GROUP WORK PROJECT # 3
GROUP NUMBER: 3573

MScFE 622: Stochastic Modeling

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Kwan Wei Ma	Hong Kong	kwanweima@gmail.com	
Alper Ülkü	Turkey	alperulku1970@gmail.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	Kwan Wei Ma
Team member 2	Alper Ülkü
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.

Team member 3 was withdrawn by the system.

Step 1: Summary of Huo paper

The exploration versus exploitation dilemma describes the trade-off in sequential decision-making under uncertainty between acting ambitiously to acquire new knowledge and behaving conservatively to benefit from existing knowledge. This is a frequent difficulty encountered in a variety of real-world situations, such as portfolio selection, clinical trials, online advertising, and robotics. In a single-state Markov decision process (MDP), the stochastic multi-armed bandit issue offers a mathematical framework for examining this conundrum.

The classic "multi-armed bandit" problem in portfolio selection is impacted by the inclusion of risk awareness since it introduces a new goal of taking into account risk rather than just maximizing cumulative reward. This variation has a tight connection to the portfolio selection issue, where risk management is a crucial consideration. The authors of this paper provide the Risk-Aware Upper Confidence Bound (RA-UCB) algorithm to solve the risk-aware multi-armed bandit issue with application to portfolio selection. They add risk awareness into the traditional problem.

In this paper the RA-UCB algorithm can be summarized as follows:

1. Compute the upper confidence bound (UCB) for each asset i:

$$UCB_i(t) = \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(t)}{n_i(t-1)}} + \lambda\sqrt{\hat{\sigma}_i^2(t-1)}$$

where $\hat{\mu}_i(t-1)$ is the empirical mean reward of asset i up to trial t-1, $n_i(t-1)$ is the number of times asset i has been selected up to trial t-1, $\hat{\sigma}_i^2(t-1)$ is the empirical variance of asset i up to trial t-1, and λ is the risk parameter.

2. Select the asset with the highest UCB:

$$I_t = \arg\max_i UCB_i(t)$$

3. Observe the reward $R_{I_t,t}$ and update the empirical mean and variance for asset I_t:

$$\hat{\mu}_{I_t}(t) = \frac{\hat{\mu}_{I_t}(t-1)\cdot n_{I_t}(t-1) + R_{I_t,t}}{n_{I_t}(t-1)+1}$$

$$\hat{\sigma}_{I_t}^2(t) = \frac{\hat{\sigma}_{I_t}^2(t-1)\cdot n_{I_t}(t-1) + (R_{I_t,t} - \hat{\mu}_{I_t}(t-1))^2}{n_{I_t}(t-1)+1}$$

4. Compute the portfolio weight for each asset i:

$$w_i(t) = \frac{(1-\lambda)\cdot UCB_i(t)}{\sum_{j=1}^{K}(1-\lambda)\cdot UCB_j(t)}$$

5. Rebalance the portfolio to the target weights $w(t) = (w_1(t), w_2(t), \ldots, w_K(t))$.

The final portfolio weights $w(T)$ are then outputted.

Figure 1: RA-UCB portfolio selection pseudocode (Huo et.al)

There are a number of potential uses for the research on the risk-aware multi-armed bandit issue with application to portfolio selection in the machine learning and quantitative finance sectors. It can be used, for instance, to improve internet advertising, where marketers must strike a balance between testing out new ads and capitalizing on the most effective ones. It can also be utilized in clinical studies, where it's important for researchers to strike a balance between using the best medicines and researching novel ones. The suggested algorithm can be used to optimize portfolio selection in quantitative finance, where investors must balance the investigation of new assets with the exploitation of the best-performing assets while taking risk management into account.

Step 2: Portfolio selection pseudocode

In order to achieve the investor's objectives, such as maximizing cumulative return in respect to some risk measure, the portfolio selection problem seeks to determine the best mix of assets that should be maintained in the portfolio. We alter the conventional multi-armed bandit paradigm to simulate portfolio selection by considering a financial market with a significant collection of assets, from which the learner selects a basket of K assets to invest in a series of N trials. Figure 1 below depicts the model's pseudocode. The learner's investment strategy consists of N mappings from the newly acquired knowledge to W in total.

Input:

- δ: length of the available history for estimating mean reterns
- N: trading horizon

Historical period:

1. Download prices for K assets
2. Calculate historical returns $H_{i,t}, t = 1, \ldots, \delta$

Investment period:

for t in $1, \ldots, N$:

1. Choose portfolio $\omega_t = (\omega_{1,t}, \ldots, \omega_{K,t})^T$ where
   - $\omega_t \in W$
   - $W = \{u \in \mathbb{R}_+^K : u^T \mathbb{1} = 1\}$
   - $\mathbb{1}$ is a column vector of ones
2. Observe $R_t = (R_{1,t}, \ldots, R_{K,t})^T$ where
   - $R_{i,t} = ln(P_{i,t+1}/P_{i,t})$
3. Receive reward $\omega_t^T R_t$

Figure 2: Sequential portfolio selection pseudocode (Model 1)

Step 3: Import and Structure Data for 2008

The sample data will be based on the daily returns of 27 selected assets between September and October 2008. The following are the asset lists for 13 financial institutions: **JPM, WFC, BAC, C, GS, USB, MS, KEY, PNC, COF, AXP, PRU, SCHW**. As STU and BBT merged in 2019, downloading the data of the two assets BBT and STI from Yahoo Finance was not possible. Thus we skipped these two.

14 non-financial institutions: **KR, PFE, XOM, WMT, DAL, CSCO, HCP, EQIX, DUK, NFLX; APA, F, REGN, CMS**. As HCP was listed in 2021, data of HCP in 2008 was not available. Thus we skipped the HCP. Disregarding these 3 assets, we have a total of 27 assets thus continue our calculations with 27 assets.

We combined the data into a suitable Python time series data structure as follows. We defined a function called:

DownloadAssetsConvertDataframe(tickers, startTime, endTime)

That outputs the prices and log returns as time series data frames called Prices0 and Returns0.Here are some lines from Prices0 dataframe:

```
1  Prices0
2
```

	JPM	WFC	BAC	C	GS	USD	MS	KEY	PNC	COF	...	DAL	CSCO	EQIX	DUK	NFLX	GE	APA	F	REGN	CMS
2008-09-02	26.68	21.06	25.74	150.86	130.56	1.71	30.49	8.48	49.77	34.99	...	8.13	16.48	63.43	26.49	4.41	115.83	85.02	2.71	20.58	8.23
2008-09-03	27.17	20.92	26.52	154.81	132.37	1.57	31.14	8.56	50.23	35.57	...	8.07	16.18	62.80	26.15	4.42	116.00	85.93	2.74	21.80	8.13
2008-09-04	25.94	20.02	24.63	144.47	127.07	1.44	29.79	8.02	49.30	33.75	...	7.94	15.46	60.73	26.41	4.27	112.47	87.97	2.64	20.38	8.14
2008-09-05	27.09	21.05	25.94	150.55	128.92	1.47	30.54	8.73	50.35	34.83	...	7.81	15.45	60.72	26.44	4.24	113.20	89.09	2.65	19.07	8.06
2008-09-08	28.43	22.64	27.95	160.41	134.05	1.47	31.95	9.25	52.03	37.96	...	7.61	16.22	60.73	27.30	4.31	118.11	87.77	2.73	18.90	8.31
2008-09-09	27.00	21.03	26.17	149.05	127.68	1.36	29.83	8.85	49.37	35.44	...	7.29	15.94	58.05	27.65	4.11	114.17	80.39	2.64	18.52	8.11
2008-09-10	26.96	21.39	26.07	147.47	124.46	1.37	28.74	8.46	48.02	35.50	...	6.98	15.95	56.94	27.30	4.11	114.05	84.24	2.68	19.43	8.19
2008-09-11	28.50	22.84	26.61	146.91	124.02	1.37	28.58	8.62	49.49	36.09	...	7.17	16.00	59.62	27.83	4.12	114.33	87.16	2.81	20.58	8.21
2008-09-12	28.17	23.14	27.15	141.78	121.79	1.37	27.49	9.00	49.45	35.78	...	7.21	16.28	60.28	27.95	4.21	108.61	89.86	2.95	20.71	8.34
2008-09-15	25.31	20.92	21.37	120.31	107.01	1.28	23.77	8.34	47.72	34.82	...	7.14	15.53	56.61	27.28	4.02	99.88	83.46	2.85	20.45	8.00
2008-09-16	27.87	23.57	23.78	124.34	105.05	1.29	21.19	8.50	52.03	38.20	...	8.81	15.82	58.67	27.56	4.07	101.75	87.02	3.04	21.64	7.67

