Problem Chosen	2022 MCM/ICM Summary Sheet	Team Control Number
C		2226134

We created three models to analyze the most profitable strategy for trading gold and bitcoin. The models had varying levels of success, ranging from barely breaking even to over $\$80,000$ when the model cashed out given an input of $\$1,000$.

Much of this success can be attributed to the spike in bitcoin in late 2020-early 2021. What we found was the more successful models had bought into bitcoin early and held onto it until after the cryptocurrency spiked in price, owning statistically nothing to gold investments. In a humorous hypothetical case, we created an algorithm that invested all $\$1,000$ in bitcoin day 1 and cashed out once it hit $\$50,000$, making $\$\mathbf{81,170.16}$ on top of the \$1k after all the fees are taken into account.

Our first predictive model took advantage of reinforcement learning, particularly a method called Q-Learning. It was a value-driven algorithm that utilized Markov Decision Processes to determine maximal value given primarily the present state while also utilizing historic data. Unfortunately this model did not take into account the transaction fees so it ended barely breaking even earning $\$\mathbf{13}$. Some adjustments are needed to make it more profitable, since it is a suitable candidate for the model at hand.

The more successful model was built from scratch and was called the Delta Method. It took the differences between each day and the day before, as well as the variance in those values and came up with a predictive difference between today's and tomorrow's price. Parameter tuning was included, which allowed the model to highlight various behaviors as important vs not. We were able to tune it to a "general" delta model, as well as a "conservative" model that was capable of taking advantage of the aforementioned behavior of holding onto bitcoin and selling after the 2020 spike.

The success between the two variations of this model is vastly different for the historic data. The more conservative model was closer to the all-in model that bought bitcoin early and then refused to sell until after the spike. This behavior was rewarded by the historic prices and gave us a profit of $\$\mathbf{67,581.04}$. The more general model is more likely to be profitable in the future but did not take advantage of the spike in bitcoin as well, and so yielded us with only $\$\mathbf{4,000}$. Though the conservative model outperformed the general model by a significant margin, we do not expect to see that sort of success being replicated in the future, discounting possibility of bitcoin having another massive spike.

Based on the scenario, we have created different strategies to maximize profit through gold and bitcoin investment. Given the historic data, the most successful models were able to take advantage of the bitcoin spike and earn several tens of thousands of dollars while a more general delta model didn't earn as much but was better tuned for the event there is no volatile spike.

Contents

1 Introduction

Following the explosion of bitcoin and other crypto-currencies as trading commodities, investors became interested in developing an algorithm for predicting the price of crypto, of which the most popular is bitcoin. This paper will take the price of gold and bitcoin from September 11, 2016 - September 11, 2021 to make three different models that will try to maximize profit. The models will be compared to find the best algorithm for predicting future purchases of gold and bitcoin.

2 Problem Statement

We are asked to create a model to determine when to buy/sell bitcoin and gold using historic price trends from 2016-2021. The model can only use past price trends in making its decision to buy or sell.

3 Background

Trade, the exchange of goods and services between countries [3], can immensely ameliorate the economic welfare of a given country, region, and of course, the individual. Research has shown that trade has contributed to mitigating the poverty of hundreds of millions of individuals [4]. As such, due to trade's tremendous financial potential, it becomes easy to see why entities like trading markets exist. Trading markets often consist of very intertwined and well-connected systems of rather volatile assets like gold, silver, or even bitcoin that can greatly range in value depending on the given day. The inherent volatility of traded items like gold or bitcoin can make secure, long term market profits a challenge for investors.

Consequently, being able to effectively predict when and what asset(s) to buy or sell can not only produce immense profit for said individual (or conversely, save large loss), but on a larger scale, can significantly enhance the economic conditions of a country (even after adjusting for inflation). Notably, this in turn can influence critical societal pillars like infrastructure, health care, energy, and or education. So, ideally, to obtain optimal profit, these predictions would be performed "real-time" to give investors live feedback on the most cost-effective opportunities.

