Multi-Armed Bandit Algorithms in Short-term Investing

William Feng
Project Mentor: Anthony Della Pella

April 25, 2025

1 Abstract

The Multi-armed Bandit (MAB) Problem is a machine learning and game theory problem with many applications, primarily dealing with exploitation versus exploration when presented with a series of multiple bandits, each with their own unknown reward distribution. In this paper, two of the most theoretically reliable bandit algorithms, the ϵ-Greedy and UCB1 algorithms were tested on sample finance data from the YFinance API to determine if they provided any realistic benefit to investing. The sample data was run through four trials: a negative control trial using a dummy dataset to show that the behavior of the two algorithms was predictable, a bernoulli trial which simplified the decision making of the bandit by assigning 0 or 1 to stock price changes, a real value trial, and a modified real value trial with large technology companies inserted into the sample.

The goal of the negative control and real value trials were to determine the effect of an increase or decrease in stock prices on cumulative regret. In the last three trials, the bandit algorithms used had no impact on the payout of the finance data, and it was concluded that the Multi-armed Bandit is ineffective. This study gives an introduction to the implementation of simple machine learning algorithms in finance by treating the stock market as a game. Although bandit algorithms are currently too "weak" to realistically act on investments in a realistic time frame, modified bandit algorithms show promise in providing stronger returns.

2 Introduction

The Multi-armed Bandit (MAB) Problem is a problem widely explored in probability and game theory, where a player is faced with multiple "bandits," or arms, each with their own payoff distributions once the arm is pulled. The player does not know the reward distribution of the bandits and seeks to find the one with the highest payoff. In the MAB problem, the goal of the player is to minimize regret, which is the difference between the reward of the optimal strategy and the reward of the current strategy. The regret at time T in a multi-armed bandit problem is given by:

$$R(T) = T\mu^* - \sum_{i=1}^{K} T_i(T)\mu_i$$

where T is the total number of time steps, μ^* is the expected reward of the optimal arm, K is the number of arms, $T_i(T)$ is the number of times arm i has been pulled up to time T, and μ_i is the expected reward of arm i.

The MAB problem has applications in financial investments, where each company's stock can be represented as a bandit. The player, when presented with a certain number of companies to choose from, can apply bandit algorithms using the known past stock prices to theoretically better predict the returns of stocks. Many algorithms have been devised to find a lower bound for risk, and this study will focus on the ϵ-greedy and UCB-1 Algorithms in the context of finance. These algorithms are largely applied to static data, so a big topic for exploration is whether they will still hold up when the dynamic nature of the stock market is applied to the algorithms.

- The ϵ-Greedy algorithm Kuleshov & Precup [2014] selects actions according to the following probabilistic rule: at each time step t, with probability $\epsilon \in [0, 1]$, the agent chooses an arm uniformly at random (exploration), and with probability $1 - \epsilon$, it selects the arm with the highest estimated mean reward (exploitation). Thus, it can be modeled by the equation

$$p_i(t+1) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{k}, & \text{if } i = \arg\max_{j=1,\ldots,K} \hat{\mu}_j(t), \\ \frac{\epsilon}{k}, & \text{otherwise.} \end{cases}$$

  where k is the number of bandits available to be pulled.

- The Upper Confidence Bound (UCB1) algorithm Auer *et al.* [2002] incorporates a principle of optimism under uncertainty. It selects the arm i that maximizes an upper bound on the expected reward, combining exploitation of high empirical means with exploration of high-uncertainty arms. At time t, the algorithm chooses arm $j(t)$ according to:

$$j(t) = \arg \max_{i \in \{1,\ldots,k\}} \left[\hat{\mu}_i(t) + \sqrt{\frac{2 \ln t}{n_i(t)}} \right],$$

where:

- $\hat{\mu}_i(t)$ is the empirical mean reward of arm i up to time t,
- $n_i(t)$ is the number of times arm i has been selected by time t,
- $\sqrt{\frac{2 \ln t}{n_i(t)}}$ is the confidence term encouraging exploration.

The ϵ-greedy algorithm has been explored in-depth in a paper by Dann, Mansour, Mohri, Sekhari, and Sridharan Dann *et al.* [2022]. It has been observed to converge to a bounded regret value very quickly, and certain conditions are able to speed up or slow down the activity of the algorithm. Although other methods will not be used in financial investing, one key strategy is the introduction of new bounds that show how fast greedy methods can converge to near-optimal policies when combined with approximate value estimation. In general, past studies Kuleshov & Precup [2014] show that when presented with a static dataset, the ϵ-greedy algorithm excels, consistently finding an upper bound for regret. It performs comparatively better in environments with higher variance and lower sample size. However, this study deals with dynamic rather than static data that is simulated on a shorter timeframe than what is normally used for the multi-armed bandit, so a goal would be to evaluate whether the ϵ-greedy algorithm keeps up its strong performance in the stock market.

The UCB1 algorithm is a more recently explored algorithm Auer *et al.* [2002] with a more calculated method of exploring exploitation versus exploitation. Theoretically, UCB1 is able to achieve a logarithmic bound for regret as well, claiming that R(n) $\leq O(\log n)$, where R(n) refers to the cumulative regret, and n being the number of turns taken by the bandit algorithm. The finite time analysis conducted by Auer, Bianchi and Fischer showed that at each turn n, the UCB1 algorithm was still able to keep cumulative regret low. In general, the UCB1 algorithm is consistent at finding the upper bound for regret, as uncertainty about an arm's reward decreases as each arm is played. This makes the UCB1 algorithm strong when the number of turns is large, and it continues to perform well when variance is low as it is stronger at finding each arm's future expected value. A goal of this research on the UCB1 algorithm is to determine whether the UCB1 algorithm ends up staying consistent when faced with stock market data.