And this ends with:

2008-10-15	26.54	22.50	19.17	128.13	89.36	0.68	13.55	7.01	42.08	29.41	...	6.59	11.50	42.06	22.62	3.22	79.21	52.19	1.38	17.11	5.54
2008-10-16	27.92	22.87	19.52	125.52	88.77	0.75	13.97	7.10	42.01	30.14	...	7.84	12.32	42.28	23.72	3.35	81.84	60.49	1.36	19.29	6.14
2008-10-17	27.12	21.63	18.70	117.47	90.27	0.73	14.38	6.79	39.64	31.09	...	8.15	12.43	44.44	23.14	3.33	80.77	59.07	1.46	18.75	6.13
2008-10-20	28.03	21.75	19.64	119.13	95.96	0.77	14.78	6.56	40.82	31.37	...	8.34	13.16	48.15	25.48	3.40	82.87	65.21	1.40	19.40	6.73
2008-10-21	27.40	22.02	19.29	111.94	95.84	0.70	15.10	7.37	40.36	30.28	...	8.65	12.39	46.03	23.98	2.98	83.73	63.40	1.30	19.25	6.43
2008-10-22	25.63	21.12	18.24	105.15	90.87	0.64	14.45	7.24	38.06	28.35	...	8.80	12.08	42.18	23.60	2.96	78.01	55.67	1.26	18.69	6.12
2008-10-23	26.10	21.14	18.51	103.50	86.01	0.61	13.52	6.78	38.92	27.82	...	7.80	11.96	41.53	25.05	2.97	77.36	58.75	1.20	17.70	6.17
2008-10-24	24.43	20.86	16.96	95.84	79.53	0.60	12.35	6.81	40.29	27.50	...	7.37	11.32	41.53	23.87	2.71	73.36	55.84	1.21	16.15	6.07
2008-10-27	23.44	20.80	16.52	92.60	73.58	0.59	10.26	6.68	40.12	26.79	...	6.79	11.17	40.49	24.01	2.56	72.95	51.56	1.22	15.46	5.84
2008-10-28	25.92	23.25	18.53	105.86	74.12	0.71	11.36	7.98	44.77	31.08	...	7.23	12.71	44.59	25.48	2.94	80.19	57.28	1.29	16.75	6.21
2008-10-29	24.62	21.67	17.96	101.92	77.36	0.66	11.03	8.18	42.98	29.48	...	7.08	12.40	46.71	24.81	3.11	79.00	59.93	1.30	17.43	6.15
2008-10-30	25.94	21.48	18.33	104.79	72.17	0.74	12.03	8.28	44.11	29.69	...	8.46	12.35	49.09	25.71	3.25	79.62	63.25	1.37	18.14	6.37

43 rows × 27 columns

Here are some lines from Returns0 data frame which consists of column-wise log returns series for each asset:

```
1  np.round(Returns0,4)
```

	JPM	WFC	BAC	C	GS	USD	MS	KEY	PNC	COF	...	DAL	CSCO	EQIX	DUK	NFLX	GE	APA	F	REGN	CMS
2008-09-03	0.0182	-0.0067	0.0299	0.0258	0.0138	-0.0854	0.0211	0.0094	0.0092	0.0164	...	-0.0074	-0.0184	-0.0100	-0.0129	0.0023	0.0015	0.0106	0.0110	0.0576	-0.0122
2008-09-04	-0.0463	-0.0440	-0.0739	-0.0691	-0.0409	-0.0864	-0.0443	-0.0652	-0.0187	-0.0525	...	-0.0162	-0.0455	-0.0335	0.0099	-0.0345	-0.0309	0.0235	-0.0372	-0.0674	0.0012
2008-09-05	0.0434	0.0502	0.0518	0.0412	0.0145	0.0206	0.0249	0.0848	0.0211	0.0315	...	-0.0165	-0.0006	-0.0002	0.0011	-0.0071	0.0065	0.0127	0.0038	-0.0664	-0.0099
2008-09-08	0.0483	0.0728	0.0746	0.0634	0.0390	0.0000	0.0451	0.0579	0.0328	0.0861	...	-0.0259	0.0486	0.0002	0.0320	0.0164	0.0425	-0.0149	0.0297	-0.0090	0.0305
2008-09-09	-0.0516	-0.0738	-0.0658	-0.0735	-0.0487	-0.0778	-0.0687	-0.0442	-0.0525	-0.0687	...	-0.0430	-0.0174	-0.0451	0.0127	-0.0475	-0.0339	-0.0878	-0.0335	-0.0203	-0.0244
2008-09-10	-0.0015	0.0170	-0.0038	-0.0107	-0.0255	0.0073	-0.0372	-0.0451	-0.0277	0.0017	...	-0.0435	0.0006	-0.0193	-0.0127	0.0000	-0.0011	0.0468	0.0150	0.0480	0.0098
2008-09-11	0.0555	0.0656	0.0205	-0.0038	-0.0035	0.0000	-0.0056	0.0187	0.0302	0.0165	...	0.0269	0.0031	0.0460	0.0192	0.0024	0.0025	0.0341	0.0474	0.0575	0.0024
2008-09-12	-0.0116	0.0130	0.0201	-0.0355	-0.0181	0.0000	-0.0389	0.0431	-0.0008	-0.0086	...	0.0056	0.0173	0.0110	0.0043	0.0216	-0.0513	0.0305	0.0486	0.0063	0.0157
2008-09-15	-0.1071	-0.1009	-0.2394	-0.1642	-0.1294	-0.0680	-0.1454	-0.0762	-0.0356	-0.0272	...	-0.0098	-0.0472	-0.0628	-0.0243	-0.0462	-0.0838	-0.0739	-0.0345	-0.0126	-0.0416

And this ends with:

2008-10-20	0.0330	0.0055	0.0490	0.0140	0.0611	0.0533	0.0274	-0.0345	0.0293	0.0090	...	0.0230	0.0571	0.0802	0.0963	0.0208	0.0257	0.0989	-0.0420	0.0341	0.0934
2008-10-21	-0.0227	0.0123	-0.0180	-0.0623	-0.0013	-0.0953	0.0214	0.1164	-0.0113	-0.0354	...	0.0365	-0.0603	-0.0450	-0.0607	-0.1319	0.0103	-0.0281	-0.0741	-0.0078	-0.0456
2008-10-22	-0.0668	-0.0417	-0.0560	-0.0626	-0.0533	-0.0896	-0.0440	-0.0178	-0.0587	-0.0659	...	0.0172	-0.0253	-0.0873	-0.0160	-0.0067	-0.0708	-0.1300	-0.0313	-0.0295	-0.0494
2008-10-23	0.0182	0.0009	0.0147	-0.0158	-0.0550	-0.0480	-0.0665	-0.0656	0.0223	-0.0189	...	-0.1206	-0.0100	-0.0155	0.0596	0.0034	-0.0084	0.0538	-0.0488	-0.0544	0.0081
2008-10-24	-0.0661	-0.0133	-0.0875	-0.0769	-0.0783	-0.0165	-0.0905	0.0044	0.0346	-0.0116	...	-0.0567	-0.0550	0.0000	-0.0483	-0.0916	-0.0531	-0.0508	0.0083	-0.0916	-0.0163
2008-10-27	-0.0414	-0.0029	-0.0263	-0.0344	-0.0778	-0.0168	-0.1854	-0.0193	-0.0042	-0.0262	...	-0.0820	-0.0133	-0.0254	0.0058	-0.0569	-0.0056	-0.0797	0.0082	-0.0437	-0.0386
2008-10-28	0.1006	0.1114	0.1148	0.1338	0.0073	0.1851	0.1018	0.1778	0.1097	0.1485	...	0.0628	0.1292	0.0965	0.0594	0.1384	0.0946	0.1052	0.0558	0.0801	0.0614
2008-10-29	-0.0515	-0.0704	-0.0312	-0.0379	0.0428	-0.0730	-0.0295	0.0248	-0.0408	-0.0529	...	-0.0210	-0.0247	0.0464	-0.0266	0.0562	-0.0150	0.0452	0.0077	0.0398	-0.0097
2008-10-30	0.0522	-0.0088	0.0204	0.0278	-0.0694	0.1144	0.0868	0.0122	0.0260	0.0071	...	0.1781	-0.0040	0.0497	0.0356	0.0440	0.0078	0.0539	0.0524	0.0399	0.0351

42 rows × 27 columns

And we visualize the cumulative performance of each stock during the sample period.

