GROUP WORK PROJECT # 3 MScFE 622: Stochastic Modeling
GROUP NUMBER: 4186

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Marin Stoyanov	Bulgaria	azonealerts@gmx.com	
Panagiotis Lois	Grenoble (France)	loispanagiotis@outlook.com	

Statement of integrity: By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).

Team member 1	Marin Stoyanov
Team member 2	Panagiotis Lois
Team member 3	

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.
Note: You may be required to provide proof of your outreach to non-contributing members upon request.

N/A

Step 2: Portfolio selection problem and Huo pseudocode

The portfolio selection problem can be presented as a multi armed bandit problem where every investment instrument (for example stock) is the arm of a slot machine (bandit). It is common case for multi-armed bandit problem to address sequential decision making under uncertainty (like price of the stock - you don't know where it will be the next day) namely the dilemma of exploration vs exploitation. [1] This means that there have to be found the balance between exploitation (taking the best reward choice) or exploration (investing in assets that have not been fully explored yet and might potentially yield higher returns)

```
Pseudocode for Model 1: Sequential Portfolio Selection Problem (from the Huo paper) :

Define δ

Define N (number of trials)

Download historical data for the prices of a list of stocks that we are interested in

Turn the prices into returns.

Estimate the correlation structure and risk level.

Filter the list of the stocks to select a basket of K assets

For 1 to N do:

Choose portfolio of stocks and give them weights.

Find the returns that this portfolio gives.
```

Step 4 : Correlation of stocks

The correlation matrix provides us the correlation between all assets in our selection. There are plenty of options when it is needed to classify those assets according to their correlation estimate. One solution is to create a family of assets based on their correlation coefficient, clustering together those that are most correlated, and leaving the rest apart. While this could apply to a small number of assets chosen, we believe that it will be more complicated to model at a 30x30 matrix since it is likely to obtain a high number of different clusters. A simpler methodology consists of sorting the assets by their average correlation with the other assets. The disadvantage in this case is that we are unaware of which are the most correlated in our selection.

Instead, we propose to consider the maximum correlation coefficient when sorting the assets, and create two categories: Those whose maximum correlation is above the mean correlation of the sample, and those whose maximum correlation is below. Considering the maximum correlation instead of the average provides us with a better selection. Averaging the correlation coefficients may be very inaccurate in case an asset is highly correlated with one sector, and not to another. On the other hand, the maximum correlation shows us which assets are the most correlated in both sectors, not between each other, but in general.

Step 6: Pseudocode about the UCB problem

```
$Pseudocode\ for\ UCB\ algorithm:$
$Initialize:$
    $N(a) = 0\ for\ all\ a$
    $Q(a) = 0\ for\ all\ a$
$For\ each\ round:$
    $For\ each\ arm\ a:$
        $If\ N(a) > 0:$
            $UCB(a) = Q(a) + sqrt((2 * log(total\ count\ of\ rounds)) / N(a))$
        $else:$
            $UCB(a) = Infinity$
    $a\_max = argmax\_a\ UCB(a)$ # $Choose\ the\ arm\ which\ has\ maximum\ UCB$
    $Reward = pullBandit(a\_max)$ # $Pull\ the\ chosen\ arm\ and\ get\ the\ reward$
    $N(a\_max) = N(a\_max) + 1$ # $Increment\ the\ count\ of\ chosen\ arm$
    $Q(a\_max) = Q(a\_max) + (Reward - Q(a\_max)) / N(a\_max)$ # $Update\ the\ estimated\ value\ of\ chosen\ arm$
```

Where in this pseudocode:

$N(a)$ *is the number of times action* a *has been selected.*

$Q(a)$ *is the estimated value of action* a.

$UCB(a)$ *is the upper confidence bound of action* a.