Aside from the ϵ-greedy and UCB1 algorithms, there have been a number of algorithms that have been introduced that will be expanded on during the literature review, such as Thompson Sampling, Boltzmann Exploration, and pursuit methods Kuleshov & Precup [2014]. However, these algorithms will not be explored in this study because they have not shown any advantages in the speed of reaching the upper bound for regret that the ϵ-greedy algorithm showed, or the consistency that the UCB1 algorithm showed.

3 Literature Review

The first paper read was an exposition on nonlinear trends in the stock market, also using the S&P 500 as an index. Abhyankar *et al.* [1997] This paper analyzes real-time data from indexes like the S&P 500, presenting an introduction to concepts applicable to the MAB problem. The results of the study show that stock market returns are not purely random or predictable, and by implementing linearity and Gaussianity tests, nonlinear trends are commonplace. Going forward into MAB research, it will be necessary to consider other distributions such as actual changes and normal distributions. Since the real-life stock market is nonlinear and unpredictable, it calls into question some methods used in the Multi-armed Bandit problem, which primarily deal with static datasets, meaning that they may not be able to deal as well with the nonlinear trends described in the paper.

The next four papers deal with specific bandit algorithms, each with their own applications. The first paper introduces six algorithms to minimize regret, such as the ϵ-greedy algorithm and Boltzmann Exploration, collecting empirical data supporting the argument that each algorithm excels in different turn-number and arm-number setups. Kuleshov & Precup [2014] The second half explores the application of the multi-armed bandit problem in clinical trials, measuring levels of regret in the treatment of opioid patients, reflecting a possible method of approach to a finance-based question. Different opioid patients can be compared to different companies in the stock market, but the methodologies cannot be directly compared due to the dynamic nature of the stock market.

The second paper presents a deeper dive than the first, explaining Thompson Sampling as a viable bandit algorithm. Chapelle & Li [2011] Regret data using Thompson Sampling is compared with other methods like Upper-Confidence-Bound methods using real-world and simulated datasets, giving a reason for the use of Thompson sampling as a standard baseline in bandit problems. Thompson Sampling works by sampling from the posterior distribution of each arm's reward probability and selecting the arm with the highest sampled value, balancing exploration and exploitation in a probabilistic manner. Although Thompson sampling methods will not be used in this study, this paper is a good baseline on understanding how bandit algorithms aim to balance exploration and exploitation.

The third paper is similar to the second in that it introduces the UCB1 algorithm and the UCB family in general. Auer *et al.* [2002] This family seeks to address the exploration-exploitation problem by considering the current average reward and the confidence interval for expected rewards. The authors derive tight theoretical bounds on regret, proving that the UCB1 algorithm achieves logarithmic regret over time—meaning that the performance loss due to uncertainty grows slowly compared to the number of rounds played. In contrast to purely greedy approaches, which may get stuck with suboptimal choices, UCB methods systematically reduce uncertainty, ensuring more informed decision-making over time. Their analysis demonstrated that UCB methods outperform simple strategies like ϵ-greedy in both theoretical and practical settings, particularly in environments where the reward distributions are stationary.

3.1 Comparing and Contrasting Bandit Algorithms

Through these three papers, we are able to contrast ϵ-greedy, Thompson sampling, and UCB1. As simulated in the Kuleshov and Precup paper, the ϵ-greedy algorithm Kuleshov & Precup [2014] is easy to implement and computationally efficient, but its exploration is uninformed and independent of the uncertainty in reward estimates, leading to linear regret unless ϵ is carefully decayed over time. In this paper, static data was simulated over 1000 turns, and the ϵ-greedy algorithm proved more viable at reaching logarithmic regret faster, although it saw a notable drop in efficiency once variance and arm count increased.

In contrast, UCB1 Auer *et al.* [2002] constructs confidence bounds for each arm's estimated reward and selects the arm with the highest upper bound, effectively favoring actions with greater uncertainty. In the same study, the UCB1 algorithm was also used to simulate the static data, and UCB1 reached a lower bound for regret more consistently, independent of variance and arm count.

Finally, Thompson Sampling Chapelle & Li [2011] takes a Bayesian approach by maintaining a posterior distribution over each arm's expected reward and selecting arms according to samples drawn from these posteriors. This enables a naturally probabilistic balance between exploration and exploitation, often yielding superior empirical performance, especially in complex or nonstationary environments. Unlike UCB1, Thompson Sampling can be easily extended to contextual settings, though it requires careful modeling and computation of posterior distributions. Since Thompson Sampling is the most theoretically complex out of the three, the only two algorithms tested in this study are the ϵ-greedy and UCB1 algorithms.

The final paper Gürsoy [2024] goes beyond the other introductions, applying bandit algorithms to a mean-field game model, modeling the average behavior of bodies within the model and applying the dilemma of choosing between known and unknown bandits. A mean field system is a mathematical model in which each individual component interacts not with every other component directly, but with the average effect of the entire population, similarly to how bandit algorithms interact with the average of past rewards when making decisions.This framework allows for studying complex systems where decentralized decision-making and uncertainty are prominent. The paper draws connections to real-world scenarios such as diffusion dynamics and stochastic systems, offering insight into how collective learning evolves over time. These results have direct implications for domains like finance, where the strategic behavior of numerous entities (e.g., companies or investors) resembles the interaction patterns seen in mean-field systems. In addition, the study provides a foundation for designing adaptive algorithms that account for group dynamics in large-scale, uncertain environments.