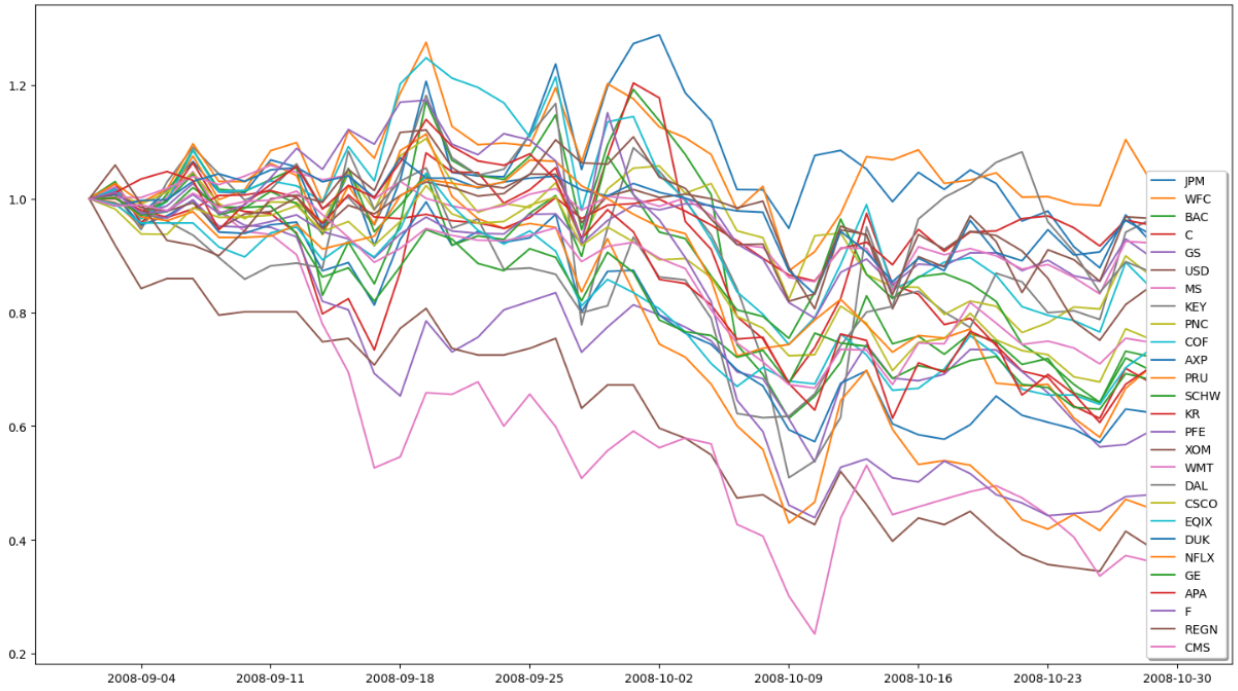

Figure 3: Price time series for 27 assets (2008 data)

Step 4: Correlation Matrix

Making use of the python code written within the defined function, PlotCorrelations(log_returns, CORR_FILTER), we managed to have the cross correlation matrix together with heatmaps.

The construction of the correlation matrices is depicted below. Due to the size of the data, a heatmap has been created for easier visualization of data.

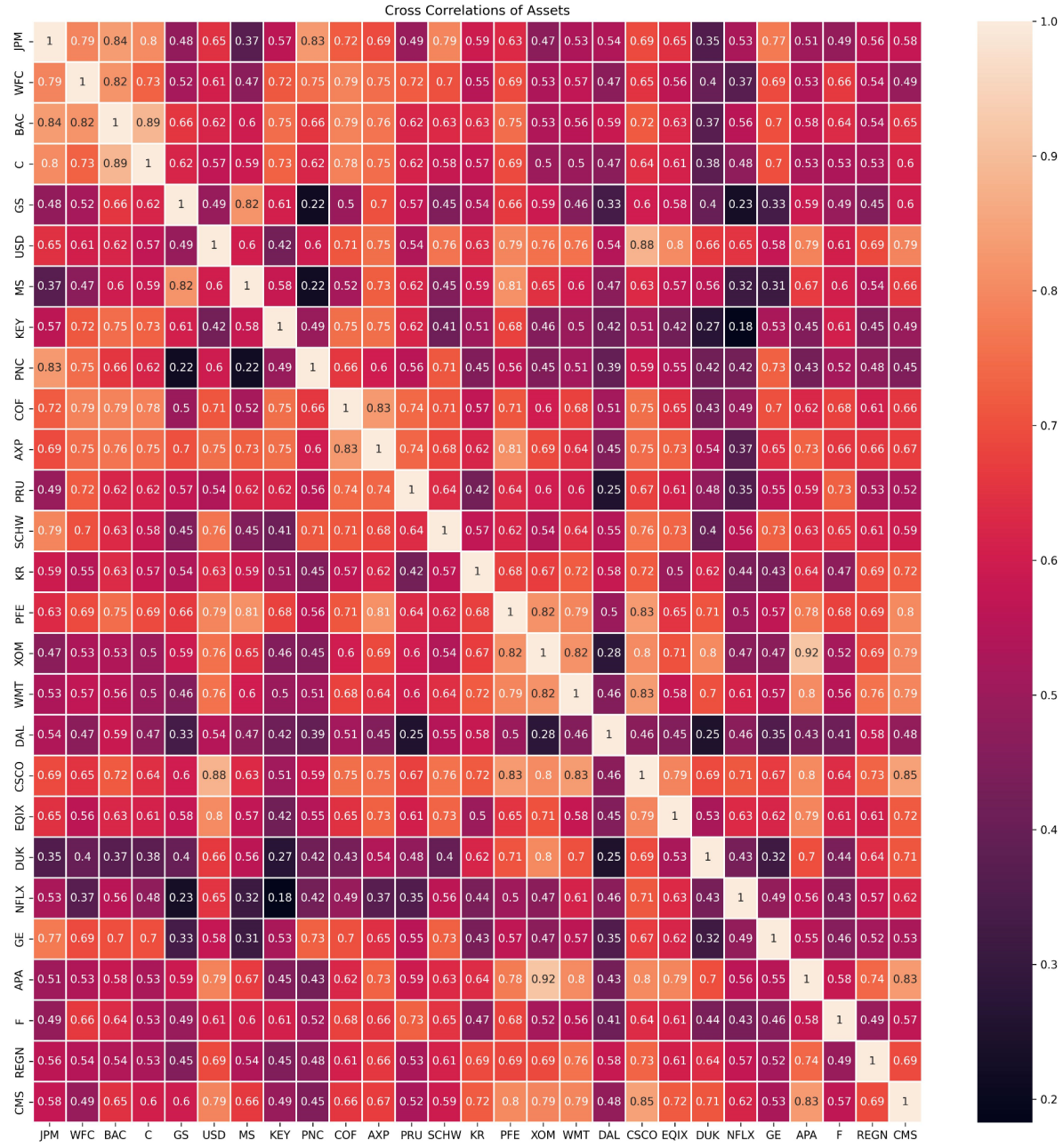

Figure 4: 27x27 correlation matrix with heatmap

Here are our observations for Figure 2.

1. We can easily observe that the financial universe assets (**JPM, WFC, BAC, C, GS, USB, MS, KEY, PNC, COF, AXP, PRU, SCHW)** shown in upper left half of the figure carry more "heat" than the rest of the figure, meaning asset returns have higher correlations with each other and less correlations with non-financial institutions. Since they are closely related or affected from each other, while forming a portfolio, there would be no need to include more than one or two of

these assets, as the portfolio diversification would be suboptimal if we chose abundantly many assets from these financial institutions .