Naturally, since current and future data often depends on prior data (even despite the inherent, very random fluctuation of markets), an effective real time model would make usage of the market data that is already available. Moreover, designing a model that can reliably (within reasonable bounds of error) predict when and what investments are most optimal at a given date should only depend on data *prior* to that date. As a result, with five year's worth of market data available for the given problem, the focus of our project is building an effectual model that will only use *past* data to obtain the largest net profit margins.

4 Diamond Hands

The first and simplest algorithm we came up with was inspired by the Reddit meme of Diamond Hands (DH) that was popularized with the GME buyup by Reddit [1]. The basic strategy of

this algorithm is to keep holding until a certain high threshold is reached. Assuming that the future is unknown, $50,000 would make for a reasonable, round threshold. This threshold is first reached on Febuary 18th, 2021 when bitcoin had a posted price of $52118.23. Thus, the 1.6 BC that would have been bought day 1 would have sold for $82170.16 post-fees, making over 8000% profit.

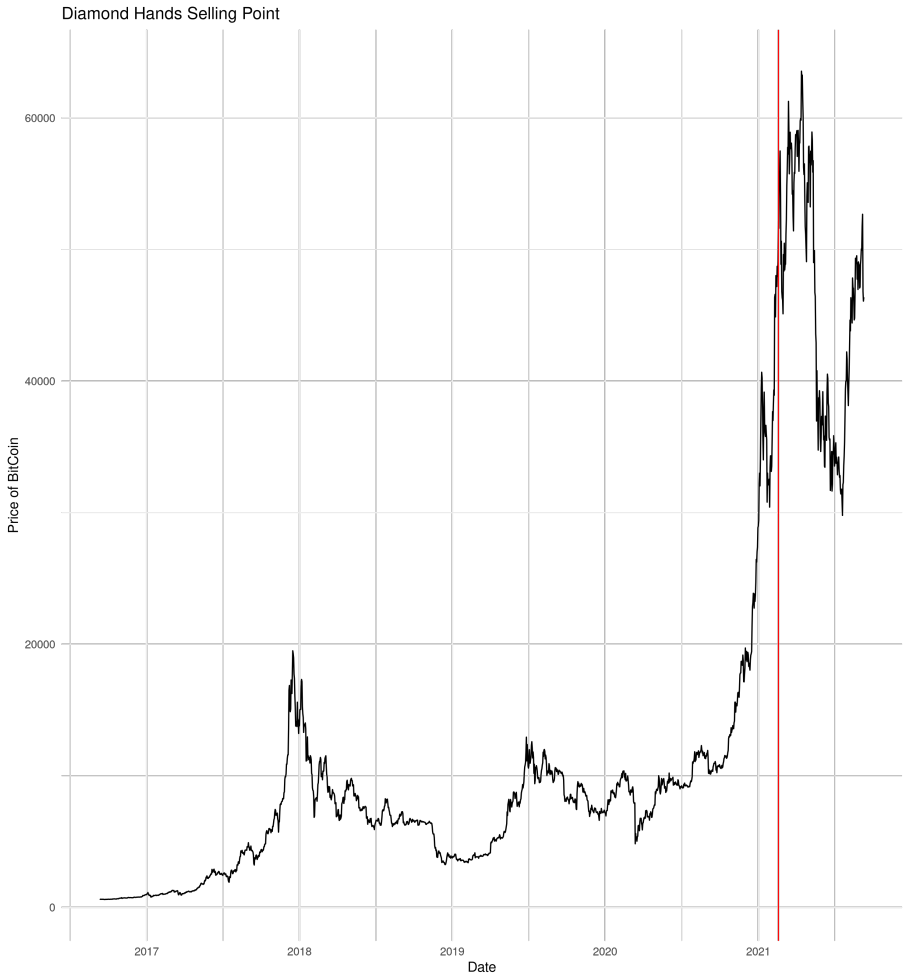


Figure 1: Plot Showing When Algorithm Sells

A similar methodology can be applied to gold, though to lesser awards since it is a less volatile resource. Going all-in on the gold market and cashing out on the 6th of August, 2020, the day when gold prices are the highest would yield only $1408.43. As a result, going all-in on bitcoin going up yields far greater rewards.