4 Methods

For this study's data, an API, which is a system that allows different software applications to exchange data, is the most efficient way to analyze finance data with Python. The YFinance API was used to retrieve daily open prices in USD for stock data sourced from Yahoo Finance, which updates following each trading day. For these particular trials, finance data was from March 1st, 2024, to March 1st, 2025, allowing the study to reflect economic conditions in the United States of America during the 2024-25 period. To situate individual stock behavior within a broader market context, data from the S&P 500 index, which represents 500 major U.S. companies, were incorporated. This allowed for comparative analysis between specific equities and general market performance. For all trials, the difference in open prices between days was what was used as the basis for data analysis, where each day's difference was entered into a NumPy array for further analysis.

4.1 Bernoulli Trial

The Bernoulli distribution models binary outcomes—success (1) or failure (0)—with a single parameter, the probability of success (p). In stock data, we can use it to simplify outcomes by framing trades as either profitable or not, reducing complex price movements to simple binary rewards. This simplification makes it easier to apply multi-armed bandit algorithms, which estimate the best option by learning the success probability of each stock over time-an optimal first trial in getting the algorithms to learn from past data.

To extract historical empirical data, 20 random companies on S&P500 from the YFinance API are called into Python and simulated with the two mentioned formulas (ϵ-Greedy and UCB-1). The Bernoulli trial creates a Bernoulli distribution by assigning a value of 1 to stocks that increase between days, and a value of 0 to stocks that decrease from the last day. These values are aggregated in a NumPy array for further analysis.

Code used to sample financial data from companies is below.

```
    import yfinance as yf
    import random
    import pandas as pd
    import numpy as np
    import sqlite3
    import plotly.express as px
```

Figure 1: Imports used for bandit algorithms

```
    Company_count = 20
random.seed(6)
random_tickers = random.sample(sp500_tickers, Company_count)
for ticker in random_tickers:
    stock = yf.Ticker(ticker)
open_binaries = {} #create the database
finance_data = yf.Tickers(random_tickers) #match random companies to YFinance. Is this
    needed?
for ticker in random_tickers:
    dat = finance_data.tickers[ticker]
    open_prices = dat.history(period = '1y')["Open"].values #retrieves open prices
    open_diff = np.diff(open_prices)
    open_binary = np.where(open_diff > 0, 1, 0) #searched up function, assigns bernoulli
    dist to data
    open_binaries[ticker]=(open_binary)
```

Figure 2: Sampling of companies for Bernoulli trial

Each algorithm was programmed using Python and applied to the sample companies. In the code below, NumPy arrays containing the data (Bernoulli and real values) was run through the algorithms using increasing K values, where K represents each turn that the algorithms are active. For each turn, variables

were assigned to the best reward at each K, the actual reward at each K, as well as how many times each algorithm has been picked, which informs the algorithms.

4.1.1 Epsilon-Greedy Algorithm

```
    for K in range(1, turns + 1):
    if np.random.rand() < EPS:
        chosen_array_index = np.random.choice(Company_count) #Exploration step
    else:
        chosen_array_index = np.argmax(average_rewards_Greedy) #Exploitation step

    best_reward_at_k = max(arrays[i][K % turns] for i in range(Company_count)) #Data that
     exploitation is based off of
    reward = arrays[chosen_array_index][K % turns]

    actual_score_Greedy += reward #Updates arrays after each turn
    counts_Greedy[chosen_array_index] += 1
    average_rewards_Greedy[chosen_array_index] += (reward - average_rewards_Greedy[
    chosen_array_index]) / counts_Greedy[chosen_array_index]
    Optimal_score += best_reward_at_k
    regret_Greedy = Optimal_score - actual_score_Greedy #Cumulative regret is calculated
```

Figure 3: ϵ-Greedy algorithm setup

4.1.2 UCB1 Algorithm

```
    for K in range(1, turns + 1):
    if 0 in counts_UCB1:
        chosen_array_index = np.argmin(counts_UCB1) #Every company is tried once
    else:
        ucb_values = average_rewards_UCB1 + np.sqrt((2 * np.log(K)) / counts_UCB1)
        chosen_array_index = np.argmax(ucb_values)

    reward = arrays[chosen_array_index][K % turns] #Updates arrays after each turn
    actual_score_UCB1 += reward
    best_reward_at_k = max(arrays[i][K % turns] for i in range(Company_count))

    counts_UCB1[chosen_array_index] += 1
    average_rewards_UCB1[chosen_array_index] += (reward - average_rewards_UCB1[
    chosen_array_index]) / counts_UCB1[chosen_array_index]
    Optimal_score += best_reward_at_k
    regret_UCB1 = Optimal_score - actual_score_UCB1 #Cumulative regret is calculated
```

Figure 4: UCB1 algorithm setup

4.2 Real-Value Trial

To accurately model the effects of investing in the stock market, algorithms from part 1 were set up using real differences in open prices between days, and the modified setup is shown below. The only main difference between the Bernoulli and Real-Value trials were that instead of using the openbinary variable, the Real-Value trial used the opendiff variable, ultimately creating a large difference in the results of the two different setups.

```
for ticker in random_tickers:
    stock = yf.Ticker(ticker)
open_binaries = {} #create the database
finance_data = yf.Tickers(random_tickers) #match random companies to YFinance. Is this
    needed?
for ticker in random_tickers:
    dat = finance_data.tickers[ticker]
    open_prices = dat.history(period = '1y')["Open"].values #retrieves open prices
    open_diff = np.diff(open_prices)
    open_binaries[ticker]=(open_diff) #open_diff is assigned rather than a binary value
    here
```