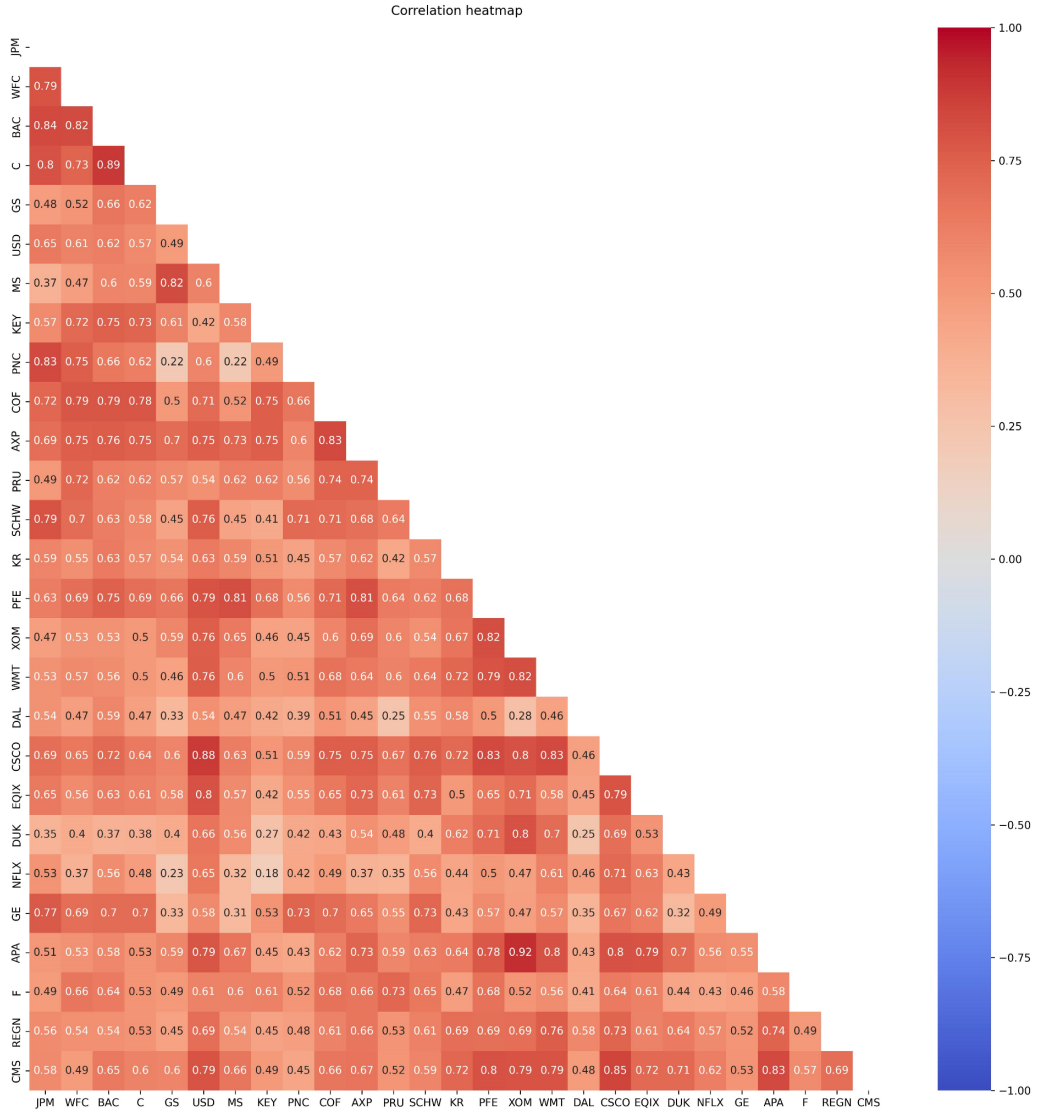

Figure 5: Triangular 27x27 correlation matrix with heatmap

2. This time we use only the lower triangular half for making assets more visible for their cross correlations.

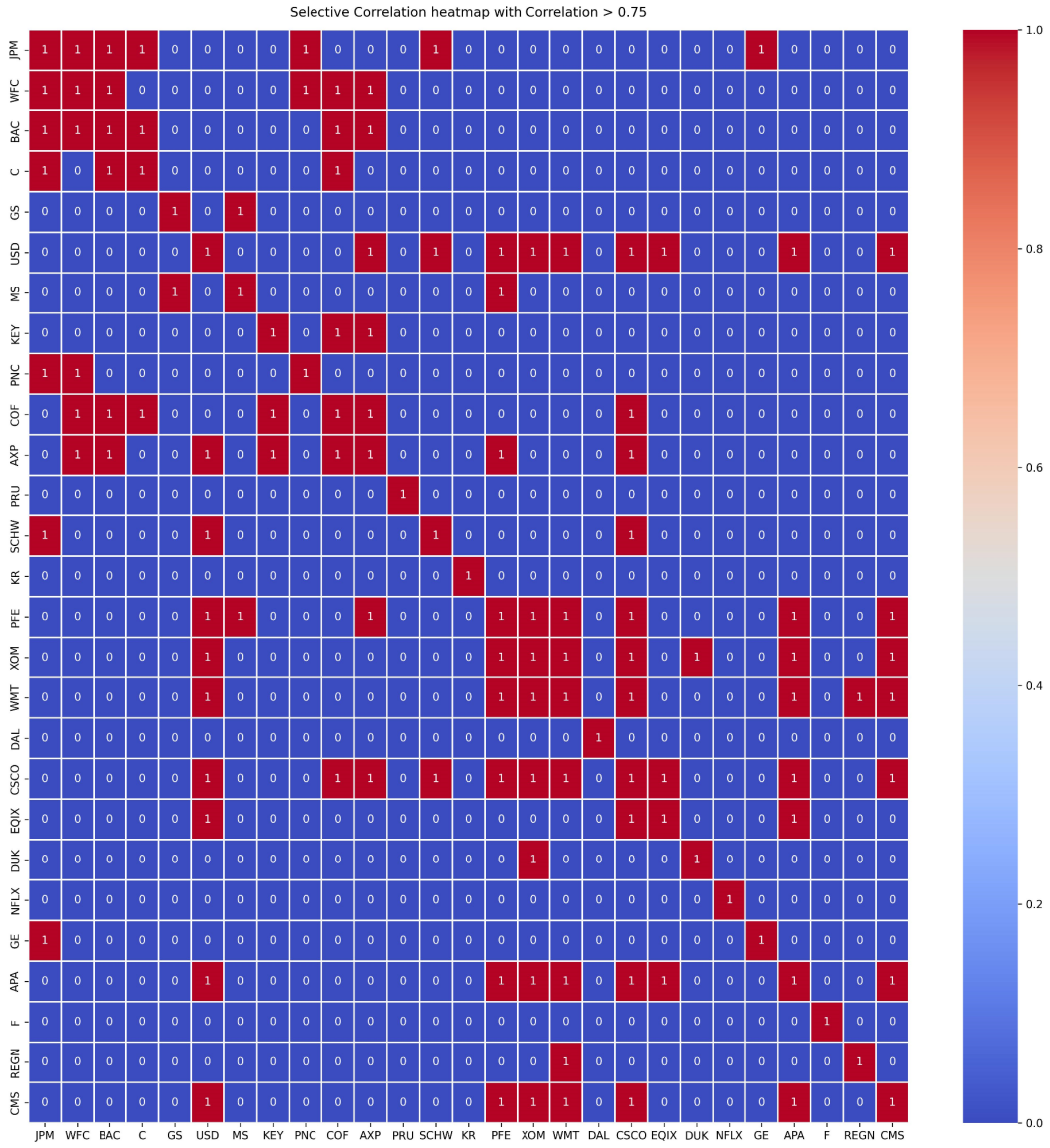

Figure 6: 27x27 correlation matrix with tagged correlations greater than 0.75

3. Here we filtered the assets for high correlation for tidying them up so that we make the discrimination for CORR_FILTER > 0.75 and filter them next time, so that these assets will not appear in the same portfolio.

