alamSYS: Development of Stock Market Price Forecasting System Using Dynamic Mode Decomposition, Long Short-Term Memory with Arnaud Legoux Moving Average Convergence-Divergence Integration

A Special Problem
Presented to
the Faculty of the Division of Physical Sciences and Mathematics
College of Arts and Sciences
University of the Philippines Visayas
Miag-ao, Iloilo

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science in Computer Science by

OLARTE, John Markton M.

Nilo C. Araneta
Adviser

June 2023

Contents

List of Figures

List of Tables

Chapter 1

Results and Discussions

This chapter presents results and discussions from this special problem. Its goal is to provide a comprehensive analysis and interpretation of the data collected for alamSYS's internal and external components. As a result, this chapter is divided into the following sections:

(a) Documentation for alamSYS

(b) DMD-LSTM Results and Discussions

(c) ALMACD Results and Discussions

(d) alamSYS System Tests Results and Discussions

(e) Results and Discussions for the Real World Application of alamSYS

1.1 alamSYS Documentation

The goal of this section is to thoroughly document the current state of the alamSYS in order to facilitate meaningful discussions.

1.1.1 Documentation for alamAPI and Database

xxx

1.1.2 Documentation for alamSYS Preprocessor

xxx

1.1.3 Documentation for alamAPP

xxx

1.1.4 Build and Deployment Guide

xxx

1.2 alamSYS System Tests Results and Discussions

With the development of alamSYS, we must ensure that all of its components are functioning properly. This section focuses on the system's performance while idle and under load, as well as the API's and database's ability to handle multiple requests at a time.

The Table 1.1 shows the CPU and memory utilization of each of the alamSYS components whenever it is not processing any information. Wherein, the data presented below is gathered by logging the system utilization of alamSYS within an hour.

Table 1.1: Idle System Average Resource Usage Statistics

	alamAPI	**alamDB**	**alamPREPROCESSOR**
CPU Utilization (%)	0.168125	0.254313	0.009769
Memory Utilization (MiB)	45.718311	166.775377	312.798300

From the table above, it is shown that the alamSYS as a whole only utilizes 0.432207% of the total CPU power on average when it is on idle. Wherein, the bulk of the CPU power is being used by the database at 0.254313% which is 58.84% of the total average CPU utilization of the alamSYS.

This result actually shows promising idle performance, as an Ubuntu system's normal idle CPU utilization is less than 10%. However, it should be noted that the alamSYS is not completely idle because background tasks such as scheduling, mongoDB processes, and the API must remain active in order to respond to any API queries.

Furthermore, the lower average CPU utilization for alamAPI and alamPREPROCESSOR could be attributed to the linux distribution used as their base image, as previously discussed on 'Docker-Compose Layer Diagram' of Section **??**.

The CPU utilization for the alamPREPROCESSOR, in particular, was lower than we would have expected given that it is running a schedule checker every second. This indicate that the schedule library utilizes an efficient way to check for schedules. However, it may not be the case for its memory utilization, which is discussed further below.

Moving on, the alamSYS's memory utilization shows that it uses 525.291988 Mebibytes (MiB) or 550.808572 Megabytes (MB) on average. The memory utilization of alamPREPROCESSOR accounts for the majority (59.55%). This demon-

strates that, despite having the lowest CPU utilization, the alamPREPROCESSOR consumes more memory than the combination of the alamAPI and alamDB.

Again, as previously discussed, the alamPREPROCESSOR's high average memory utilization is due to the background schedule checking, which in this case is programmed to use more memory than CPU power.

This average utilization result implies that the alamSYS may be able to be deployed on devices with lower specifications. A Raspberry Pi 4, which has a quad core CPU and at least 2GB RAM, is one example (Zwetsloot, 2019). However, idle performance only shows the minimum CPU and memory utilization of the alamSYS components and does not provide a complete picture of its utilization, particularly under load. As a result, we must investigate the system's CPU and memory utilization while under load.

The internal load averages system utilization of the alamSYS' preprocessor is shown in the following table.

Table 1.2: Internal Load Average Resource Usage Statistics

	Data Collector	**Data Processor**	**alamSYS PREPROCESSOR (Data Collector & Data Processor)**
Failure Rate (%)	0	0	0
Success Rate (%)	100	100	100
Average Runtime (s)	41.72398	8.38061	48.30466
Average CPU Utilization (%)	11.40659	92.71117	20.03138

Table 1.2 continued from previous page

	Data Collector	Data Processor	alamSYS PREPROCESSOR (Data Collector & Data Processor)
Average Memory Utilization (MiB)	3.64200	57.09545	794.29436
Average Network Utilization (Mb)	232.73640	154	77.27655

Before exploring into the results shown in Table 1.2, it is critical to first establish a context for how the data was gathered. The data in the above table was collected while the alamSYS, specifically the alamPREPROCESSOR, was subjected to a stress test load of 100 consecutive data collection and processing. Also, the data of average utilization as indicated for each column are independently gathered from each other.