Figure 5: Modified setup for Real-Value trial

4.3 Real-Value Trial 2.0

At this point in the study, it was observed that the finance data was too short-term to produce a noticeable result with the two algorithms and randomly sampled companies. This branch-off of the real-value trial aimed to speed up the Bandit algorithms' learning processes and see if enough change could be made in a short period of time. Here, 15 random companies were sampled, and 2019 data from Apple, Microsoft, Tesla, Google, and AMD were added. These five companies performed strongly in 2019, so the desired effect of adding them was that the algorithms would hopefully learn to choose them more often. The modified setup for the real-value trial 2.0 is shown below.

```
Company_count = 20
best_tech_tickers = ["AAPL", "TSLA", "GOOGL", "MSFT", "AMD"]  # Top tech companies from
    2008
random_tickers = random.sample(sp500_tickers, Company_count - 5) + best_tech_tickers  #
    Ensure inclusion of top tech

open_binaries = {}  # Create the database
finance_data = yf.Tickers(random_tickers)  # Match companies to YFinance

# Store open price changes in binary form
for ticker in random_tickers:
    dat = finance_data.tickers[ticker]

    if ticker in best_tech_tickers:
        # Use data from 2008 for the 5 best tech companies
        open_prices = dat.history(start="2019-01-01", end="2019-12-31")["Open"].values
    else:
        # Use last year's data for the other 15 companies
        open_prices = dat.history(period="1y")["Open"].values

    # Compute binary changes
    open_diff = np.diff(open_prices)
    open_binary = np.where(open_diff > 0, 1, 0)
    open_binaries[ticker] = open_diff

# Find the minimum length to trim all arrays
min_length = min(len(arr) for arr in open_binaries.values())

# Trim all arrays to the same length
for ticker in open_binaries:
    open_binaries[ticker] = open_binaries[ticker][:min_length]
```

Figure 6: Modified setup for Real-Value 2.0 trial

4.4 Negative Control Trial

To confirm that both algorithms were working as intended, a dummy dataset was created with predictable arm selections for each algorithm. The three dummy sets of data were:

1. `[1,1,1,1,1,1,1,1,1,1]`
   `[0,0,0,0,0,0,0,0,0,0]`

```
[1,0,1,0,1,0,1,0,1,0]
```

The predicted effect of this dataset is that the ϵ-greedy algorithm quickly converges to the first arm and stays there, with only small occasional fluctuations. The UCB1 algorithm is projected to briefly sample the third arm but eventually converge to the first arm. Overall, the UCB1 algorithm should converge to the first arm more consistently, but the ϵ-greedy algorithm should do it faster.

2.
```
[1,1,1,1,1,1,1,1,1,1]
[0,0,0,0,0,0,0,0,0,0]
```

Since the first arm in this dataset is perfect, both algorithms should have an easy time decoding their reward distributions. It is projected that UCB1 will stick to arm 1 after receiving a payoff of 0 after trying arm 2, but ϵ-greedy should still occasionally switch to arm 2 for exploration.

3.
```
[10,10,10,10,10,10,10,10,10,10]
[0,0,0,0,0,0,0,0,0,0]
[2,4,6,8,10,12,14,16,18,20]
```

Although the third arm eventually has a higher payoff as K increases, we predict that it'll be hard for both algorithms to predict this shift in behavior. If the algorithms consistently knew when to switch to the third arm, it would be an indication that the algorithms are programmed incorrectly, but UCB1 should consistently sample the other two arms and eventually land on the third arm due to its possibility for variance.

5 Results

5.1 Negative Control Trial

The negative control trial brought up a bug that was addressed early on, allowing the algorithms to be implemented accurately. Previously, turns had been sampled incorrectly, with the algorithm choosing data one turn ahead, and graphing data from the dummy dataset allowed this issue to be addressed. Afterwards, graphs produced showing payout and regret were consistent with what was expected from both algorithms. In all three datasets, UCB1 generally produced a stronger payout, presumably due to a more calculated method of balancing exploration with exploitation. However, due to its inconsistencies, the ϵ-greedy algorithm occasionally produced a desirable result. An example dataframe showing the behavior of the two bandit algorithms when presented with the third dummy dataset is shown below.

Table 1: Payoff and Regret results for Negative Control Trial

K	Optimal Score	Actual Score-Greedy	Actual Score-UCB1	Regret-Greedy	Regret-UCB1
1	1	1	1	0	0
2	2	1	1	1	1
3	3	1	1	2	2
4	4	1	1	3	3
5	5	2	1	3	4
...	...	...	...	...	...
246	246	125	122	121	124
247	247	126	123	121	124
248	248	127	123	121	125
249	249	127	123	122	126
250	250	127	123	123	127

5.2 Bernoulli Trial

The results of the trial with a Bernoulli distribution are collected into a NumPy dataframe below. Values for the optimal score, the actual score for each algorithm, and cumulative regret for both algorithms is listed as columns on the dataframe.

Table 2: Payoff and Regret results for Bernoulli Trial

K	Optimal Score	Actual Score-Greedy	Actual Score-UCB1	Regret-Greedy	Regret-UCB1
1	1	1	1	0	0
2	2	1	1	1	1
3	3	1	1	2	2
4	4	1	1	3	3
5	5	2	1	3	4
...	...	...	...	...	...
246	246	125	122	121	124
247	247	126	123	121	124
248	248	127	123	121	125
249	249	127	123	122	126
250	250	127	123	123	127

The data can be visualized more clearly as a graphical representation of regret using Plotly. Here, K represents the number of turns, and regret data and payoff data for both algorithms have been plotted.