The DH algorithm has a few noticeable upsides, as well as some clear downsides. It is good in its simplicity, there are not any moving pieces that makes it easy for people to understand. It also does not care much for transaction costs, since there will only be two transactions, one to turn cash into bitcoin and a second one to reverse. The downside of this strategy is it only really works if a currency/resource becomes very valuable, such as was the case with bitcoin. Beyond hoping that a currency reaches an arbitrarily high threshold, the model does nothing in terms of analyzing past trends and will fail if the threshold is not reached.

Thus, the DH model will not be useful in the future unless its price fluctuates again. It will still be useful as a metric to measure the other models in terms of to see the overall potential of waiting and not being excessively taxed by the resulting transaction costs.

5 Q Learning Method

For our first predictive model, we used a powerful learning technique called Q-learning. Q-learning fits under a much broader category of methods in a field called reinforcement learning. Reinforcement learning is an area of machine learning where there are two main entities [5]: environments and agents. Agents take actions to maximize their rewards [6] .

Q-learning specifically seeks to educate the agent on what actions will maximize the reward in a given situation [7]. It is very much value-driven. The q in the name denotes quality, that is, how cost effective the reward will be in light of the action performed [5]. Q-tables, which form the basis for Q-learning, are created with dimensions corresponding to the state of the system and the resulting action [5]. The given agent can interact with the environment by either exploiting or exploring [5].

Exploiting entails considering all actions to determine the one which yields the maximum value for the environment. Exploring refers to when a random action is taken without accounting for the maximum future reward. Due to its ability to consider optimal choices at a given step in time, Q-learning is a tool that can be used to help develop an automated trading strategy [5]. Because Q learning assumes the ensuing state is directly contingent on the prior state (also referred to as the Markov Decision Process) [5], it lends itself towards market trading that focuses on only *past* data.

Since trading here can either involve buying, selling, and or holding, Q-learning would rank action with each other to choose the one that gives the optimal yield. The agent would be given an initial amount of money and would ideally, over time, learn enough about the market trends to produce a very high net profit margin.

In utilizing Q-learning, since the model did not provide to us an opportunity to account for commissions in its decision tuning, we applied the commissioning rate to the data by scaling it before feeding it into the model. This ensured decisions at each step would already take into account the cost associated with them. Additionally, since gold could only be traded on the days the market was open, we separated gold and bitcoin data into two different fields that had resulting dates associated with them. While we couldn't account for both gold and bitcoin amounts in the same market "deal", one approach we formulated was splitting the investment in half towards bitcoin and the other half in gold. That is, we ran two of the same type of models on $500 The reasoning being of course, even with the half-half split, the total profit margin we get by aggregating their yield may very well be an actual margin you could obtain (perhaps even the optimal one in the long run). In contrast, running a model (or two models of the same type) into two separate directions with full investment on both ends would not allow us to combine the total yield at the end since we would be over counting our initial investment twice over. Moreover, the equal cut allows us to weigh both gold and bitcoin evenly and eliminate any unfair bias from the onset. Of course, while one was likely to be more profitable than the other, since their gains would vary from iteration to iteration, their respective advantages would average or "even each" other out (at least to a point where there is not net significant benefit

in choosing one over the other) and so, an even split was the most sensible approach for the purposes of this technique.