Based on the table above, each component was capable of processing 100 consecutive processes without failure. This means that the alamSYS, specifically the alamPREPROCESSOR, can run at least 100 times in a row without ailing. Also, keep in mind that the alamPREPROCESSOR only collects and processes data once per day, and the developer has included an option for system users or maintainers to manually rerun these processes in cases where the alamPREPROCESSOR fails - for example, due to a lost or slow internet connection, a power outage, and so on.

Meanwhile, the data processor's average runtime is faster than the data collector's, and it runs on an average of 48.30466 seconds. This was already expected because the data collector needed to connect to the internet and was thus constrained by internet speed. Whereas the average speed during the course of this test was 52.98Mbps, this could imply that the data collector's runtime may be slower or faster depending on the internet speed.

Furthermore, the data processor's average runtime was surprisingly fast given that it needed to apply DMD-LSTM to a total of 20 stocks and calculate the position using the ALMACD given a total of 205 data points. Furthermore, looking at its CPU and memory utilization, it can be seen that it uses more than the data collector, with approximately 87.68% more average CPU power used and approximately 93.63% more average memory used.

In line with the CPU and memory utilization, the average CPU utilization of the alamPREPROCESSOR on load is 99.95% higher than when it is idle. And it uses 60.62% more memory on average.

Lastly, looking at the network utilization of the alamPREPROCESSOR, we can see that the data collector used the most network bandwidth, as expected. However, it may be surprising to see that the data processor uses the network when it should not because it does not need to process or collect data from the internet, but this is still within expectations because network bandwidth utilization also accounts for local network utilization. The data processor of alamPREPROCESSOR, in particular, uses the local network to connect to and update the new data in the alamDB.

The following tables show the system's deployment load results. Specifically, to access the buy and sell collections, as shown in Tables 1.3 and 1.4. Additionally, to have a background on the test that results in the following outcomes. It should be noted that the test consisted of three instances of the same tester application running on ten different computers. Where each application requests 10, 100, 1000 buy and sell data from the alamDB via the alamAPI, which is tunneled over the internet for server access outside the university's local network using LocalXpose services.

Table 1.3: Deployment Load Test Results (Buy Requests)

	Number or Requests		
	10	**100**	**1000**
Success Rate (%)	100	100	100
Average Processing Time (s)	11.905222	139.618550	1159.773569

Table 1.4: Deployment Load Test Results (Sell Requests)

	Number or Requests		
	10	**100**	**1000**
Success Rate (%)	100	100	100
Average Processing Time (s)	13.384126	130.119867	1642.995011

As per the tables above, the alamSYS was able to handle all consecutive and simultaneous requests from all external devices. It was able to handle 71,000 requests in approximately one hour and 20 minutes.

Furthermore, the data shows that a request is processed in 1.34 seconds on average. Wherein, the fastest processing time from the data collected was 0.93 seconds. This result is within the expected response time of APIs, where good is defined as 0.1 to one second, and any time between one and two seconds is acceptable, as long as it does not exceed two seconds, which users may perceive as an interruption in the process (Juviler, 2022).

It is also worth noting that in a separate internal test for the alamSYS, the alamAPI only takes on average 0.0094 seconds to send its response. And that the additional delay in response time on deployment is caused by network-related

factors, such as the time it takes the tunneling service to accept the query and send back the system's response. In fact the network related delay contributes 99.30% of the average deployment response time. Hence, it might be useful to deploy the system in a network that further minimizes this additional delay in the response time.

Meanwhile Table 1.5 shows the average CPU and memory utilization of alamAPI and alamDB as the load testing as stated above, was being processed.

Table 1.5: CPU and Memory Utilization Statistics of alamAPI and alamDB Under Deployment Load Testing

	alamAPI	**alamDB**
CPU Utilization (%)	17.725949	1.070133
Memory Utilization (MiB)	44.620837	129.273305

To further visualize the system utilization over time, Figure 1.1 shows the CPU utilization of alamAPI and alamDB over time. And Figure 1.2 shows the memory utilization of alamAPI and alamDB.