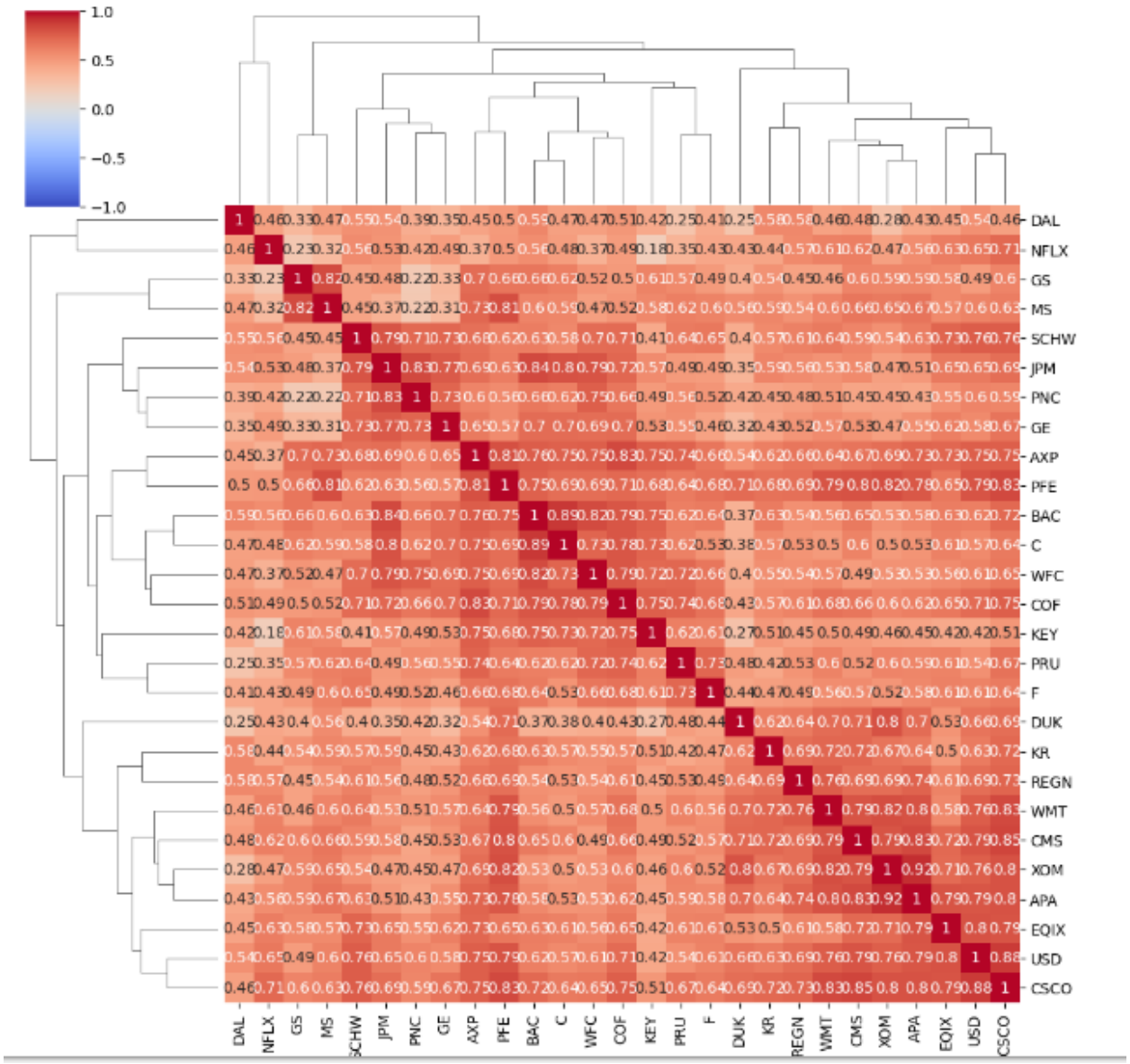

Figure 7: 27x27 correlation matrix with cluster map

4. This time by clustermap function we grouped similar correlations closer to each other. From this figure we can easily see that these groups of assets have similar highly correlated to each other.
    a. Group-1: [AXP, PFE]
    b. Group-2: [BAC,C, WFC, COF, KEY, PRU],
    c. Group-3: [DUK, KR, REGN, WMT, CMS, XDM, APA],
    d. Group-4: [SCHW, JPM, PNC, GE],
    e. Group-5: [GS, MS]
5. Now one best asset selection from each group is required so that our portfolio can be diversified much easier and in an optimal manner.

Step 6: Upper-confidence bound (UCB) algorithm

Initialize the parameters:

NK = Number of arms (actions)

EPSILON = Epsilon value for exploration

ALPHA = Learning rate

NEPISODES = Number of episodes

HOLD = Number of days to hold an action

TMAX = Total number of time steps

UCB_weight = Weight for UCB adjustment

Choosing a random action with probability of initialized epsilon and selecting the optimal action given information about the past:

If the random generated number is less than or equal to the initialized epsilon greedy parameter, choose randomly across actions with equal probability.

If not, choose the action with the highest q-value using formula below:

$$A_t = \arg\max_a Q_t(a)$$

Updating the expected reward from choosing the given action based on formula below:

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

Loop the optimal selection and reward updating part over initialized time steps and episodes.

During the loop, we prioritize the exploration of actions that we are more uncertain about, we follow the next optimal choice using formula below by setting an upper-confidence bound c.

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

Step 8: Epsilon-greedy algorithm

Initialize the parameters:

NK = Number of arms (actions)

EPSILON = Epsilon value for exploration

ALPHA = Learning rate

NEPISODES = Number of episodes

HOLD = Number of days to hold an action

TMAX = Total number of time steps

Choosing a random action with probability of initialized epsilon and selecting the optimal action given information about the past:

If the random generated number is less than or equal to the initialized epsilon greedy parameter, choose randomly across actions with equal probability.

If not, choose the action with the highest q-value using formula below:

$$A_t = \arg\max_a Q_t(a)$$

Updating the expected reward from choosing the given action based on formula below:

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

Loop the optimal selection and reward updating part over initialized time steps and episodes.

Step 9: Results Analysis and Comparison

Results of the UCB method:

From the results, the strategy basically performs similar to an equal-weighted portfolio. However, it fails to track the most significant increases in returns across the sample period.

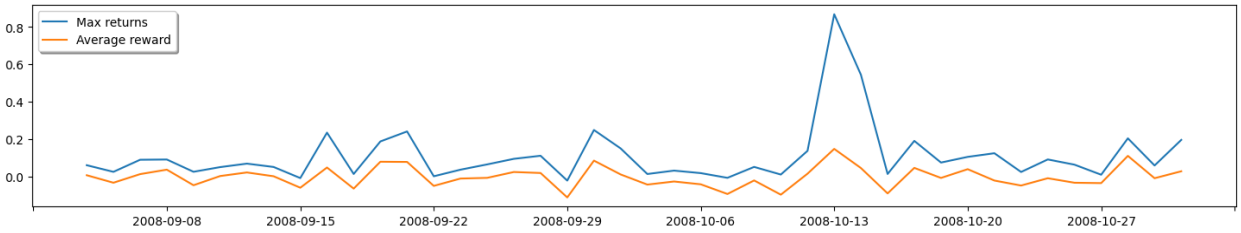

Figure 8: Average reward vs Max returns (UCB method)

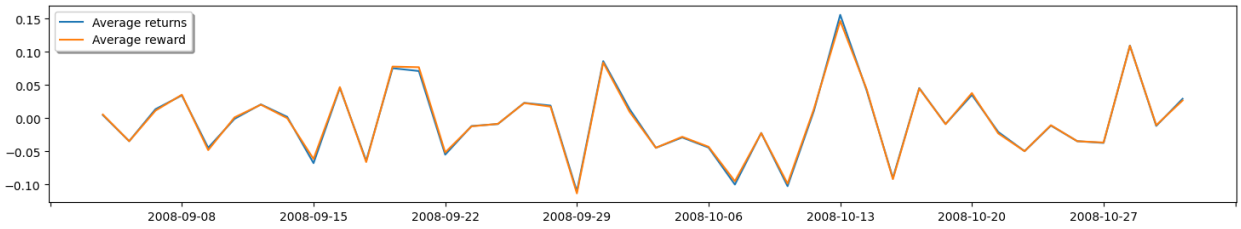

Figure 9: Average reward vs Average returns (UCB method)

We evaluate the performance of the strategy mainly using two metrics: the average frequency of optimal action selection and the average annualized return. The average frequency of optimal action selection assesses the proportion of times the algorithm selects the arm with the highest expected reward while the average annualized return measures the overall performance of the algorithms in terms of rewards generated over the investment horizon.

