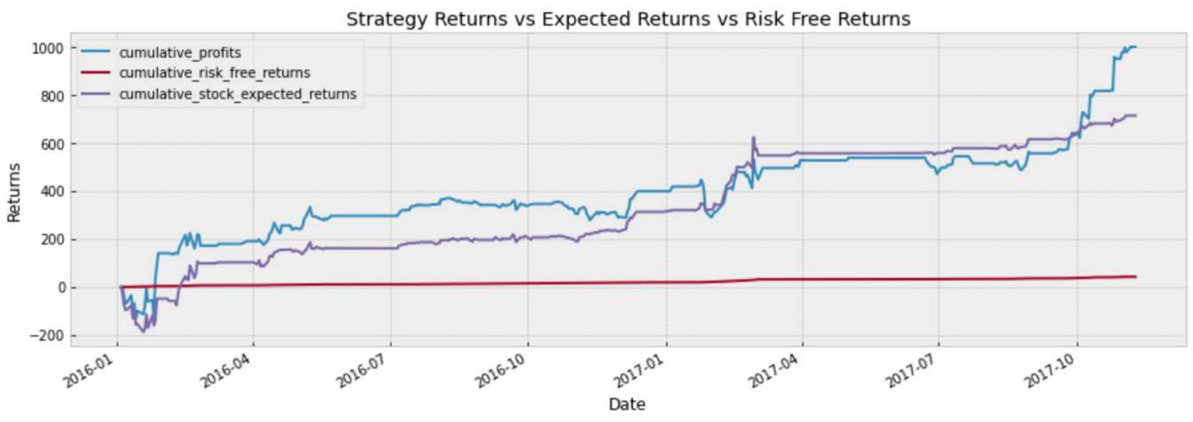
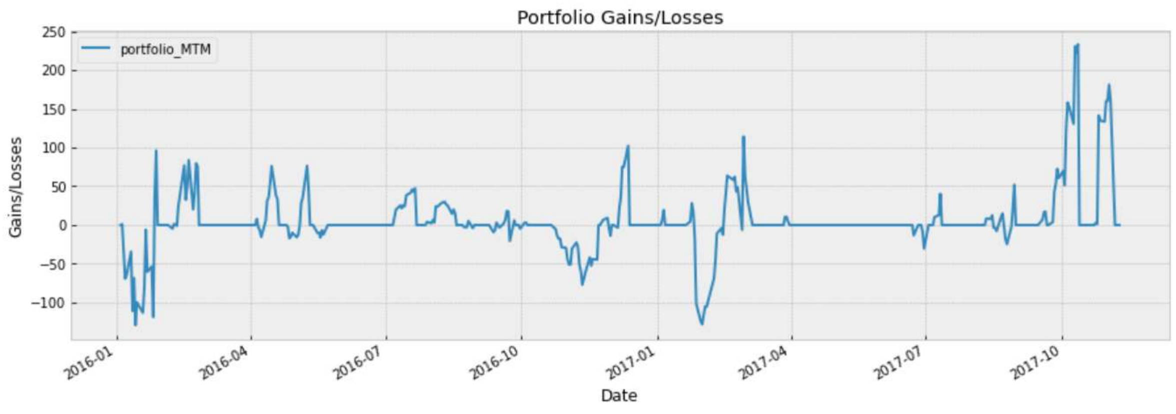


Figure 7: Strategy Return vs Benchmarks (Phase 3)



```
Average Portfolio Size: 1120.821748400853, Maximum drawdown = -129.2399999999999
```

Figure 8: Maximum drawdown (Phase 3)