$pullBandit(a)$ *is a function to pull arm* a *of the bandit and it returns a reward.*

Step 8: Pseudocode of the epsilon-greedy algorithm

First step of algorithm:

```
$Create\ a\ function\ to\ obtain\ the\ optimal\ action\ choice\ depending\ on\ a\ measure\ of\ past\ expected\ rewards.$
$if\ rand()\ <=\ eps:$
$random\ action$
$else:$
$greedy\ action$
```

Second step of the Algorithm:

```
$Create\ a\ function\ to\ update\ the\ expected\ reward$
$Set\ the\ new\ qvalue\ equal\ to\ the\ old\ qvalue\ +\ a\ variance\ 'alpha'$
```

Third step:

```
$Estimate\ the\ Algorithm\ over\ a\ number\ of\ steps$
1) $Create\ a\ for\ loop\ over\ the\ number\ of\ steps$
2) $Initialize\ the\ parameters$
3) $Use\ the\ optimal\ action\ algorithm\ to\ determine\ the\ type\ of\ action\ (random\ or\ greedy).$
4) $Use\ the\ expected\ reward\ algorithm\ based\ on\ the\ choice\ that\ comes\ from\ the\ previous\ function$
.5) $Calculate\ the\ average\ and\ optimal\ rewards\ across\ episodes$
```

Step 9: Results Comparison

a. <u>Describe the results of UCB and epsilon-greedy algorithm</u>

The context of our study is related to the theory of multi-armed bandits of reinforcement learning. The general situation that an agent has to treat can be described below:

- She repeatedly has to choose among different actions
- After each action, she receives a reward, or penalty, that depends on the probability distribution of the action

- Her goal is to over time maximize the expected total reward

Through a repetitive number of episodes, the agent learns the optimal action, the one that maximizes returns for example, based on the parameters given.

In our case, we use two variations of the multi-armed bandit model to modern portfolio selection, where the learner selects an asset to invest in a sequence of trials over a large set of assets.

For our analysis we used 28 stocks, 13 of them representing financial companies (JPM, WFC, BAC, C, GS, USB, MS, KEY, PNC, COF, AXP, PRU, SCHW), and 15 non-financial companies (KR, PFE, XOM, WMT, DAL, CSCO, HCP, EQIX, DUK, NFLX, GE, APA, F, REGN, CMS). We exclude from our selection the ticker STI, which is actually a ticker, and the BBT ticker (BB&T Corporation) that was merged in 2019.

Our data consists of financial returns of these stocks over the period of September and October 2008 (60 days). This period is particularly volatile, and the peak of the 2007-2008 Global Financial Crisis, where the US listed companies suffered multi-billion losses.

First, we have considered the Upper-Confidence-Bound (UCB) action selection. The UCB selects among the non-greedy actions taken into account the possibility being optimal. The algorithm first considers how close the estimates are to being maximal, and second the uncertainties in those estimates. Those uncertainties are given by a square-root term of the estimate α. This means that all actions will be eventually selected, but higher values estimates will be selected with increasing frequency over time.

The Upper-Confidence-Bound (UCB) method can be defined like this [5]:

$$A_t = arg\,max_a\{Q_t(a) + c\sqrt{\frac{ln\,t}{N_t(a)}}\}$$

Where:

$c > 0$ *is the parameter that control the degree of exploration*

To measure the performance of an armed-bandit algorithm we are based on two indicators: i) the average reward across steps, and ii) the average times that the actual optimal action was considered. The i) average reward measure, is used to indicate how fast the algorithm learns, while the later, the frequency of optimal action, is used to rate its overall performance. The results obtained using the UCB algorithm over the selected dataset are i) average reward = -1.12, and ii) average of optimal reward = 0.043.

The epsilon-greedy algorithm is a variation of the greedy action selection that promotes exploration about previous rewards. Specifically, the epsilon-greedy algorithm performs as the greedy selection, that is to select the action that maximizes immediate reward based on current knowledge, but with some probability ε to select randomly among all the possible actions.