The average frequency of optimal action using the UCB method is 0.036 while the average annualized return from holding the Bandit portfolio is -0.6138, which is slightly better than the performance of holding the equally-weighted portfolio with the average annualized return equal to -0.6263.

```
Average frequency of optimal action = 0.036
Holding the equally-weighted portfolio, average annualized return = -0.6263; average annualized standard deviation = 0.8868.
Holding the Bandit portfolio, average annualized return = -0.6138; average annualized standard deviation = 0.8756.
```

The graph shows the value that would result from starting with $1 and investing it in either the bandit approach or an equally weighted portfolio over time. Similar returns are produced by the buy-and-hold strategy of the equally weighted portfolio and the "bandit" strategy.

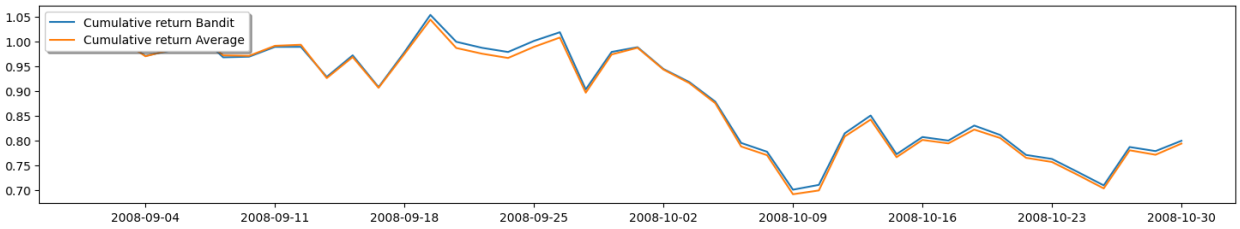

Figure 10: Cumulative return of Bandit portfolio vs equally-weighted portfolio (UCB method)

Results of Epsilon-greedy method:

From the results, the strategy basically performs similar to an equal-weighted portfolio. However, it fails to track the most significant increases in returns across the sample period.

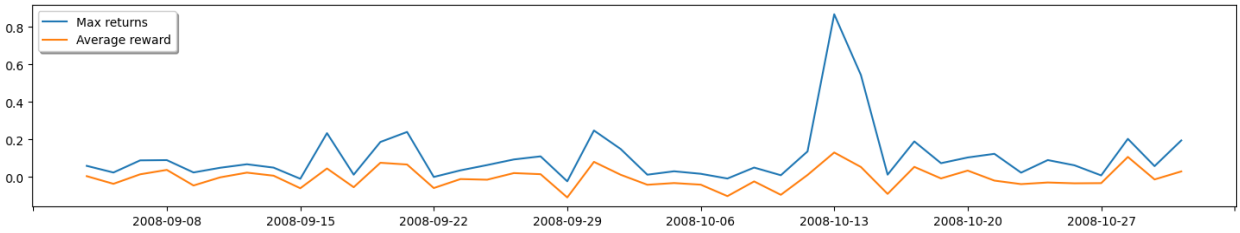

Figure 11: Average reward vs Max returns (Epsilon-greedy method)

Figure 12: Average reward vs Average returns (Epsilon-greedy method)

We evaluate the performance of the strategy mainly using two metrics: the average frequency of optimal action selection and the average annualized return. The average frequency of optimal action selection assesses the proportion of times the algorithm selects the arm with the highest expected reward while the average annualized return measures the overall performance of the algorithms in terms of rewards generated over the investment horizon.

The average frequency of optimal action using Epsilon-greedy method is 0.039 while the average annualized return from holding the Bandit portfolio is -0.6491, which is slightly worse than the performance of holding the equally-weighted portfolio with the average annualized return equal to -0.6263.

```
Average frequency of optimal action = 0.039
Holding the equally-weighted portfolio, average annualized return = -0.6263;average annualized standard deviation = 0.8868.
Holding the Bandit portfolio, average annualized return = -0.6491; average annualized standard deviation = 0.8489.
```

The figure displays the value over time of initiating with $1 and investing it either in the bandit strategy or the equally weighted portfolio. The 'bandit' strategy generates similar returns to the buy-and-hold strategy of the equally weighted portfolio.

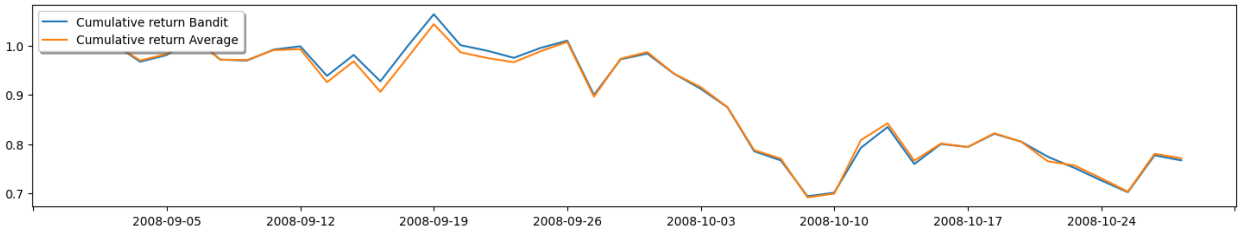

Figure 13: Cumulative return of Bandit portfolio vs equally-weighted portfolio (Epsilon-greedy method)

Comparison between results of the UCB method and results of Epsilon-greedy method:

From the results, both methods fail to track the most significant increases in returns across the sample period.

The Epsilon-greedy method shows a slightly higher average frequency of optimal action compared with the UCB method. However, it generates a lower average annualized return.

Both methods generate similar cumulative returns compared with the buy-and hold strategy of the equally weighted portfolio.

Comparison between group's result with results of the Huo paper:

The RA-UCB method is an expansion of the standard UCB algorithm that addresses the portfolio selection problem with risk management. The regular UCB algorithm balances the exploration of novel arms with the exploitation of the best-performing arms in order to maximize the cumulative reward. To do this, it calculates the UCB for each arm, chooses the one with the highest UCB, and then updates the empirical mean and variance for the chosen arm. Risk management was not specifically taken into account by our algorithm.

The RA-UCB algorithm, on the other hand, tries to maximize cumulative reward while simultaneously limiting risk. To do this, the UCB for each asset is calculated, the asset with the highest UCB is chosen, the empirical mean and variance are updated for the chosen asset, the portfolio weights are calculated for each asset, and the portfolio is then rebalanced to the target weights. The risk parameter lambda regulates the trade-off between reward and risk; a larger value of lambda indicates a choice for greater risk, while a lower value indicates a preference for reduced risk.

Step 10: Import and Structure Data for 2020

The sample data will be based on the daily returns of 27 selected assets in 2020. The sample period would cover the time when COVID began. The following are the asset lists for 13 financial institutions: **JPM, WFC, BAC, C, GS, USB, MS, KEY, PNC, COF, AXP, PRU, SCHW**. As STU and BBT merged in 2019, downloading the data of the two assets BBT and STI from Yahoo Finance was not possible. Thus we skipped these two.

14 non-financial institutions: **KR, PFE, XOM, WMT, DAL, CSCO, HCP, EQIX, DUK, NFLX; APA, F, REGN, CMS**. As HCP was listed in 2021, data of HCP in 2020 was not available. Thus we skipped the HCP. Disregarding these 3 assets, we have a total of 27 assets thus continue our calculations with 27 assets.

We combined the data with steps similar to Step 3.