After running our two Q-learning models and aggregating the total net profit obtained from both gold and bitcoin, the models yielded a total net profit of $13. Interestingly, gold resulted in a $11 loss and bitcoin in a $24 gain all respective to their $500 yield.

```
day 1228: buy 1 unit at price 18.291000, total balance 436.989000
day 1229, sell 1 unit at price 18.007500, investment -1.290906 %, total balance 454.996500,
day 1231, sell 1 unit at price 17.388500, investment -4.071387 %, total balance 472.385000,
day 1232, sell 1 unit at price 17.233500, investment -5.781532 %, total balance 489.618500,
day 1236: buy 1 unit at price 17.863500, total balance 471.755000
day 1238: buy 1 unit at price 17.834500, total balance 453.920500
day 1239: buy 1 unit at price 17.820500, total balance 436.100000
day 1240, sell 1 unit at price 17.790500, investment -0.408655 %, total balance 453.890500,
day 1241, sell 1 unit at price 18.020000, investment 1.040119 %, total balance 471.910500,
day 1242: buy 1 unit at price 18.084500, total balance 453.826000
day 1243: buy 1 unit at price 17.887000, total balance 435.939000
day 1244, sell 1 unit at price 17.866000, investment 0.255324 %, total balance 453.805000,
day 1245, sell 1 unit at price 17.985000, investment -0.550195 %, total balance 471.790000,
day 1246: buy 1 unit at price 18.148500, total balance 453.641500
day 1252, sell 1 unit at price 17.860000, investment -0.150948 %, total balance 471.501500,
day 1253, sell 1 unit at price 17.882500, investment -1.465686 %, total balance 489.384000,

  epoch: 10, total rewards: 216.892925.3, cost: 1.496600, total money: 716.892925
  epoch: 20, total rewards: 142.180558.3, cost: 0.408958, total money: 642.180558
  epoch: 30, total rewards: 191.391656.3, cost: 0.253130, total money: 691.391656
  epoch: 40, total rewards: 72.075653.3, cost: 0.179983, total money: 572.075653
  epoch: 50, total rewards: 128.795183.3, cost: 0.128322, total money: 628.795183
  epoch: 60, total rewards: 49.846849.3, cost: 0.094761, total money: 549.846849
  epoch: 70, total rewards: 23.339644.3, cost: 0.695151, total money: 523.339644
  epoch: 80, total rewards: 6.946708.3, cost: 0.580360, total money: 506.946708
  epoch: 90, total rewards: 80.160171.3, cost: 0.058143, total money: 580.160171
  epoch: 100, total rewards: 37.340328.3, cost: 0.057804, total money: 537.340328
  epoch: 110, total rewards: 77.257566.3, cost: 0.037170, total money: 577.257566
  epoch: 120, total rewards: 83.633993.3, cost: 0.034283, total money: 583.633993
  epoch: 130, total rewards: 82.305400.3, cost: 0.032240, total money: 582.305400
  epoch: 140, total rewards: 99.352710.3, cost: 0.037235, total money: 599.352710
  epoch: 150, total rewards: 96.820723.3, cost: 0.034636, total money: 596.820723
  epoch: 160, total rewards: 24.250414.3, cost: 0.025483, total money: 524.250414
```

While our Q-learning model did not perform nearly as well as we hoped, the method itself seemed to have been a very good fit for this particular type of problem, notwithstanding some adjustments like differing dates for golds' availability and the commissioning scale we applied before processing data. Likely, those very adjustments contributed to the model's weaknesses. It is very likely that just having one model that could easily and seamlessly integrate both investments at each step would have resulted in a much (if not significantly) higher net profit yield. Consequently, the natural limitations of this model lead us right into our next modeling algorithm: the delta method.

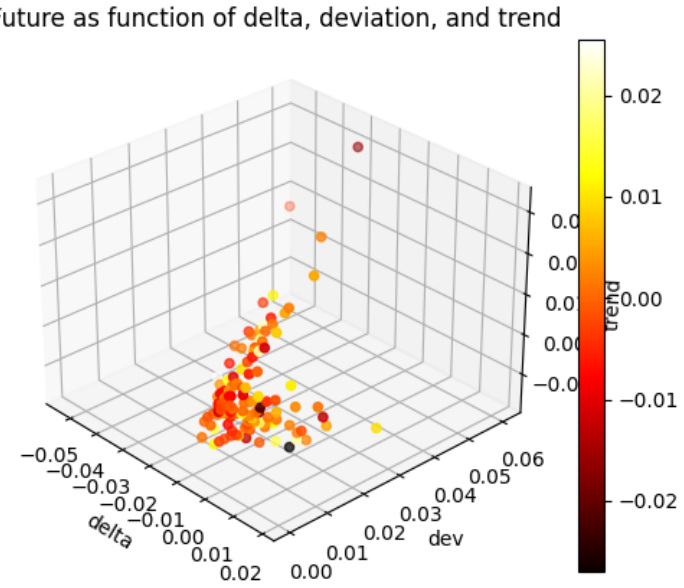
6 Delta Method

The algorithm of we ultimately chose is one we devised called the "Delta Method". It seeks to extrapolate three separate metrics, parametized by i, from the data, and use the findings to

faciliate decision making in a model. The metrics are as follows:

- Δ: the difference between the price today and the price yesterday, scaled
- σ_i: the variance in Δ values, from the current Δ, counting i days backwards, scaled
- τ_i: the 'trend': the sum of all Δ values going i days backwards, scaled
- (scaling was simply done by dividing by current price)

The final metric - our 'dependent variable', simply denoted 'future', is the Δ between today and tomorrow. The metrics individually exhibit very low correlation with 'future'.



With these low correlations, it was not possible to directly use our findings (i.e., with PMF function) to give our model decision making. Instead, we implemented the following logic based on the above correlations:

For gold:

- A positive previous delta is GOOD
- A positive trend in short term ($i \leq 2$) is GOOD
- A positive trend in long term ($i > 2$) is BAD
- Variance in long term ($i = 5$) is GOOD

For crypto:

- A positive previous delta is BAD

Table 1: Linear Correlations with Future

i	Delta	Var	Trend	
1	0.024	-0.019	0.05	<- Gold
2	0.024	0.003	0.028	
3	0.023	0.008	-0.033	
4	0.024	0.0166	-0.04	
5	0.024	0.016	-0.039	
1	-0.047	0.025	0.037	<- Crypto
2	-0.047	0.05	0.002	
3	-0.047	0.037	0.02	
4	-0.047	0.033	0.026	
5	-0.047	0.041	0.036	

- A positive trend is always GOOD, but more so for greater i
- Positive var is always good, but better in long term

These observations lead to the following scoring system implemented in Python:

```python
#d, v, t are the weights for delta, var, and trend, respectively
#delta, var, and trend are lists
#with list[0] corresponding to i = 1, list[1] corresponding
#to i = 2, etc.
def get_score_gold(delta, var, trend, d, v, t):
    #Positive previous delta is good
    delta_score = 0
    for row in delta:
        delta_score += d * (row / GOLD_avg_delta)

    #Variance seems to only matter in the long term
    var_score = v * (var[4] / GOLD_var_avgs[4])

    #Positive trend is good in short term, bad in long term
    trend_score = 0
    i = 0
    for row in trend:
        if (i <= 2):
            #If short term, negatively impact score
            trend_score += t * (row / GOLD_trend_avgs[i])
        else:
            #If long term, positive impact score
            trend_score = trend_score - t * (row / GOLD_trend_avgs[i])
        i += 1

    return delta_score + var_score + trend_score
```

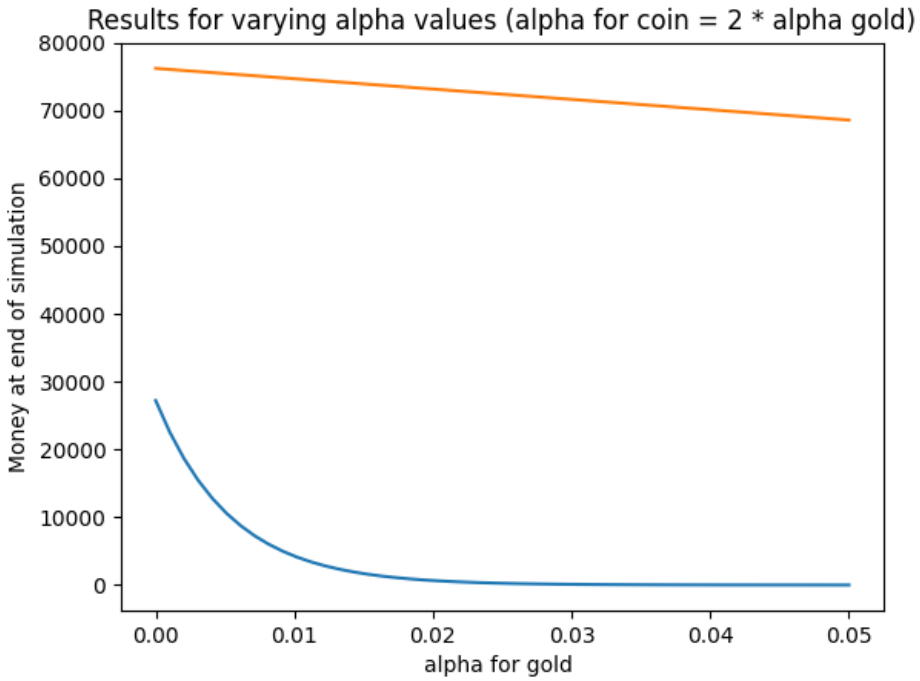
```
def get_score_coin(delta, var, trend, D, V, T):
    #Positive previous delta is bad
    delta_score = 0
    for row in delta:
        delta_score = delta_score - D * (row / COIN_avg_delta)

    #Var is good, but gets better in the long term
    var_score = 0
    modifier = 1
    i = 0
    for row in var:
        #Modifier increases as i increases
        var_score += modifier * V * (row / COIN_var_avgs[i])
        i += 1
        modifier += 1
    var_score = var_score / 15 #15 is sum of modifiers

    #Trend is good, but better in the long term
    trend_score = 0
    modifier = 1
    i = 0
    for row in trend:
        #Modifier increases as i increases
        trend_score += modifier * T * (row / COIN_trend_avgs[i])
        i += 1
        modifier += 1
    trend_score = var_score / 15

    return delta_score + var_score + trend_score
```