The epsilon-greedy method can be defined [6] like this:

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{N_t(a)}[r_t - Q_{t-1}(a)]$$

For stationary problems (the reward probabilities do not change over time)

Where:

N_t *measures the number of times that a has been chosen up to and including time step t*

And in case the reward changes with time the method can be defined [6] like this :

$$Q_t(a) = Q_{t-1}(a) + \alpha[r_t - Q_{t-1}(a)]$$

Where:

$\alpha \in (0, 1)$ *is the constant step size parameter*

After some simplifications [6]:

$$Q_n = (1 - \alpha)^n Q_0(a) + \alpha \sum_{i=1}^{n} (1 - \alpha)^{n-i} r_i$$

In our study we choose an ε of 0.4 to apply to the epsilon-greedy algorithm. Our coefficient was that high because we were interested in observing the differences between the other algorithms considered. The results obtained using the epsilon-greedy algorithm over the selected dataset are i) average reward = -1.14, and ii) average of optimal reward = 0.039.

b. Compare the results of the Huo paper

In the Huo paper, the authors decided to implement another feature which is responsible for minimizing the risk (portfolio variance). They are using VaR and CVaR:

The VaR at a confidence level $\beta \in (0, 1)$ is defined like this:

$$VaR_\beta(X) := inf\{x \in \mathbb{R} : \mathbb{P}(x + X < 0) \leq 1 - \beta\}$$

And the CVaR t a confidence level $\gamma \in (0, 1)$ is defined [3] like this:

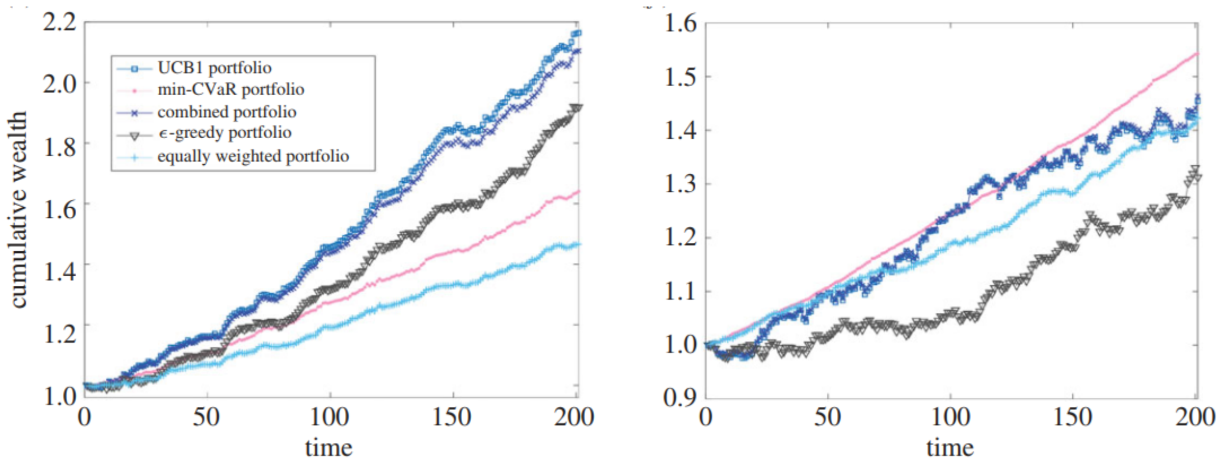
$$CVaR_\gamma := \frac{1}{1-\gamma}\int_{\gamma}^{1} VaR_\beta(X)d\beta$$

The authors are suggesting for better portfolio selection to use both methods k-armed bandit (maximizing the reward) with combination of a CVaR method for minimizing the portfolio variance (risk).

After applying these two methods a new portfolio is created while taking into consideration the results of the previous steps. Thus this new portfolio has a lower variance (risk) while still trying to maximize the rewards (returns).

This is a common trade-off: when you want to have lower risk you cannot have maximum return and vise versa when you want maximum returns you just cannot have minimum risk [4]. So if all being equal the Huo's portfolio has lower risk and lower returns compared to our portfolio generated by the 30-armed bandit UCB algorithm.