That outputs the prices and log returns as time series data frames called Prices1 and Returns1.Here are some lines from Price1 dataframe:

Prices1

	JPM	WFC	BAC	C	GS	USD	MS	KEY	PNC	COF	...	DAL	CSCO	EQIX	DUK	NFLX	GE	APA	F	REGN	CMS
2020-01-02	125.94	48.73	32.80	70.53	215.88	15.94	46.45	17.29	140.33	96.77	...	58.51	43.41	546.46	78.50	329.81	73.46	23.97	8.20	373.35	55.40
2020-01-03	124.28	48.43	32.12	69.20	213.36	15.43	45.70	16.99	138.98	95.27	...	57.54	42.70	551.90	78.55	325.90	73.70	24.28	8.02	369.16	55.94
2020-01-06	124.18	48.14	32.08	68.98	215.54	15.19	45.54	16.73	138.70	94.41	...	57.14	42.85	551.30	78.94	335.83	74.75	24.23	7.98	371.74	56.19
2020-01-07	122.07	47.74	31.86	68.38	216.96	15.55	45.45	16.61	137.60	93.47	...	57.09	42.57	548.85	78.44	330.75	74.19	30.72	8.05	373.13	55.90
2020-01-08	123.02	47.88	32.19	68.90	219.05	15.54	46.03	16.77	139.03	94.46	...	58.32	42.60	550.66	78.55	339.26	73.52	30.93	8.05	382.42	55.89
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2020-12-23	115.70	28.62	28.40	54.90	241.96	24.63	62.71	14.70	133.70	91.64	...	40.16	40.85	673.28	80.75	514.48	67.21	14.18	7.96	492.19	53.11
2020-12-24	115.19	28.14	28.31	54.71	241.69	24.90	62.60	14.57	133.20	92.10	...	39.65	41.00	676.02	81.33	513.97	65.91	13.84	7.85	488.93	53.61
2020-12-28	115.95	28.22	28.47	55.22	244.92	24.83	62.57	14.62	133.56	92.33	...	40.07	41.35	679.79	81.85	519.12	65.85	13.63	7.87	482.96	54.22
2020-12-29	115.64	28.08	28.36	55.02	243.43	24.81	62.25	14.37	132.04	91.04	...	39.95	41.09	678.74	81.50	530.87	65.35	13.57	7.81	484.22	54.53
2020-12-30	115.97	28.05	28.33	54.97	244.79	25.55	62.37	14.62	133.88	92.64	...	40.47	40.95	670.72	82.13	524.59	66.28	13.98	7.85	478.30	55.23

252 rows × 27 columns

Here are some lines from Returns1 data frame which consists of column-wise log returns series for each asset:

```
np.round(Returns1,4)
```

	JPM	WFC	BAC	C	GS	USD	MS	KEY	PNC	COF	...	DAL	CSCO	EQIX	DUK	NFLX	GE	APA	F	REGN	CMS
2020-01-03	-0.0133	-0.0062	-0.0209	-0.0190	-0.0117	-0.0325	-0.0163	-0.0175	-0.0097	-0.0156	...	-0.0167	-0.0165	0.0099	0.0006	-0.0119	0.0033	0.0128	-0.0222	-0.0113	0.0097
2020-01-06	-0.0008	-0.0060	-0.0012	-0.0032	0.0102	-0.0157	-0.0035	-0.0154	-0.0020	-0.0091	...	-0.0070	0.0035	-0.0011	0.0050	0.0300	0.0141	-0.0021	-0.0050	0.0070	0.0045
2020-01-07	-0.0171	-0.0083	-0.0069	-0.0087	0.0066	0.0234	-0.0020	-0.0072	-0.0080	-0.0100	...	-0.0009	-0.0066	-0.0045	-0.0064	-0.0152	-0.0075	0.2373	0.0087	0.0037	-0.0052
2020-01-08	0.0078	0.0029	0.0103	0.0076	0.0096	-0.0006	0.0127	0.0096	0.0103	0.0105	...	0.0213	0.0007	0.0033	0.0014	0.0254	-0.0091	0.0068	0.0000	0.0246	-0.0002
2020-01-09	0.0037	-0.0017	0.0016	0.0091	0.0202	0.0115	0.0095	0.0018	0.0120	0.0135	...	0.0019	-0.0042	0.0088	0.0025	-0.0107	-0.0026	-0.0039	0.0012	0.0027	0.0077
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2020-12-23	0.0276	0.0469	0.0282	0.0253	0.0251	-0.0169	0.0232	0.0374	0.0204	0.0191	...	0.0329	-0.0071	-0.0146	-0.0056	-0.0247	0.0233	0.0454	0.0229	0.0043	0.0002
2020-12-24	-0.0044	-0.0169	-0.0032	-0.0035	-0.0011	0.0109	-0.0018	-0.0089	-0.0037	0.0050	...	-0.0128	0.0037	0.0041	0.0072	-0.0010	-0.0195	-0.0243	-0.0139	-0.0066	0.0094
2020-12-28	0.0066	0.0028	0.0056	0.0093	0.0133	-0.0028	-0.0005	0.0034	0.0027	0.0025	...	0.0105	0.0085	0.0056	0.0064	0.0100	-0.0009	-0.0153	0.0025	-0.0123	0.0113
2020-12-29	-0.0027	-0.0050	-0.0039	-0.0036	-0.0061	-0.0008	-0.0051	-0.0172	-0.0114	-0.0141	...	-0.0030	-0.0063	-0.0015	-0.0043	0.0224	-0.0076	-0.0044	-0.0077	0.0026	0.0057
2020-12-30	0.0028	-0.0011	-0.0011	-0.0009	0.0056	0.0294	0.0019	0.0172	0.0138	0.0174	...	0.0129	-0.0034	-0.0119	0.0077	-0.0119	0.0141	0.0298	0.0051	-0.0123	0.0128

251 rows × 27 columns

And we visualize the cumulative performance of each stock during the sample period.

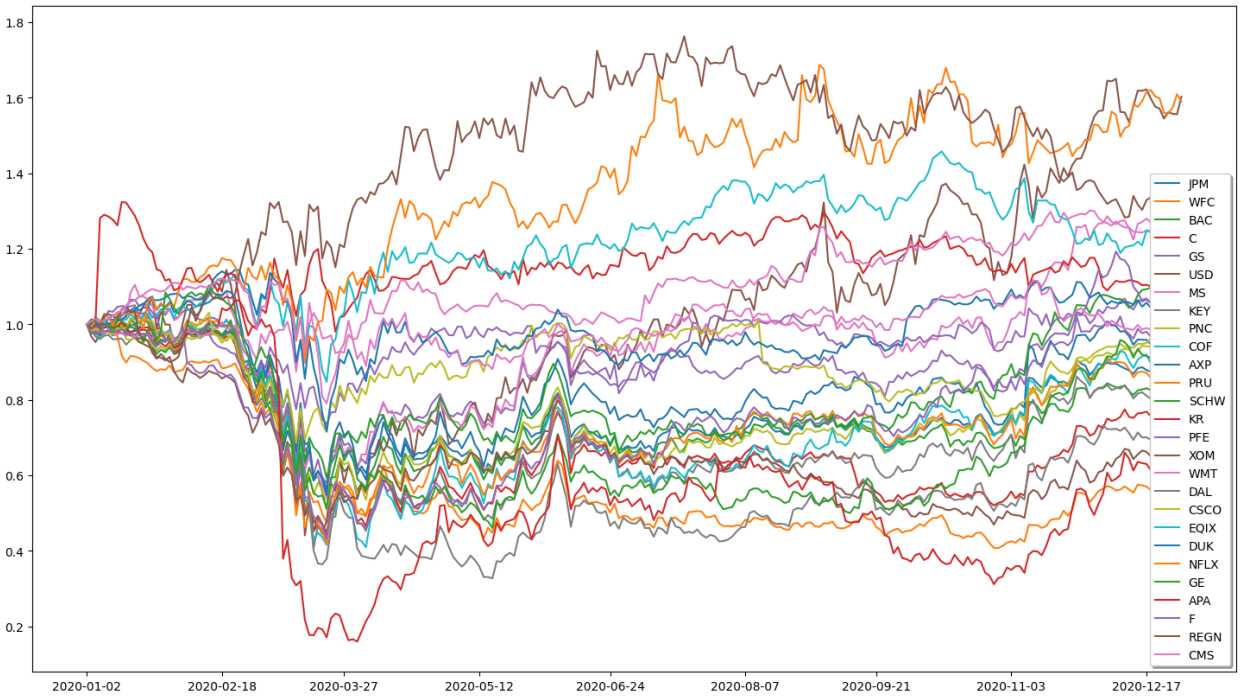

Figure 14: Price time series for 27 assets (2020 data)

Step 11: Rerun algorithm for new data (2020)

We rerun the same code in Step 6 and Step 8 for new data (2020). In order to allow more exploration, we adjust the parameters using higher epsilon and higher upper confidence bound. We also changed the holding period from 1 to 2.

Most of the results show a similar situation to the results using the data in 2008. For example, both methods fail to track the most significant increases in returns across the sample period and the Epsilon-greedy method shows a slightly higher average frequency of optimal action compared with the UCB method (0.0407 vs 0.0369).