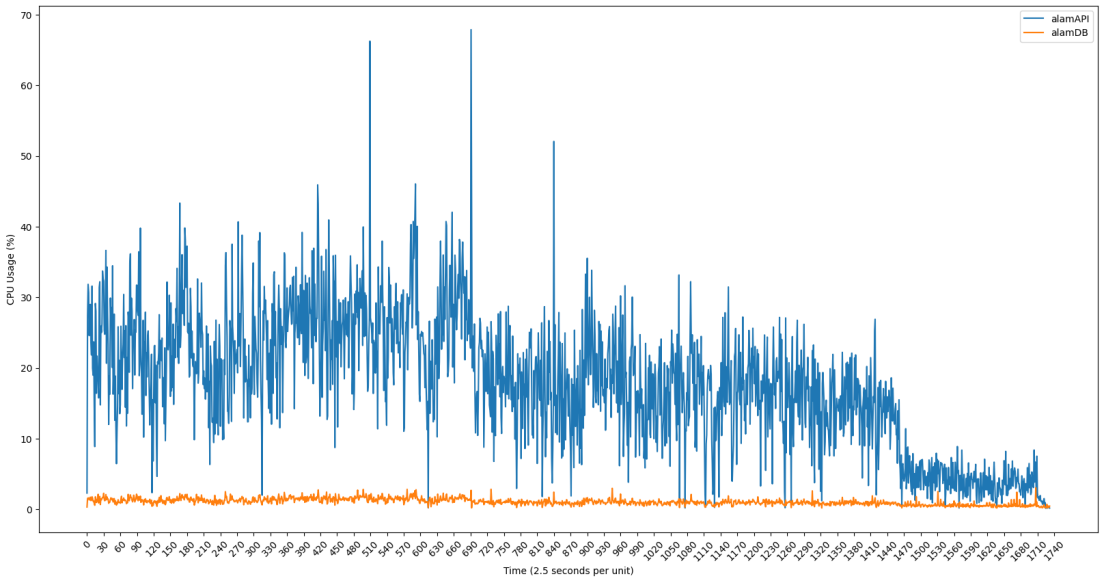

Figure 1.1: Deployment Load CPU Utilization of alamAPI and alamDB Over Time

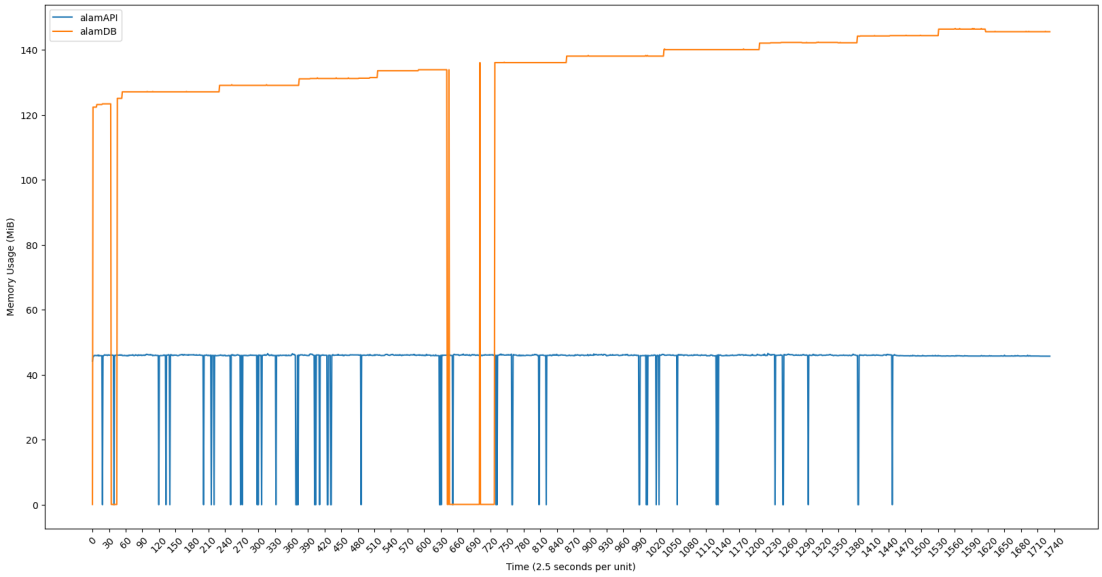

Figure 1.2: Deployment Load Memory Utilization of alamAPI and alamDB Over Time

Based on the table and figures above, processing all requests consumes only 17.73% and 1.07% of the CPU for alamAPI and alamDB, respectively. In terms of memory usage, they are 44.62 MiB and 129.27 MiB, respectively.

Furthermore, at around 1740 time units, the CPU utilization for alamAPI was shown to be reduced. This is due to the fact that most tester applications have already completed processing all of the requests that they are programmed to run, reducing the load on the server by half.

1.3 DMD-LSTM Model Results and Discussions

This section presents and discusses the Deep Learning Model's training, testing, and cross-validation results.

In Table 1.6 the training error metrics are shown for each of the window sizes tested.

Table 1.6: DMD-LSTM Training Error Metrics Scores for Different Window Sizes

Error Metrics	*Window Sizes*			
	5	**10**	**15**	**20**
MSE	0.000037	0.787877	0.006917	0.057851
RMSE	0.006106	0.887624	0.083166	0.240522
MAE	0.004175	0.755407	0.067645	0.202746
MAPE	**0.000001**	0.000194	0.000017	0.000053

Where it is observed that the best performing model based on having the lowest MAPE score is the DMD-LSTM with a window size of 5. Morever, we can see the differences from each MAPE score for each window size in the Figure 1.3 shown below.