The model obtains a score for both commodities, and, if the score is above G or C, for gold and crypto, respectively, it purchases an amount equal in value to half of its cash reverse. If the score is below, it sells equal to half of its reserve of that commodity.

All together, the model has 8 parameters: d, v, t, D, V, T, C, G, with d, v, t being the weights for gold, D, V, T being the weights for crypto, and C, G being the score threshold for crypto and gold, respectively.

In an initial test runs, with parameters $(1, 1, 1, 1, 1, 1, 10, 10)$, **without considering transaction cost**, the model yielded a profit of \$23751.57

Testing over 6 thousand parameters (which amounted to only 3 values per parameter), the maximum profit, **without considering transaction cost**, was $\$75,214.64$. The parameters were $(0, 1, -1, 1, -1, 0, 0, 10)$ - with the values for t and V contradicting our scoring assumptions. The 0 values are somewhat less surprising, as they are less of a direct refutation of our assumptions than perhaps an indication that the ideal value is somewhere closer nearer to 0 than the other (3) options. However, upon examination of the model's record, it becomes apparent that the high profit is due to quickly expending the majority of its cash reverse on bitcoin, as a result of the coin threshold being equal to 0, and the buy/sell decision for coins being placed before the decision for gold. Applying the transaction cost resulted in profits of \$67581.04 and approximately \$4000 for the optimized and general case, respectively.



6.1 Results for different values of alpha

Varying the value of alpha has a nearly perfectly linear proportionality with the return on investment for the 'optimized' parameters. This is to be expected, as all trades after obtaining approximately 1.6 bitcoin are negligible - thus, the value of α mainly only the first 10 or so transactions.

The effect on the default parameter model is exponential decay, suggesting that the threshold for purchasing is incorrect. It in fact suggests that there needs to be a separate threshold for selling, rather than automatically selling if buying is deemed the incorrect choice (ex: buying is not a good choice if the price is going to go down by .5%, but neither is selling).

6.2 Recommendations for Future Optimization

The general version of the model, with default parameters, is capable of making a profit over 20 times initial investment without transaction fees. The "optimized" version makes an over 74 fold profit - but as previously discussed, this is due to the parameters essentially blocking the model from buying any gold and selling any bitcoin. To optimize the model overall, rather than for this particular time period, we pose the following recommendation:

- Divide the dataset into multiple parts
- Compute the maximum potential earnings, or approximate them
- Run model with variety of parameters on each data set, and score out of maximum potential earnings

The goal of this approach is to prevent the process from simply 'identifying' a singular event (like the sharp rise in bitcoin price) and declaring that the optimal model.