To compare our result with the one in Huo paper we would need his algorithm and to play around with it and create results based on the same stocks and on the same time period. Since we don't have it we cannot do good comparison, but we can do a deduction based on his results on Graph. 1 (Hou's results for low and high volatility respectfully left and right)



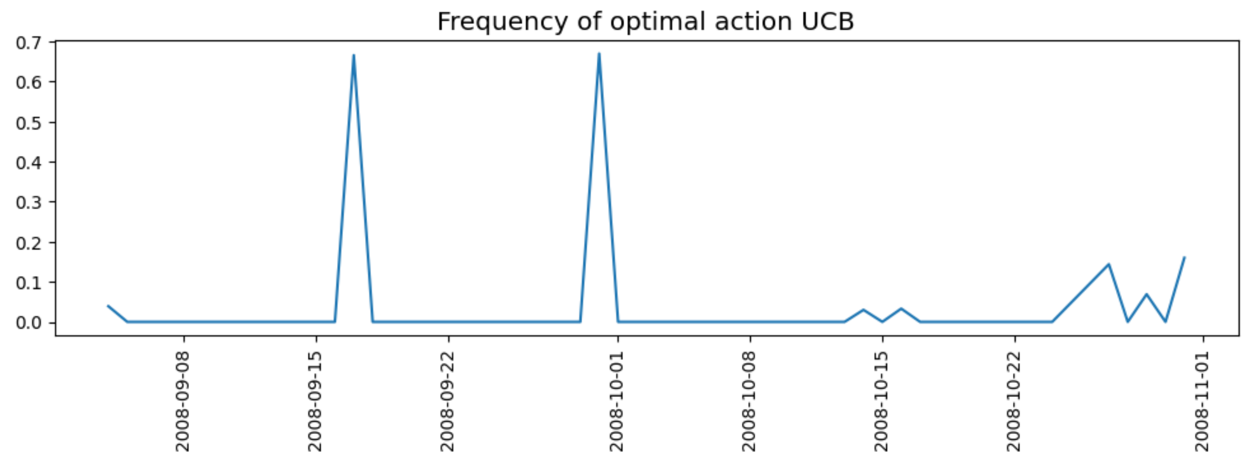
Graph.1. Portfolio comparison with low and high volatility (left chart and right chart)

The deduction is that his combiner portfolio is the best choice because it brings almost the best returns (very close to UCB) with low volatility (low risk).

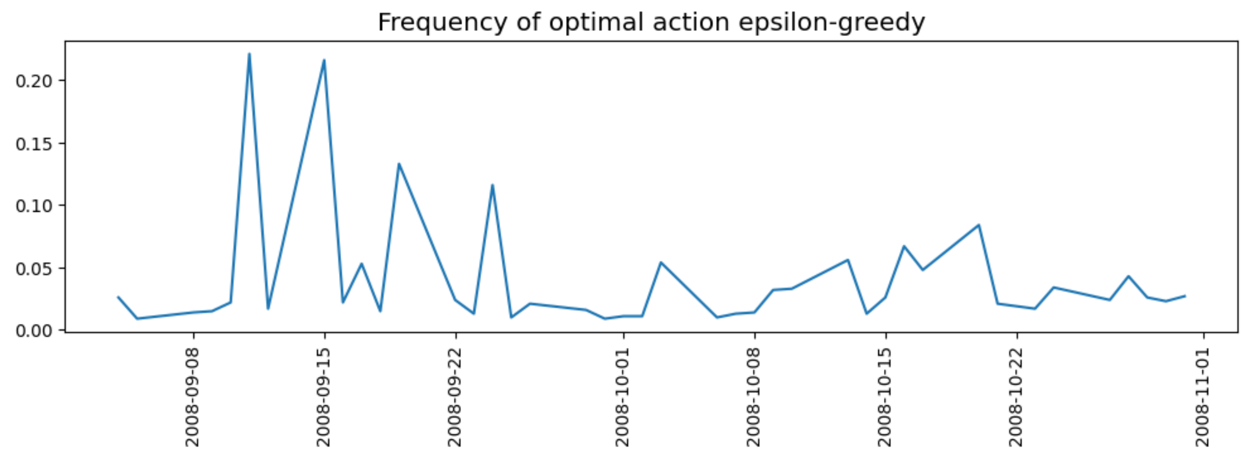
We can say that our UCB portfolio (with all everything being equal to Huo's simulation) is going to act the same as Huo's UCB1 portfolio so the conclusions could be applied the same way towards our simulation.