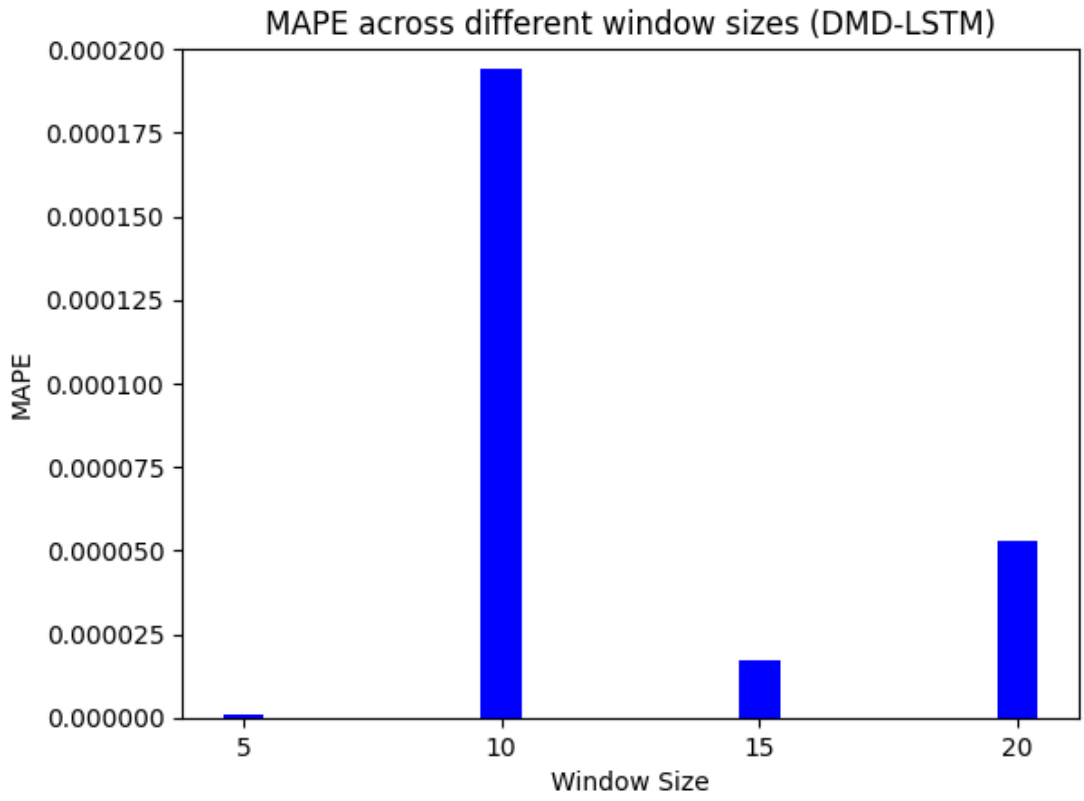

Figure 1.3: Comparison of MAPE Scores for DMD-LSTM Model Training Across Different Window Sizes

The figure above also shows that the MAPE score for window sizes 15 and 20 is higher than the MAPE score for window size 10. MAPE score increases from window size 15 to size 20, indicating that increasing window size may result in a lower performing model.

Furthermore, as previously stated, the window size of 5 results in the best MAPE score being the lowest. Where it outperforms the worst performing model (DMD-LSTM with window size 10) by 0.000193 units. As illustrated clearly in Figure 1.3.

Knowing that the DMD-LSTM model performs as expected based on the training data scores, it is critical that we also examine the training data results from

a baseline LSTM. The baseline LSTM is, as the name implies, a simple LSTM model lacking the DMD component. The table below shows the results of the baseline LSTM training.

Table 1.7: Baseline LSTM Training Error Metrics Scores for Different Window Sizes

Error Metrics	*Window Sizes*			
	5	**10**	**15**	**20**
MSE	2912.840703	191.935882	1118.183283	706.136814
RMSE	53.970739	13.854093	33.439248	26.573235
MAE	35.301888	9.480864	22.099720	18.285352
MAPE	0.009618	**0.002527**	0.006024	0.005004

According to the table above, the baseline LSTM with window size 10 performs the best, with the lowest MAPE score of 0.002527 when compared to the other baseline LSTM models.

However, the DMD-LSTM model with window size 5 outperforms it by 0.002526. As a result, the alamSYS makes use of the DMD-LSTM model, specifically the one with a window size of 5. Where from now on, the DMD-LSTM model refers to the DMD-LSTM model with a window size of 5.

Nonetheless, the DMD-LSTM model's performance is limited to the training dataset from PSEI, and it must be cross-validated using data from other stocks, which includes the PSEI validation dataset. The results of this cross-validation is presented in Table 1.8. It should also be noted that cross-validation uses logarithmic normalization as a data preprocessing technique to make the dataset more normal, which aids in analyzing the model's performance with the given dataset. Normalization techniques, in particular, allow for closer variation within the forecasted data. (S.Gopal Krishna Patro, 2015).