7 Memorandum

To the trader,

Our proposed model is an innovative, novel way of predicting price trends. While its full potential has not yet been utilized, the model has demonstrated an impressive ability to return yields ranging from 4 to as much as a 65 fold return on investment. Notably, this is only with testing a single, non-split sample, with our optimizations considering only three possible values for each of our eight parameters. It is very likely this severely limits the model's potential, so with further testing, particularly of the decision-making-threshold variables, we believe we can achieve even significantly larger profit margins.

The model operates by extracting three metrics from the daily price data, each parametrized by i, reflecting how many days backwards they measure. The current implementation of the model uses $i = 5$, providing fifteen individual data points on each given day to be used in the decision making process.

Data points are fed to a scoring system parametrized with weight values for each of the data points. The scoring system evaluates the input based on observed trends in the 2016-2021 periods. Measurements indicating poor future performance hurt the score; measures indicative of future gains increase it.

The scores are then used in a simple decision making process. If a commodity is expected to increase in value on the next day, indicated by the score being over a specified threshold, you will buy - and otherwise, sell. Further work will likely suggest that there needs to be a separate selling threshold, with some type of a middle ground for the hold behavior.

We have some recommendations for where to proceed with this model, listed in the Recommendations section of the full report. Most importantly, we believe splitting the test data into sets and optimizing performance across all sets. Please see this section for more details.

Please also note that in its current state, the model assumes it can sell gold on non-trade days for the previous closing price. This should not severely impact performance, but before implementation this feature should be removed.

8 Conclusion

While the delta method greatly outperformed Q-learning, one rather apparent trend we've noticed in our models is the overwhelmingly disruptive nature and profit margins bitcoin can yield for the market. For their final results, both the Q-learning model (even despite its even split) and the variants of the delta method seemed to favor and utilize the advantages bitcoin's volatility brought to the market. Of course, because bitcoin is a crypto-currency, it shouldn't be surprising that we notice such extremely volatile changes in prices. As a result, even besides its contrast to other commodities like gold, bitcoin (and possible other crypto-currencies) seem to be the singularly defining characteristic of trading markets as a whole.

As such, regardless of what model is utilized to predict market trends and maximize net profit margins, succeeding in the market will largely compromise of being able to understand

and predict *fundamentally* unpredictable fluctuation in bitcoin prices. This can largely explain the immense difficulty even sophisticated models like neural networks or techniques like Heston Model that assume an inherent randomness in market trends [2] face when it to accurately predicting when and how to buy and sell items like bitcoin. Of course, as we saw from our higher performing models, this doesn't mean there isn't a capacity to obtain very large profit margins, but it does always leave a great deal of uncertainty in the long term sustainability of one's margins.

As Yogi Berra succinctly stated, "'It's tough to make predictions, especially about the future". Bitcoin prices seem to be no exception to this rule. Consequently, predicting market potential is really like playing a game of whack a mole: it's whoever strikes the most "in time" opportunities (and sells out appropriately) that "wins" in the end. Unlike the tale of the tortoise and the hare, slow and steady likely don't always win the race. And accounting, perhaps even fully embracing this fickle reality, is going to be the key that makes these predictions hold just a little more weight for all investors alike.

References

[1] Dictionary. Dictionary diamond hands, 2021. https://www.dictionary.com/e/slang/diamond-hands/.

[2] Akhilesh Ganti. Heston model, 2021. https://www.investopedia.com/terms/h/heston-model.asp.

[3] Reem Heakal. Investor's guide to global trade, 2021. https://www.investopedia.com/insights/what-is-international-trade/.

[4] OECD. Why open markets matter, 2021. https://www.oecd.org/trade/understanding-the-global-trading-system/why-open-markets-matter/.

[5] Ekta Shah. Bear run or bull run, can reinforcement learning help in automated trading?, 2021. https://www.analyticsvidhya.com/blog/2021/01/bear-run-or-bull-run-can-reinforcement-learning-help-in-automated-trading/#:~:text=Q%2DLearning%20is%20such%20a,RL%20agents%20with%20different%20stocks.

[6] Richard Sutton and Andrew Barto. Reinforcement learning. 2015. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[7] Andre Violante. Simple reinforcement learning: Q-learning, 2019. https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56.