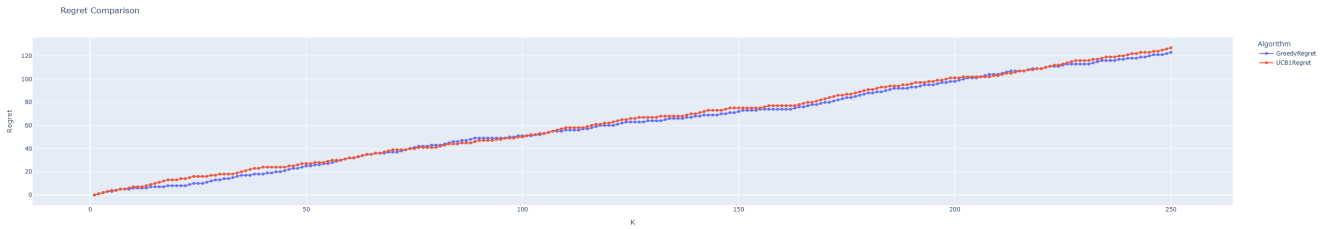

Figure 7: Regret Comparison Between Algorithms (Bernoulli)

The same format is used to visualize payoff between algorithms, again with K on the X axis representing the number of turns, the optimal score, as well as actual scores for the two algorithms.

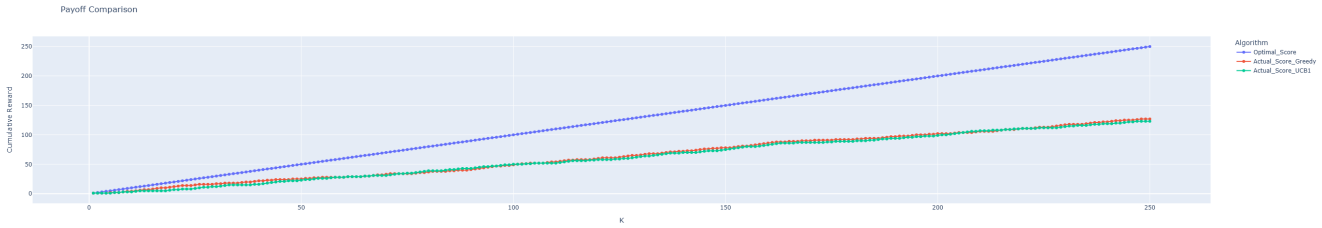

Figure 8: Payoff Comparison Between Algorithms (Bernoulli)

5.3 Real Value Trial

For the trial using real stock price changes, figures for payoff and regret are shown below.

Table 3: Payoff and Regret Results for Real Value Trial

K	Optimal Score	Actual Score-Greedy	Actual Score-UCB1	Regret-Greedy	Regret-UCB1
1	8.399	0.293	0.751	8.106	7.648
2	10.245	0.551	1.023	9.694	9.216
3	17.006	0.489	0.559	16.517	16.447
4	21.952	-0.275	0.301	22.226	21.651
5	27.737	1.406	1.101	26.331	26.636
...	...	...	...	...	...
250	1772.884	-20.768	-55.272	1793.652	1828.156
251	1780.220	-23.040	-55.062	1803.260	1835.282
252	1786.262	-23.021	-54.888	1809.283	1841.150
253	1791.037	-21.884	-53.885	1812.922	1844.922
254	1793.536	-22.065	-54.059	1815.601	1847.595

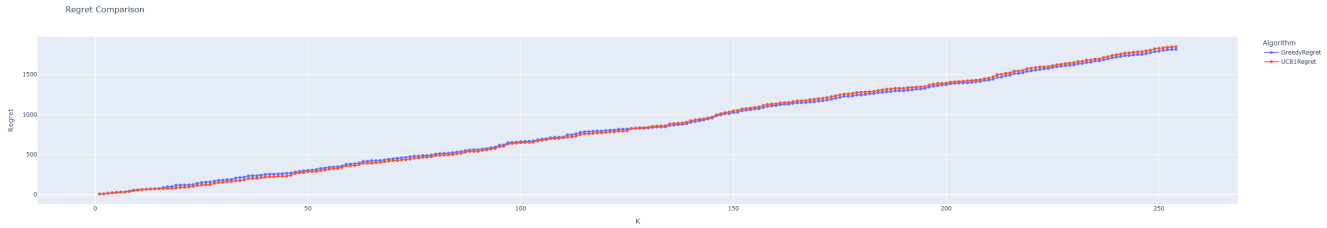

Figure 9: Regret Comparison Between Algorithms (Real Values)

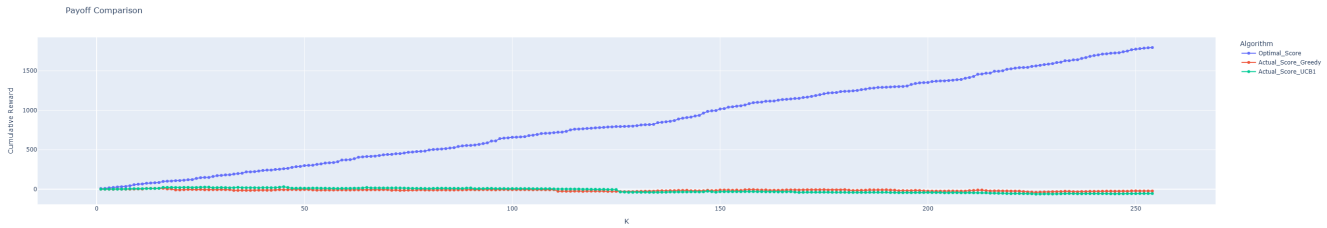

Figure 10: Payoff Comparison Between Algorithms (Real Values)

5.4 Real Value Trial 2.0

Real value trial 2.0 yielded a slightly different result, but the data was formatted in the same way as the previous two trials. Notably, the regret dataframe, Fig. 7, is placed along a graph of the logarithmic curve EQUATION NEEDED in order to show the slightly logarithmic behavior of the algorithms in Real Value Trial 2.0.