Table 1.8: DMD-LSTM Cross-Validation Error Metrics Scores

Stocks	MSE	RMSE	MAE	MAPE
PSEI	0.00002	0.00419	0.00328	1.510000e-03
AC	0.00236	0.04856	0.03414	6.110000e-03
ALI	0.00255	0.05054	0.03645	1.597000e-02
AP	0.00129	0.03596	0.02515	9.220000e-03
BDO	0.00160	0.03999	0.02799	7.250000e-03
BLOOM	0.01883	0.13721	0.06901	1.052898e+12
FGEN	0.00224	0.04733	0.03265	1.197000e-02
GLO	0.00211	0.04595	0.03149	4.680000e-03
ICT	0.00335	0.05785	0.03731	3.005818e+11
JGS	0.00331	0.05752	0.03992	2.009923e+11
LTG	0.01567	0.12518	0.05858	3.583335e+12
MEG	0.00431	0.06565	0.04422	1.393042e+11
MER	0.00326	0.05708	0.03770	9.170000e-03
MPI	0.00273	0.05230	0.03390	2.497000e-02
PGOLD	0.00149	0.03865	0.02818	7.880000e-03
RLC	0.00338	0.05817	0.03978	6.922000e-02
RRHI	0.00131	0.03618	0.02699	6.390000e-03
SMC	0.00137	0.03702	0.02317	5.690000e-03
TEL	0.00178	0.04214	0.03002	4.240000e-03
URC	0.00297	0.05447	0.03742	1.798000e-02

As shown in the table above, the chosen DMD-LSTM model performs well across all other stocks, demonstrating that the model is not overfitted to the training dataset. This score additionally suggests that the model works with non-training data.

The figures below show a 100-day worth of predicted prices versus actual prices to better visualize the performance of the DMD-LSTM model for each stock.