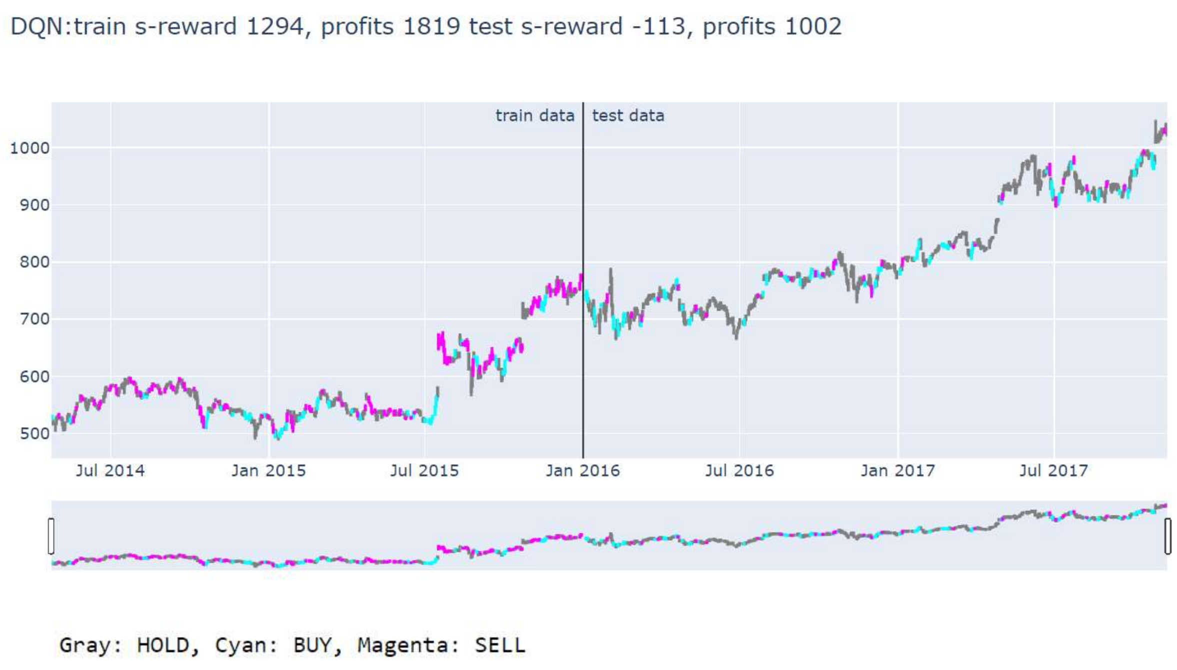


Figure 9: Actions Suggested by Model (Phase 3)

3.2.4.**COMPARISON BETWEEN DIFFERENT ITERATIONS**

Iterative Phases	**Profits vs Average Portfolio Size**	**Maximum Drawdown vs Average Portfolio Size**
Phase 1 : Improved Based Model	34%	18%
Phase 2 : With TA	72%	17%
Phase 3 : With transaction and holding cost	89%	11%

Table 1: Summary of Results

Table 1 shows the summary of results from the iterative phases. Comparing the results of Phase 1 and 2, it was no surprise that Phase 2 model performs better versus benchmark than the results from the Phase 1 model. This is because with the inclusion of technical analysis indicators, more state variables are available for the representation of the stock environment and subsequently used for prediction. Although, some of the technical analysis indicators, such as moving averages, are a summarisation of price, it can be argued that it still helps with learning as a less hidden layers are required for functional approximation and learning.

Comparing the results of Phase 2 and 3, we were slightly surprise that Phase 3 model performed slightly better as compared to Phase 2. The intuition is because with the inclusion of transaction and holding cost, more constraints are added to the environment and thus restricting some of the possible best actions as compared Phase 2. This can be explained when take a closer look at **Figure 4** and **Figure 7**. The Phase 3 model performed only moderately well against the expected market return and only gain the bulk of its profits in the second half of 2017, while the Phase 2 model earnings are more consistent over the whole period. This finding seems to reconcile the intuition of the restricted action space and was only more profitable over some unique scenarios. However, it is important to note that addition of transaction and holding cost makes the model very much more realistic and will translate to better results in the real world.

The maximum drawdown from the average portfolio ranges from 11% to 18% across the three phases and signifies that strategy is relatively conservative. It is also interesting to note that the lowest drawdown of 11% occurs in the Phase 3, which suggest that inclusion of holding cost prompts the sale of stock sooner than later in a bearish market.