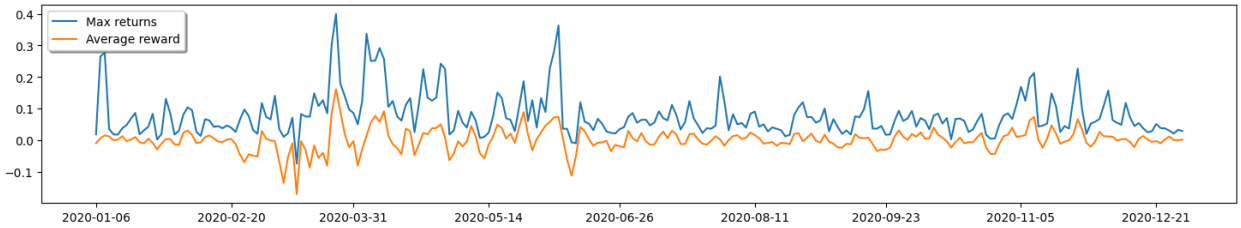

Figure 15: Average reward vs Max returns (UCB method)

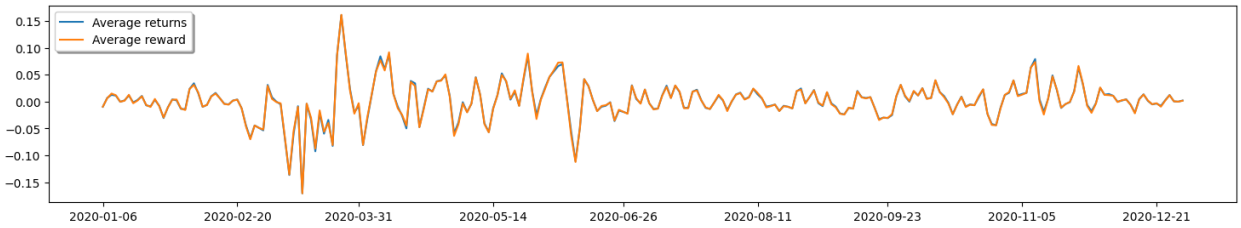

Figure 16: Average reward vs Average returns (UCB method)

```
Average frequency of optimal action = 0.0369
Holding the equally-weighted portfolio, average annualized return = 0.1515; average annualized standard deviation = 0.3865.
Holding the Bandit portfolio, average annualized return = 0.125; average annualized standard deviation = 0.3871.
```

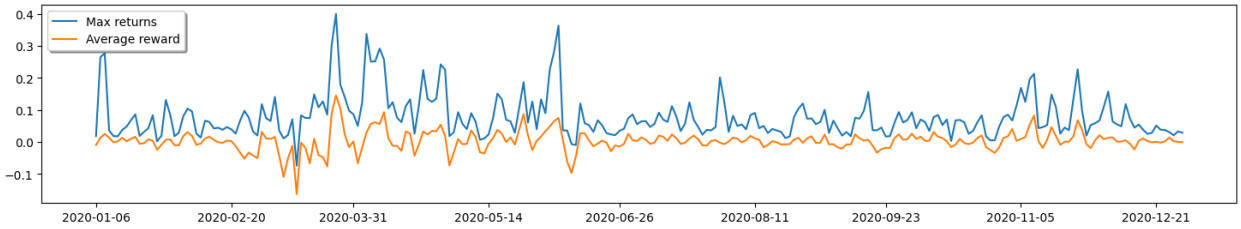

Figure 17: Average reward vs Max returns (Epsilon-greedy method)

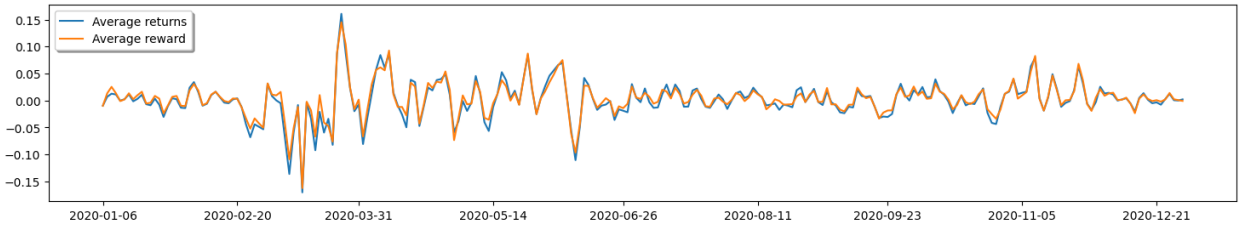

Figure 18: Average reward vs Average returns (Epsilon-greedy method)

```
Average frequency of optimal action = 0.0407
Holding the equally-weighted portfolio, average annualized return = 0.1515; average annualized standard deviation = 0.3865.
Holding the Bandit portfolio, average annualized return = 0.5006; average annualized standard deviation = 0.3489.
```

The most significant difference is shown in the simulation of investment initiating with $1 in the portfolio. In the analysis using 2008 data, the 'bandit' strategy using both methods generates similar returns to the buy-and-hold strategy of the equally weighted portfolio. Nevertheless, when we rerun our analysis using more recent data (2020 data), the cumulative returns show a larger deviation. The 'bandit' strategy using the UCB method performs slightly poorer then the buy-and-hold strategy of the equally weighted portfolio while that using Epsilon-greedy method performs much better than the buy-and-hold strategy of the equally weighted portfolio.

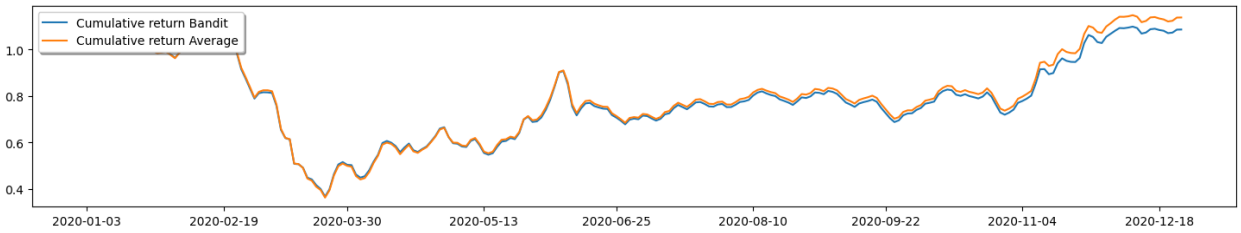

Figure 19: Cumulative return of Bandit portfolio vs equally-weighted portfolio (UCB method)

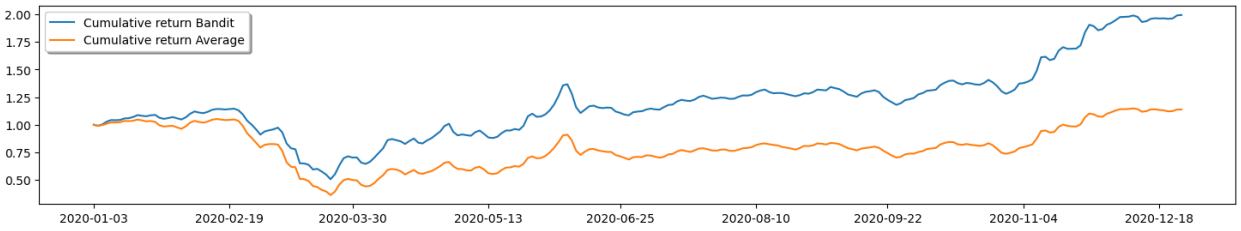

Figure 20: Cumulative return of Bandit portfolio vs equally-weighted portfolio (Epsilon-greedy method)

Although the use of recent data may indeed provide valuable insights and potentially enhance the performance of the algorithms employed, we cannot solely attribute the observed differences in results to the utilization of more recent data. While rerunning the algorithm, it is essential to acknowledge that other factors may have contributed to the variations in outcomes. For instance, modifying certain parameters as well as altering the length of the evaluation period can introduce additional elements of change.

References

1. Stochastic Modeling. WorldQuant University, 2023.
2. Huo. “Risk-aware multi-armed bandit problem with application to portfolio selection.” 15 Nov. 2017, https://royalsocietypublishing.org/doi/full/10.1098/rsos.171377

—----------