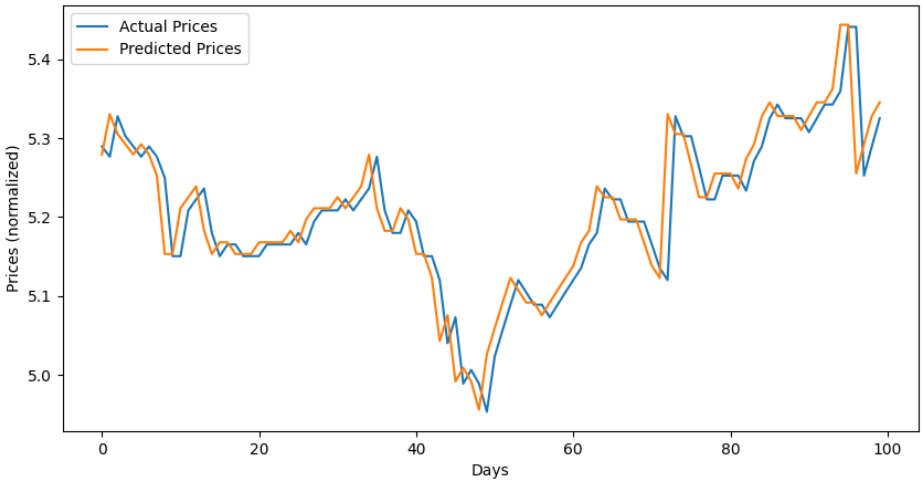

Figure 1.4: Actual vs Predicted Prices on AC for 100 days

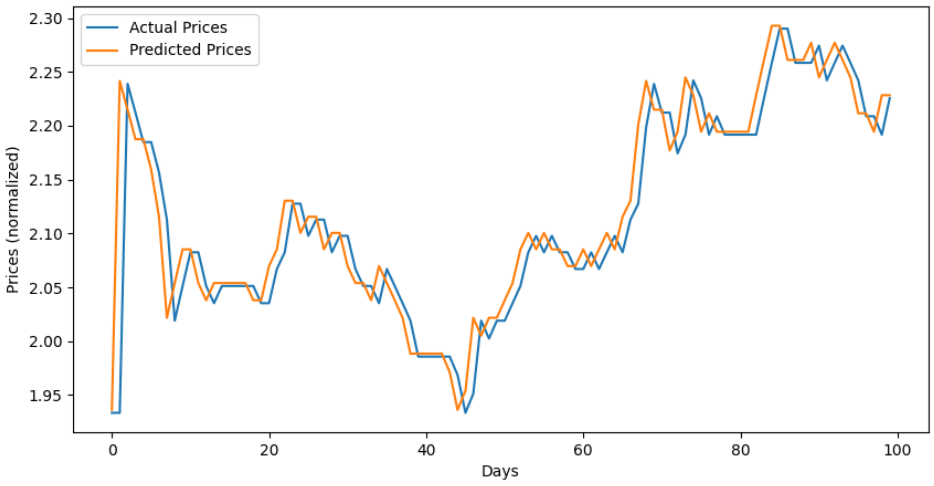

Figure 1.5: Actual vs Predicted Prices for ALI over 100 days

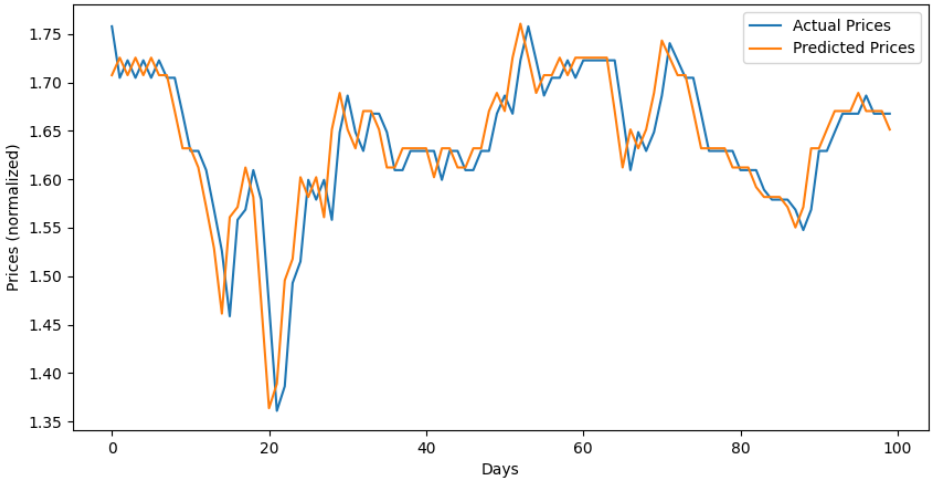

Figure 1.6: Actual vs Predicted Prices for AP over 100 days

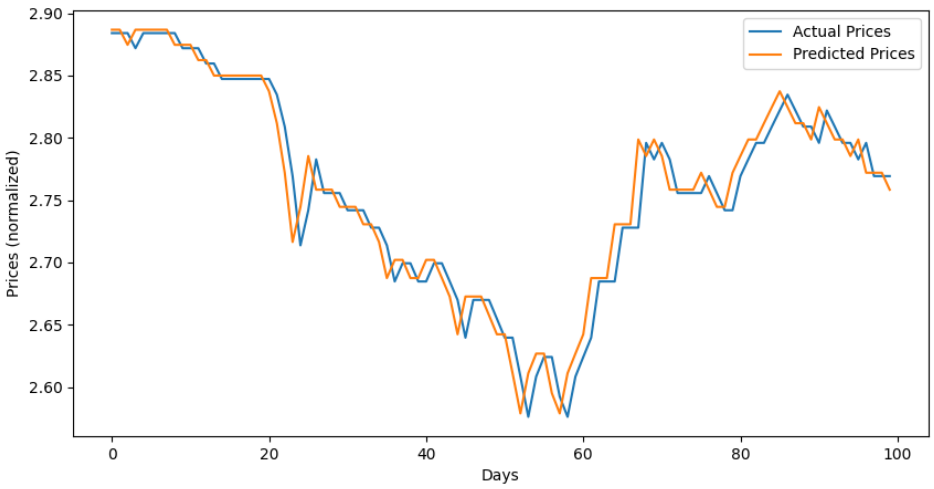

Figure 1.7: Actual vs Predicted Prices for BDO over 100 days