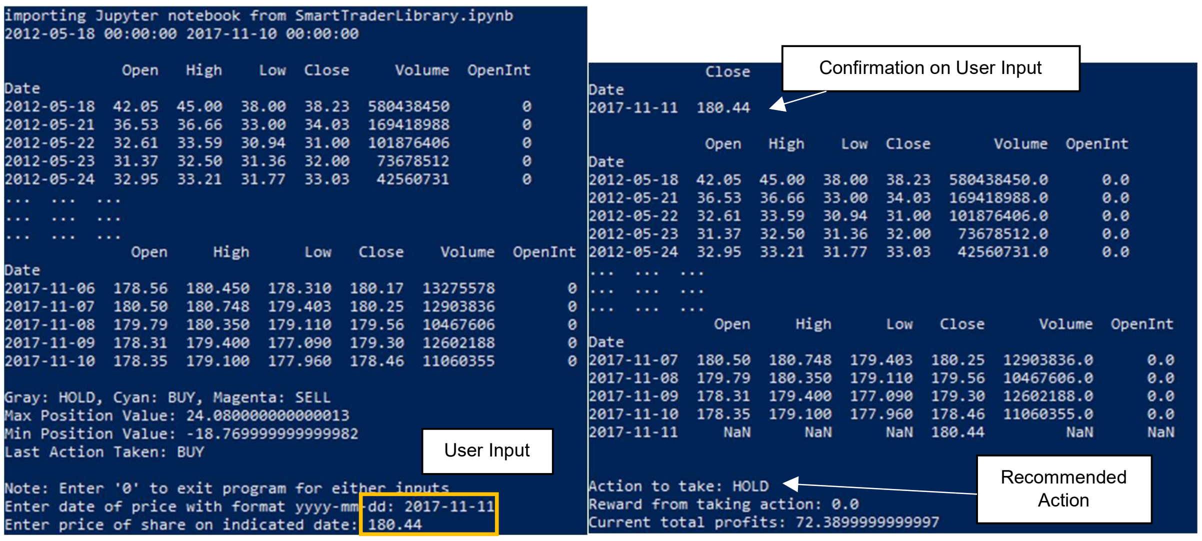
3.2.5. **LEARNINGS FROM USING REINFORCEMENT LEARNING FOR PRICE PREDICTIONS**

However, through the optimisation and training of the model, we found that the model is sometimes not able to converge during training. We believe that this is related to the fundamental issue with prediction for price moments using only price and trading information. Assuming an efficient market, the share price includes and incorporates all information about the company. Therefore, some changes in the external environment, for example, a change in executive management, will affect the share price, but might not be able to be predicted by just price and trading information. This might signify more states are required to better represent the environment, a topic for future work.

4. USER INTERFACE

One way for users to try out the trained model in actual trading scenarios is to use the wrapper python script (Refer to Appendix II for installation guide). The script starts by asking user to input historical data for a selected stock. Once completed, the program will run the model and apply actions to the historical data and simulates the profits made based on the environment.