c. Present the key differences

To recognize the key differences between the results of the two algorithms we can represent graphically their frequency of optimal action:



Graph 2: Frequency of optimal action of UCB algorithm

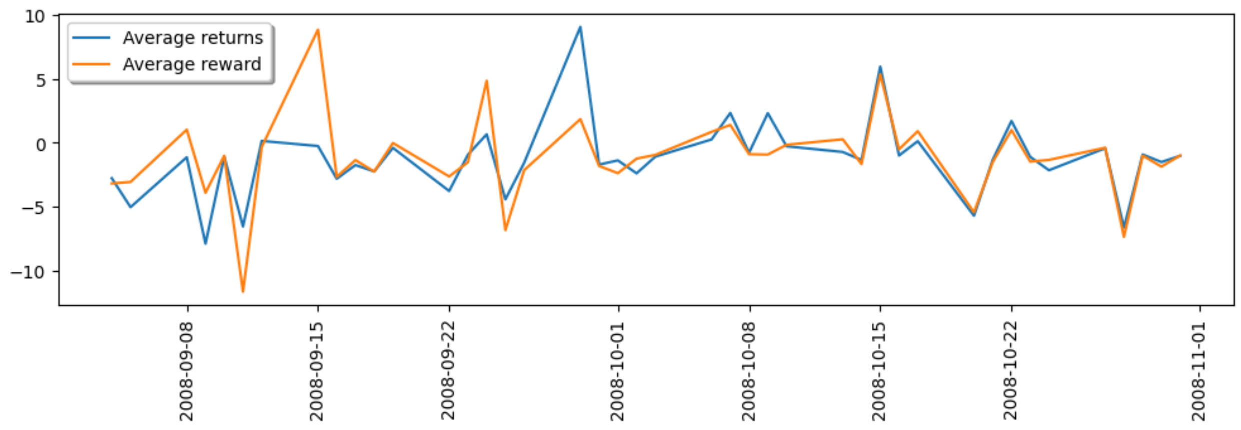


Graph 3: Frequency of optimal action of epsilon-greedy algorithm

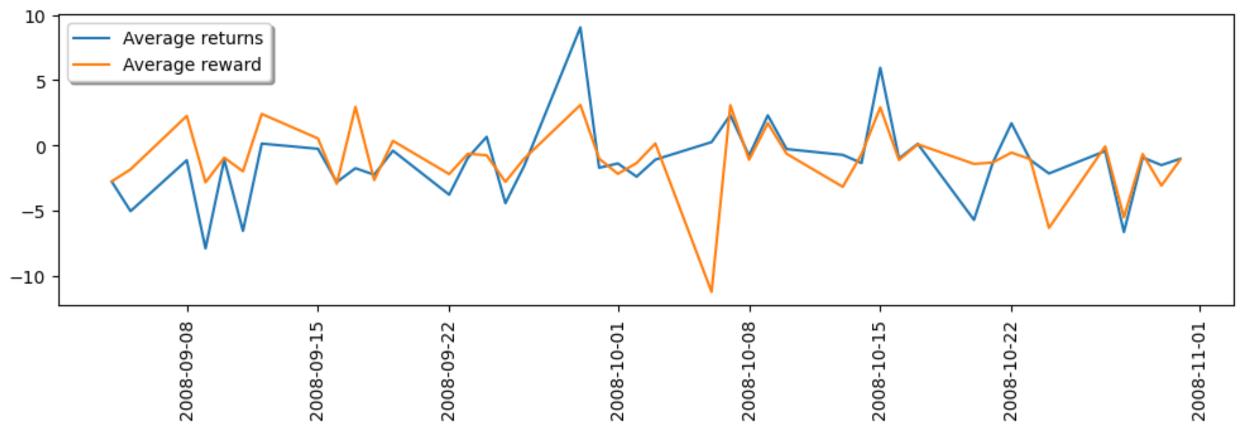
We can see that the epsilon-greedy algorithm acquires more frequently to an optimal choice but the UCB algorithm outperforms it by obtaining twice very high values. To compare them we will use the average of the optimal actions. We have provided earlier that the average of optimal rewards is 0.043 for the UCB, and 0.039 for the epsilon-greedy. Therefore, the UCB algorithm is more effective in exploiting the optimal action.