Table 4: Payoff and Regret Results for Real Value Trial 2.0

K	Optimal Score	Actual Score-Greedy	Actual Score-UCB1	Regret-Greedy	Regret-UCB1
1	4.081	-0.038	0.205	4.119	3.875
2	8.685	1.429	-0.155	7.256	8.840
3	13.866	-0.997	-2.412	14.862	16.278
4	22.911	-1.946	-0.219	24.857	23.130
5	34.370	-1.533	-1.046	35.904	35.417
...	...	...	...	...	...
245	3804.403	-19.254	15.664	3823.657	3788.739
246	3805.543	-19.155	14.714	3824.698	3790.829
247	3825.103	-18.716	15.153	3843.820	3809.950
248	3832.253	-18.079	14.243	3850.333	3818.010
249	3838.023	-17.607	13.483	3855.629	3824.540

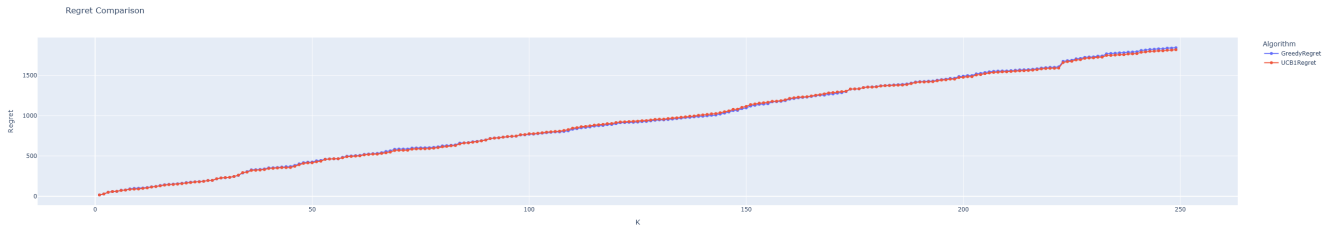

Figure 11: Regret result for Real Value Trial 2.0

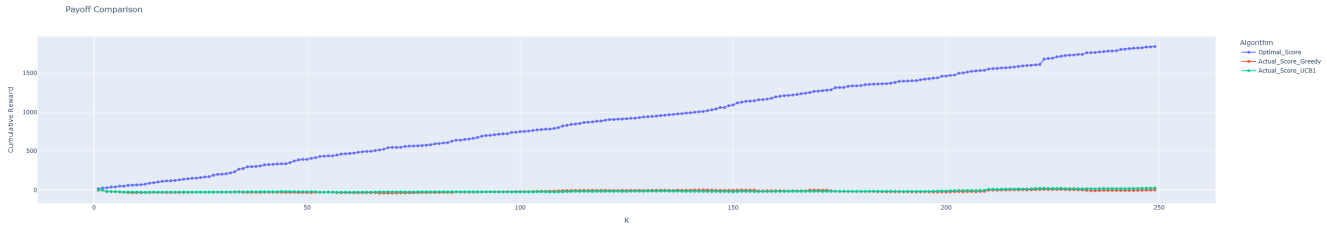

Figure 12: Payoff result for Real Value Trial 2.0

6 Discussion

After the four trials were run on finance data from YFinance, the discussion section serves to interpret the trends that showed up in PandaS dataframes, regret graphs, and payoff graphs.

The negative control group in Table 1 showed that the UCB1 algorithm offers a large payout for the Multi-armed bandit when presented with a controlled reward distribution that had predictable outcomes. However, the algorithms get more complicated when presented with real-world data, detailed in the Bernoulli trial and real value trial.

For the Bernoulli trial, data from Table 2 suggests that the ϵ-Greedy and UCB1 algorithms had no meaningful impact on the payout or regret when faced with real-world data. Both algorithms ended

with similar cumulative regret, with 123 and 127, respectively. Furthermore, in the classical Multi-armed Bandit scenario, regret should scale logarithmically as the algorithms "learn," or adapt to given data. However, Fig. 7 shows that regret scaled linearly for both algorithms, and both algorithms produced similar lines for regret.

For both algorithms, regret could be approximated using a line with slope 1/2, suggesting that the multi-armed bandit was unable to "learn" from previous data. In fact, random selection of financial investments would have produced a similar result, with an expected value of 1/2 for random selections. Finally, Fig. 7 confirmed that the bandit algorithms had little to no effect, as the slope of the optimal score line had almost exactly double the slope of the lines with bandit algorithms. This also meant that the multi-armed bandit couldn't learn in time. All three of these figures showed that finance data is too unpredictable to be implemented into the Multi-armed bandit.

The real value trial confirms the conclusions drawn from the Bernoulli trial. Table 3 shows that regret and payout again ended up being similar for both the ϵ-Greedy and UCB1 algorithm. Both algorithms had cumulative regret of above 1800, and both trials had negative payoff. This indicates that the algorithms were still not able to learn from the given finance data within a year. After visualizing the data in Fig. 9, regret still scaled linearly, with no real difference between algorithms. Similarly, the visualization in Fig. 10 showed that for both algorithms, payout hovered around zero, while the optimal score kept increasing as K increased. These three figures also showed that financial investments did not benefit from usage of the bandit algorithms in such a short period of time.