After which, user is prompted to input current date and stock information. Based on the new information, the action value will be generated for each step and the action with the highest action value is recommended to the user.



```
importing Jupyter notebook from SmartTraderLibrary.ipynb
2012-05-18 00:00:00 2017-11-10 00:00:00

             Open   High    Low  Close     Volume  OpenInt
Date
2012-05-18  42.05  45.00  38.00  38.23  580438450        0
2012-05-21  36.53  36.66  33.00  34.03  169418988        0
2012-05-22  32.61  33.59  30.94  31.00  101876406        0
2012-05-23  31.37  32.50  31.36  32.00   73678512        0
2012-05-24  32.95  33.21  31.77  33.03   42560731        0
...   ...   ...
...   ...   ...
...   ...   ...
              Open     High      Low   Close    Volume  OpenInt
Date
2017-11-06  178.56  180.450  178.310  180.17  13275578        0
2017-11-07  180.50  180.748  179.403  180.25  12903836        0
2017-11-08  179.79  180.350  179.110  179.56  10467606        0
2017-11-09  178.31  179.400  177.090  179.30  12602188        0
2017-11-10  178.35  179.100  177.960  178.46  11060355        0

Gray: HOLD, Cyan: BUY, Magenta: SELL
Max Position Value: 24.080000000000013
Min Position Value: -18.769999999999982
Last Action Taken: BUY

Note: Enter '0' to exit program for either inputs
Enter date of price with format yyyy-mm-dd: 2017-11-11
Enter price of share on indicated date: 180.44
```

```
             Close
Date
2017-11-11  180.44

             Open   High    Low  Close       Volume  OpenInt
Date
2012-05-18  42.05  45.00  38.00  38.23  580438450.0      0.0
2012-05-21  36.53  36.66  33.00  34.03  169418988.0      0.0
2012-05-22  32.61  33.59  30.94  31.00  101876406.0      0.0
2012-05-23  31.37  32.50  31.36  32.00   73678512.0      0.0
2012-05-24  32.95  33.21  31.77  33.03   42560731.0      0.0
...   ...   ...
...   ...   ...
...   ...   ...
              Open     High      Low   Close      Volume  OpenInt
Date
2017-11-07  180.50  180.748  179.403  180.25  12903836.0      0.0
2017-11-08  179.79  180.350  179.110  179.56  10467606.0      0.0
2017-11-09  178.31  179.400  177.090  179.30  12602188.0      0.0
2017-11-10  178.35  179.100  177.960  178.46  11060355.0      0.0
2017-11-11     NaN      NaN      NaN  180.44         NaN      NaN


Action to take: HOLD
Reward from taking action: 0.0
Current total profits: 72.3899999999997
```

Figure 10: User Interface for SmartTrader Wrapper

The program will also advance the model by applying the action to calculate the reward and profit of the next state.

The current state of the program only runs based on a model which only requires user input of the closing price. Once there are other models generated using other technical analysis, the program is easily scalable to include user inputs for that information.

5. CONCLUSION

The project set out to build a reinforcement learning model based on Price Action Trading. We also made improvements iteratively through the exploration of various states and rewards that can provide a better representation of the environment and model real-life cost respectively. We found that improving the state model with the addition of technical analysis, improves the performance of the agent, while the adding transaction and funding cost makes the model more realistic. As it is difficult to capture all the attributes of a real-life trading environment, therefore further improvements in the model should bring about better performance of the model.

5.1. FUTURE IMPROVEMENTS

Although successful in our objectives, there are some future improvements that might help to improve the model.

- **Better Network Architecture using GA**
  As our project focuses on the exploration on the representation of the environment through the changes made in state and rewards, more focus can be placed on the deep learning to optimise learning. A good suggestion will be to use genetic algorithm to help to decide the architecture of the neural network.

- **Improving the State Model**
  As we have established, the current states variables that are chosen only gives a partial view of the whole environment, we can look into ways to include more state variables to provide a more complete understanding of the environment. This can include the using a larger historical window for the Markov state and/or adding the current position as part of the state to allow the model to factor this into its learning.

- **Improving the Actions Available**
  As our current model only allows for only long positions, an improvement will be for the model to allow for short-selling to take advantage of the actions that is provided by the model, especially since that is allowed in the US stock market. Other improvements include expanding the range of actions from BUY, SELL and HOLD to include the quantity of stocks to be purchased.

APPENDIX I: NEURAL NETWORK STRUCTURE

Layer Type	Output Shape	# of Weights
Dense	(512)	(Input + 1) x 512
Dense	(256)	131,328
Dense	(128)	32,896
Dense	(3)	387

Table 2: Neural Network Structure

All activation functions used are ReLU functions.

APPENDIX II: SMART TRADER INSTALLATION & USER GUIDE

1. Install Python 3.8
2. Click on *install.bat* to install required python packages
3. Click on *run.bat* to start SmartTrader
4. A prompt will appear to choose the data file consisting of the stock price history
5. There are sample datasets available in the “Data” folder to choose from