Similarly, we can compare the results of the average reward over all the time steps that we estimated around -1.12, and -1.14, for UCB, and epsilon-greedy respectively. Higher average rewards acoss steps means that the algorithms provide adaptation to the environment. Therefore, the UCB outperforms the epsilon-greedy algorithm in this metric as well.

Lastly, we will evaluate the performance of the two algorithms by comparing the average reward of the algorithm, over the average returns of our dataset. The bandit algorithm is preferred if it can track the variance in returns and acts accordingly.



Graph 4: Average reward over average returns using the UBC algorithm

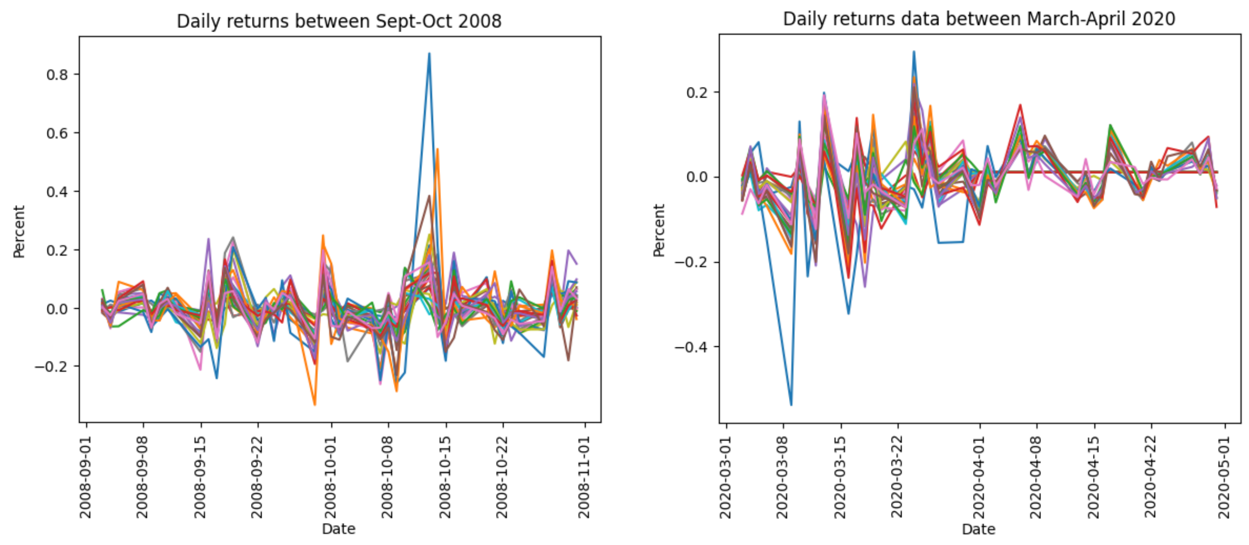


Graph 5: Average reward over average returns using the epsilon-greedy algorithm

Although the UCB takes a longer time to learn to choose the optimal action, it performs better in the long run, leading to more stable estimates of average rewards. Moreover, in the epsilon-greedy bandit we note a relationship between extreme observations and the average reward, which is not optimal in a tremble of the market.