In the real value trial 2.0, the expected trend with the multi-armed bandit algorithms began to show up. As shown in Fig. 11, as K increased, regret showed a slight logarithmic trend. Although these specific conditions produced a logarithmically increasing regret graph, it still took a year for this trend to begin to appear, and would take even longer under normal circumstances. Therefore, it can be concluded that even with the conditions in the real value trial 2.0, bandit algorithms did not prove advantageous, though under extremely favorable market conditions, bandit algorithms could prove useful.

Overall, the results of this study show that the randomness of the stock market is too great to be accurately predicted by a bandit algorithm. Although Bandits have large applications in known reward distributions like normal distributions, the reward distribution of the stock market is unsuitable for the both the ϵ-Greedy and UCB1 algorithm.

7 Limitation/Future Directions

7.1 Limitation

When setting up the study, there were initially difficulties in getting the code to work properly. Due to some companies not reporting finance data on certain days, it was necessary to implement an algorithm that cut the finance data down to the length of the lowest company's data. Although the effect was minimal, the cutting of the finance data could have compromised the code's sequence by removing increases or decreases in the code that the bandit could have used. Another variable that was introduced into the study was the varying value of ϵ in the ϵ-Greedy algorithm. Although an epsilon value of 0.2-0.3 produced the most consistent results, further studies should be done to determine an optimal ϵ value.

Analysis of the results showed that usage of the bandit algorithms produced no real benefit, which could be due to a number of limitations. First, past studies done with the Multi-armed Bandit problem on real-world datasets analyzed data that was thousands of entries long Kuleshov & Precup [2014], Auer *et al.* [2002], and regret is often not observed until K reaches values above 1000. In the context of the stock market, running the bandit algorithm for that long is simply unrealistic, and finance data often doesn't stretch back far enough to achieve those results. Furthermore, another likely limitation for the use of bandit algorithms in investing is the dynamic nature of the stock market. Bandit algorithms are traditionally designed to operate on static data, where the underlying probability distribution of rewards for each action (or stock) remains constant over time. In such environments, the reward for pulling a specific arm is drawn from a fixed distribution, allowing the algorithm to "learn" optimal strategies more quickly and with higher confidence. However, financial markets are inherently non-stationary, with stock behaviors and reward structures evolving due to economic shifts, news events, and investor sentiment, which poses a challenge to bandit models that assume static conditions.

Although many companies have continuing positive or negative trends over time, the dynamic nature

evidently still provides difficulties when applying the bandit algorithms. Where the data stands, neither algorithm is beneficial in short-term investing.

7.2 Future Directions

After four trials were run on the data, neither algorithm necessarily outperformed the other in short-term investing, because both algorithms produced linear regret graphs, a sign that they both were ineffective. In order to achieve more conclusive data, it would be beneficial to repeat the study with modifications. In this dynamic study using bandit algorithms to try and select from a dynamic stock market that had unpredictable data, the implementation of algorithms that have been optimized to deal with static datasets was not optimal.

The strongest proposition for a future Multi-Armed Bandit study would be to have the bandit algorithm be able to look at past data starting from a certain period before the trial begins, because that is what is realistic given the stock market. In order to optimize data and give the bandits more data to consider, a modified trial will allow the algorithms to look at the previous year's data, in order to give it a stronger prediction for its investments. This trial will be conducted in the future with the UCB1 algorithm, yielding stronger payoff and reaching an upper bound for regret sooner, as the algorithm is able to learn.

During future trials, a variety of methods can be employed to determine if the concerns brought up in limitations are valid;

1. Run bandit algorithms over time periods ranging from 6 months to 5 years to ensure that the linearity or logarithmic behavior is not due to inadequate time for simulation.
2. Optimize epsilon constants on a range from 0.05 to 0.5, to ensure that ϵ-Greedy's results are not due to an epsilon parameter that has not been explored.
3. Incorporate more dummy data into future trials that accurately models the volatility of the stock market, where stock prices are random instead of following a pattern of reward distributions, as modeled in Table 1.

7.2.1 Effect of Time Period on Bandit Algorithms

The Multi-Armed bandit algorithms are also affected by the state of the United States stock market when the study was conducted in 2025. This was reflected in the differing results between Fig. 9 and Fig. 11. The former contains stock data mainly from 2025, where global economic issues have caused the stock market to experience negative trends, and the latter has a quarter of its stock data from 2019, where many big technology companies were on the rise. As a result, the predicted effects of the bandit algorithms were felt more strongly in Real Value Trial 2.0, which slightly resembled a logarithmic curve.

If this experiment were repeated in a different year with different economic trends, the results would be different as the bandit algorithms would learn at different rates. Therefore, for a future result that would justify the use of a bandit algorithm, a similar study conducted in a period of economic prosperity would see results that closely follow the classical Multi-armed Bandit problem. A strong future direction that would accurately gauge the bandits' performance in a different economic condition would be in a future year where global political and economic conditions are less inflamed. A period that is suitable for a future study can be measured using policy analysis and trade interdependence Clausing & Obstfeld [2024]. The success or failure of bandit algorithms under these conditions can be measured on whether regret is more linear or logarithmic.

As a baseline for future studies with bandits, regret data from a trial with Bernoulli distribution and finance data from 2010 to 2015 is listed in Fig. 13. As observed, regret follows a more logarithmic pattern than any of the trials conducted with 2025 data. This is especially evident when comparing this trial with the first Bernoulli trial Fig. 7. Both trials use a Bernoulli distribution to organize data, but the trial using 2025 data is much more linear.