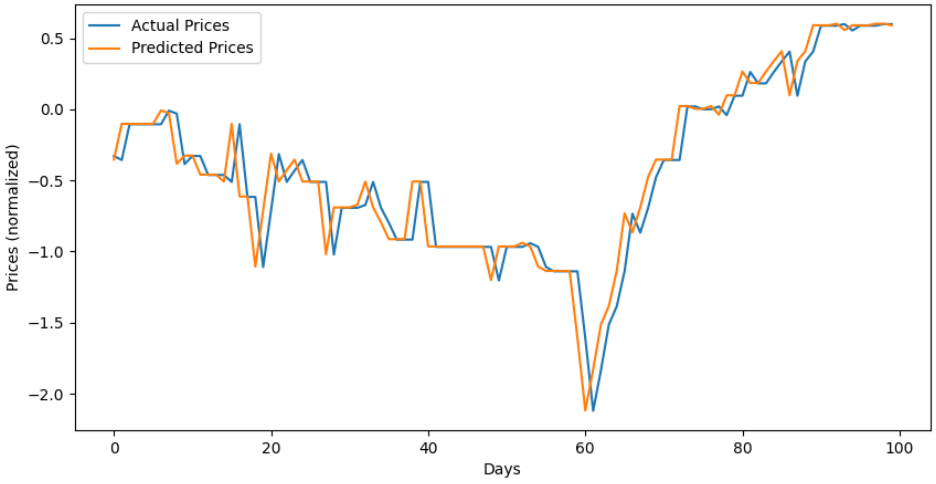

Figure 1.8: Actual vs Predicted Prices for BLOOM over 100 days

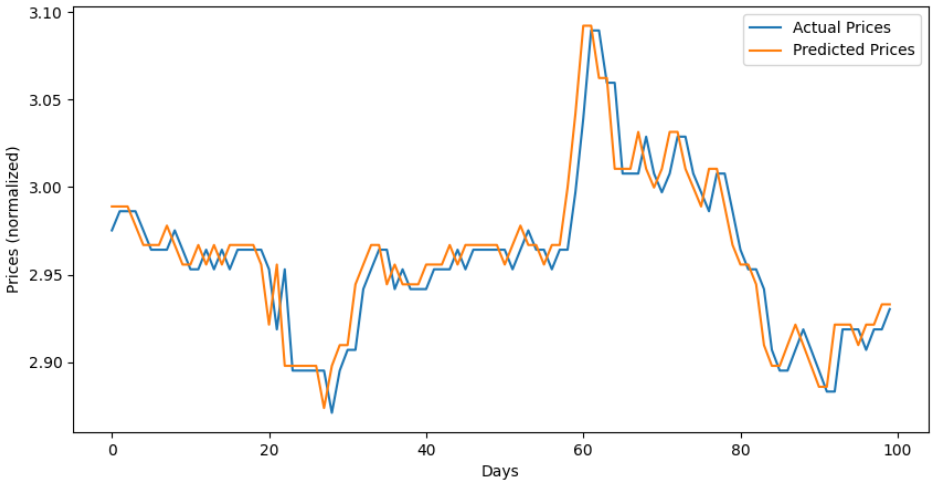

Figure 1.9: Actual vs Predicted Prices for FGEN over 100 days

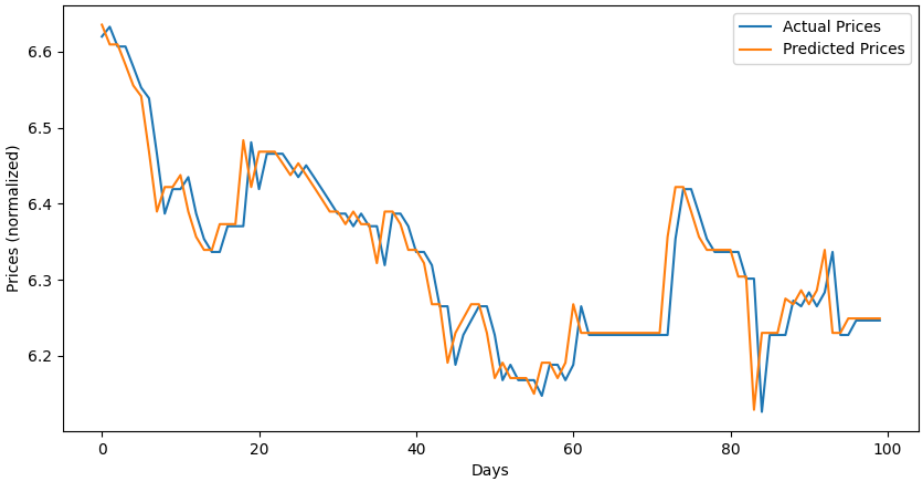

Figure 1.10: Actual vs Predicted Prices for GLO over 100 days

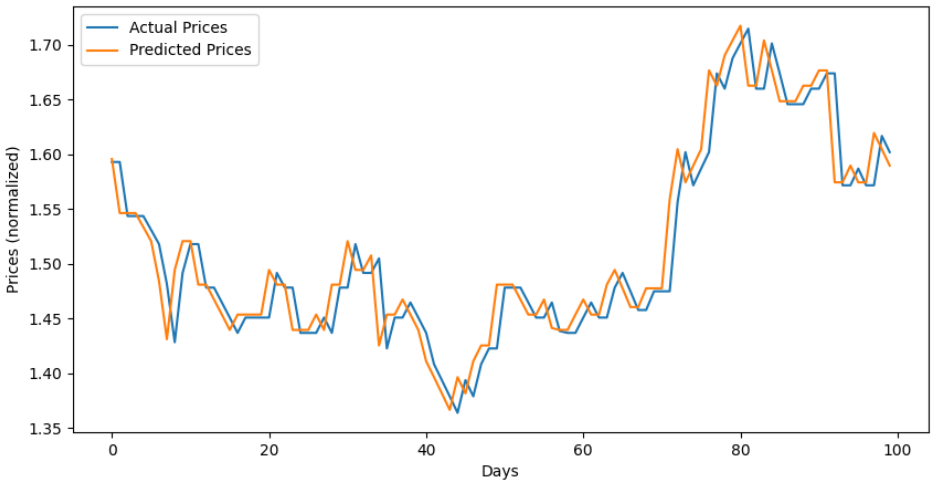

Figure 1.11: Actual vs Predicted Prices for ICT over 100 days