Step 11: Analysis of UCB and epsilon-greedy algorithms using recent data

The data considered in the last part of the analysis were between March and April 2020. Compared to the period of September-October 2008, we are expected to find a stable market, and no systematic risk. The financial returns for the same 28 stocks over the two periods can be represented below:

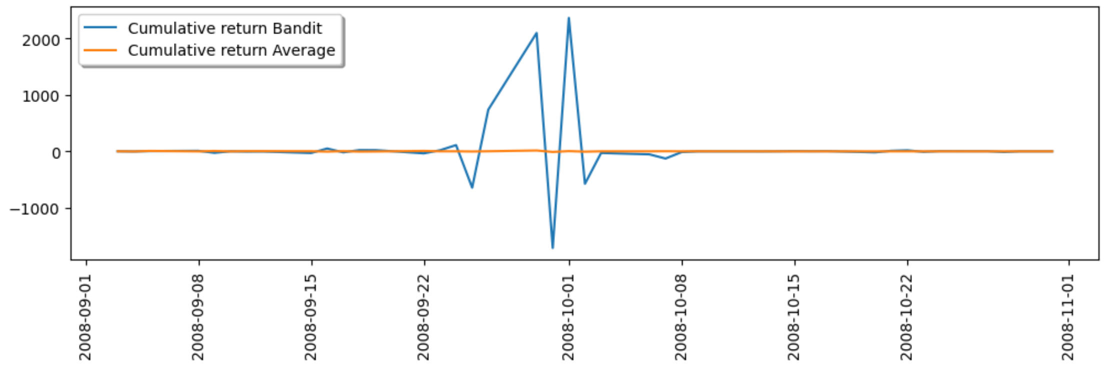


Graph 6: Presentation of financial returns of financial and non-financial companies for the periods of Sept-Oct 2008 and March-April.

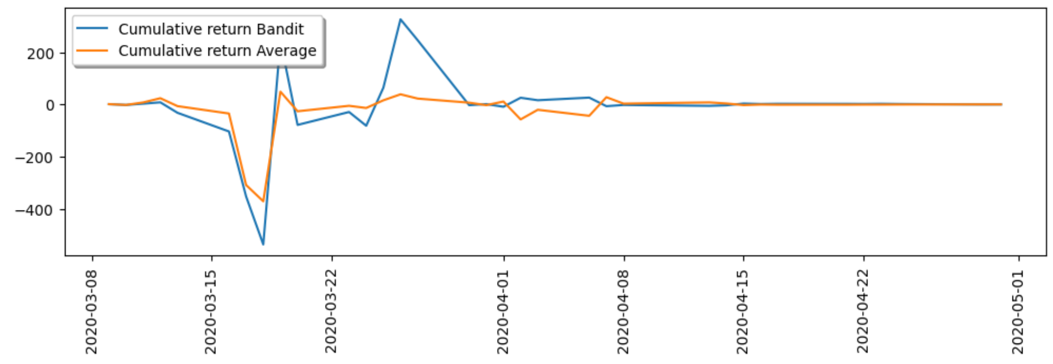
The parameters of exploitation have remained unchanged in our algorithms to evocate a relationship between the performance of the algorithms, and the volatility of the market. Thus, α=0.9 and ε=0.4 in the case of the epsilon-greedy algorithm, while ε=0 in the case of UCB. Also, the holding period was set to 5 days in both cases.

First, we will compare the frequency that the algorithms considered the optimal action over all episodes. On average, the UCB tested in 2020, has selected the optimal action obtained averaged at 6.3%, while the epsilon-greedy at 6.8%. With recent estimates, both algorithms performed seem to perform better, leading to more informed decisions.