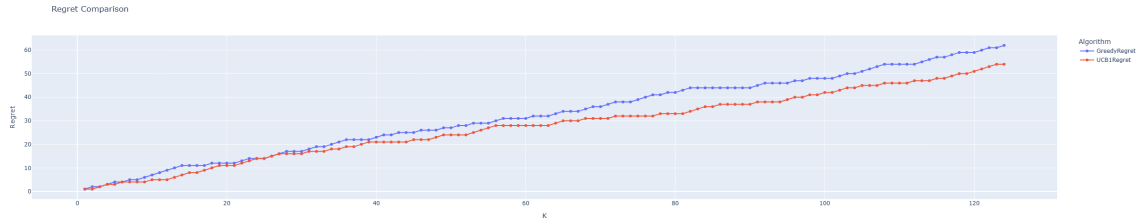

Figure 13: Regret result for data from 2010-2015

The likely explanation for the results experienced in Fig. 13 is the period of economic turbulence that occurred in the United States stock market from 2010-2015 Zeleza [n.d.]. The stock market was recovering from the 2008 financial crisis, with the S&P 500 nearly doubling. The economy shifted from sluggish recovery to more stable prosperity, driven by low interest rates and improving corporate earnings. Innovation, particularly in tech with companies like Apple, Amazon, and Tesla, fueled much of this growth, making the period a boom for the sector. The economic certainty during this period likely reflected itself in finance data, as there were likely more linearities where companies experienced deliberate growth for extended periods of time, driven by innovation. Consequently, the bandit algorithms were able to learn from this predictable data much more easily, leading to logarithmic regret.

8 Conclusion

The Multi-armed Bandit as a game theory problem has seen widespread use in static datasets, and this study saw its incorporation into the dynamic stock market of the United States. After importing finance data from the YFinance API, the year-long data was simulated using two theoretically effective algorithms: ϵ-Greedy and UCB1. The results of this study indicate that the Multi-armed Bandit algorithms, specifically ϵ-Greedy and UCB1, did not offer any significant advantage in stock market investing, as evidenced by their limited impact on payout and regret Fig. 10. Both algorithms failed to demonstrate meaningful learning or adaptation when faced with real-world financial data, as regret scaled linearly rather than logarithmically, contrary to expected behavior in traditional Multi-armed Bandit problems. However, in the real value trial 2.0, there was a slight emergence of a logarithmic trend in regret as the number of trials increased, Fig. 11 which suggests that, under more favorable market conditions, the bandit algorithms could eventually begin to show more promising results. While this trend took a full year to manifest, it offers hope that with further adjustments, such as incorporating historical data for better context, these algorithms could prove useful in longer-term investment strategies.

The accomplishments and results of the study also served as an indication that a dynamic dataset produced less desirable results, but also that results could vary given different economic conditions. First, the negative control trial using three sets of dummy data confirmed that bandit algorithms worked on datasets with varying values, with both the ϵ-Greedy and UCB1 algorithms testing many different companies before choosing the company with the objectively higher reward distribution, as planned Table 1. However, these results did not translate well into the unpredictable, dynamic economic conditions of the past year.

In the Bernoulli and real value trials, daily finance data from March 1st, 2024, to March 1st, 2025 was gathered from the YFinance API and simulated through the two bandit algorithms, and the regret result was rather linear Fig. 9 Fig. 7. However, a slight logarithmic trend for regret, which would indicate that the bandit is learning from the dataset, began appearing in the real value trial 2.0 Fig. 11. The setup for this trial was different in that 5 well-known technology companies that had well-performing stocks in 2019 were incorporated with 15 other random companies, in an attempt to create a more deliberately positive and predictable market. As expected, the bandit algorithms more easily found a logarithmic bound for regret, and this result served as the basis for future directions.

Future studies should focus on how different economic conditions influence the performance of bandit algorithms. The results from 2025, marked by global economic challenges, showed more linear regret patterns, suggesting that the algorithms struggled to learn effectively during periods of instability. In contrast, the data from 2010–2015 Fig. 13, a period characterized by post-crisis recovery and significant growth in the tech sector, displayed more logarithmic regret, indicating that the algorithms performed

more efficiently in a time of relative economic stability. This suggests that bandit algorithms may be better suited to environments with more predictable and consistent growth. Therefore, future studies should investigate how these algorithms behave during periods of economic prosperity or stability, as such conditions may allow the algorithms to learn at a more optimal rate. By testing these algorithms across varying economic climates, we can gain deeper insights into their effectiveness and limitations.

References

Abhyankar, Abhay, Copeland, Laurence S, & Wong, Woon. 1997. Uncovering nonlinear structure in real-time stock-market indexes: the s&p 500, the dax, the nikkei 225, and the ftse-100. *Journal of business & economic statistics*, **15**(1), 1–14.

Auer, Peter, Cesa-Bianchi, Nicolo, & Fischer, Paul. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, **47**, 235–256.

Chapelle, Olivier, & Li, Lihong. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, **24**.

Clausing, Kimberly A, & Obstfeld, Maurice. 2024. Letter from america: Trump's 2025 tariff threats. *Intereconomics*, **59**(4), 243–244.

Dann, Chris, Mansour, Yishay, Mohri, Mehryar, Sekhari, Ayush, & Sridharan, Karthik. 2022. Guarantees for epsilon-greedy reinforcement learning with function approximation. *Pages 4666–4689 of: International conference on machine learning*. PMLR.

Gürsoy, Kemal. 2024. Multi-armed bandit games. *Annals of operations research*, 1–13.

Kuleshov, Volodymyr, & Precup, Doina. 2014. Algorithms for multi-armed bandit problems. *arxiv preprint arxiv:1402.6028*.

Zeleza, Paul Tiyambe. The turbulent 2010s: Rising economic disequilibrium and shifting global hierarchies and hegemonies.