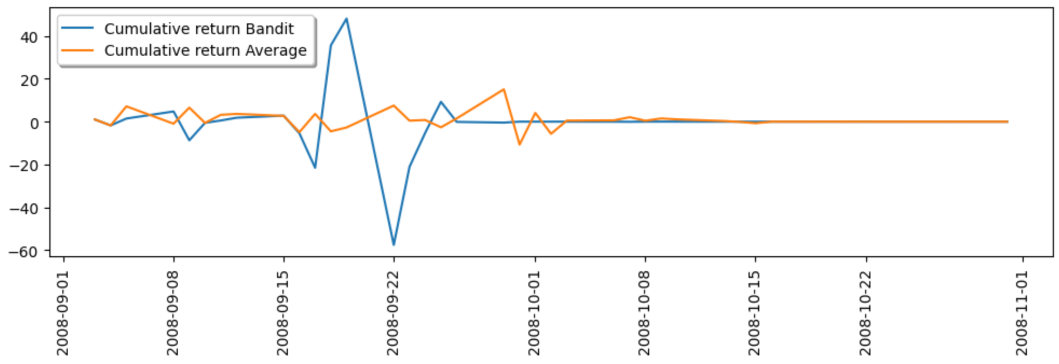
Next, we will compare the performance of the two algorithms in the portfolio selection problem. Specifically, we will display the cumulative return of an investment for both, UCB, and epsilon-greedy algorithm versus the equally weighted portfolio for both periods, Sept-Oct 2008, and March-April 2020.



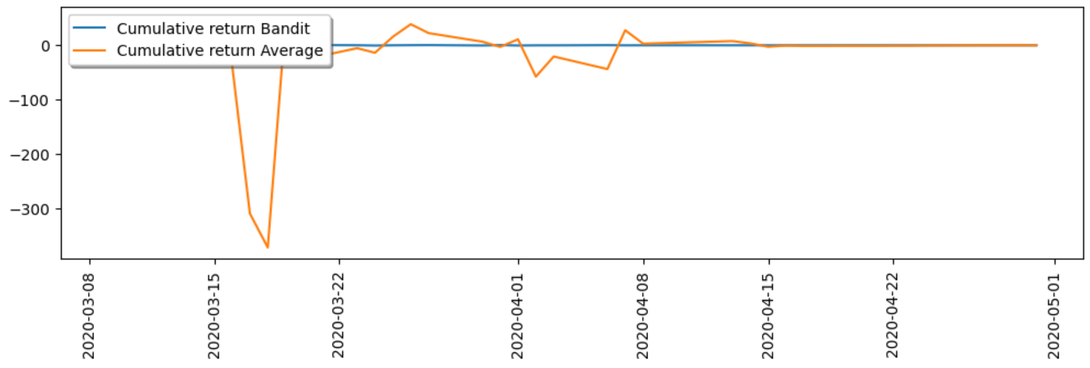
Graph 7: Cumulative return of epsilon-greedy versus the weighted portfolio in 2008



Graph 8: Cumulative return of epsilon-greedy versus the weighted portfolio in 2020



Graph 9: Cumulative return of UCB algorithm versus the weighted portfolio in 2008



Graph 10: Cumulative return of UCB algorithm versus the weighted portfolio in 2020 (right)

In general, the bandit algorithms perform better in both periods, taking into account the risk of financial losses, and making an optimal portfolio selection. We can see that the cumulative returns of the algorithms had much higher spikes in 2008, than 2020. This is of course related to the volatility of the market, and as we know it can lead to both higher profits, and higher losses for the investors.

References:

1. Huo, X. and Fu, F. (2017). Risk-aware multi-armed bandit problem with application to portfolio selection. Royal Society Open Science, 4(11), 171377. https://doi.org/10.1098/rsos.171377

2. Sutton, Richard S., and Andrew G. Barto. (2018) Reinforcement Learning: An Introduction. MIT Press

3. Pflug GC. (2000). Some remarks on the value-at-risk and the conditional value-at-risk. In Probabilistic constrained optimization (ed. S Uryasev), pp. 272–281. Boston, MA: Springer. https://link.springer.com/chapter/10.1007/978-1- 4757-3150-7_15.

4. David Blitz, Pim van Vliet; (Fall 2007). The Volatility Effect: Lower Risk Without Lower Return, Journal of Portfolio Management, pp. 102-113,, ERIM Report Series Reference No. ERS-2007-044-F&A,

5. Weiwei Shen , Jun Wang , Yu-Gang Jiang , Hongyuan Zha; (2015) Portfolio Choices with Orthogonal Bandit Learning, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)

6. J. Langford and T. Zhang. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In Advances in neural information processing systems, pages 817–824