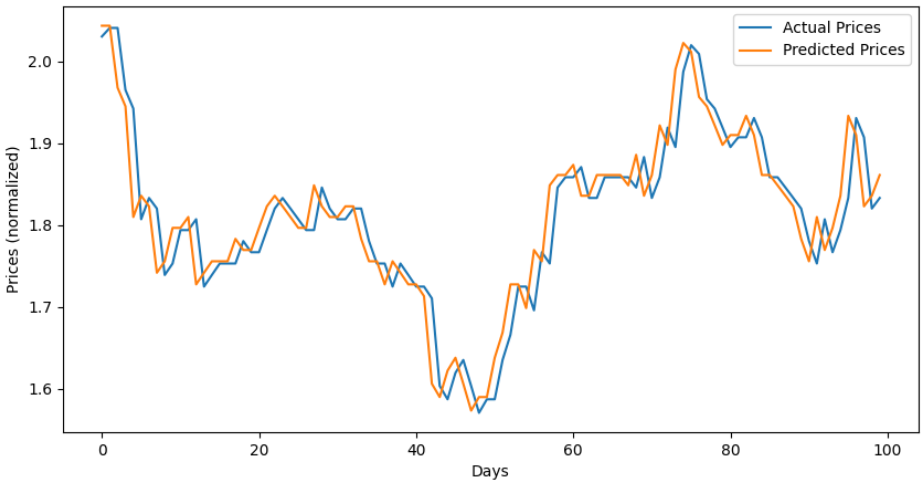

Figure 1.12: Actual vs Predicted Prices on JGS for 100 days

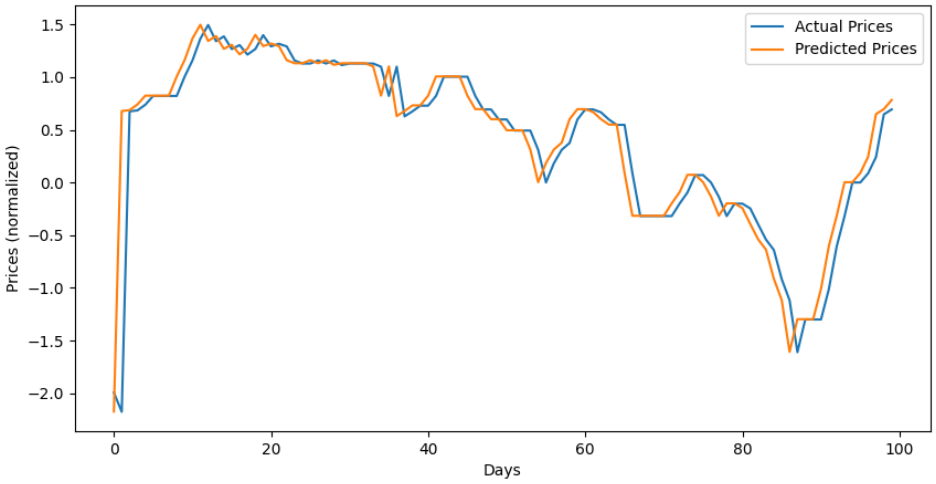

Figure 1.13: Actual vs Predicted Prices on LTG for 100 days

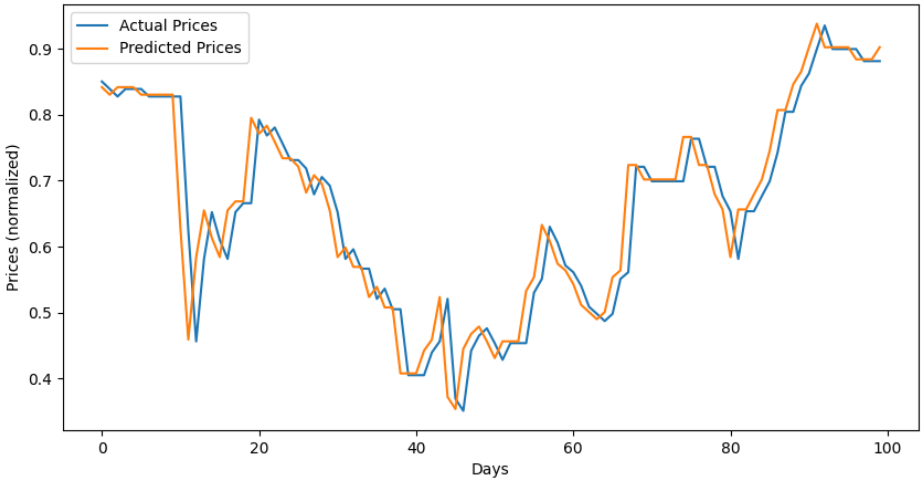

Figure 1.14: Actual vs Predicted Prices on MEG for 100 days

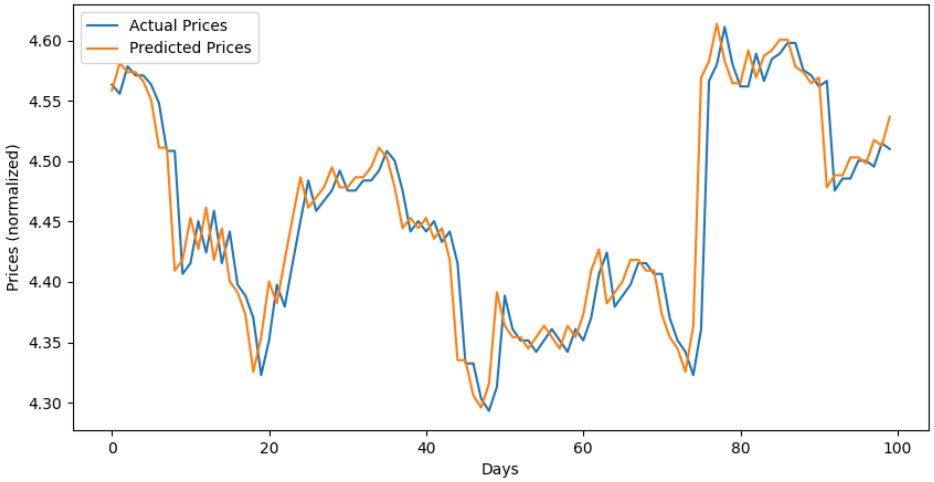

Figure 1.15: Actual vs Predicted Prices on MER for 100 days

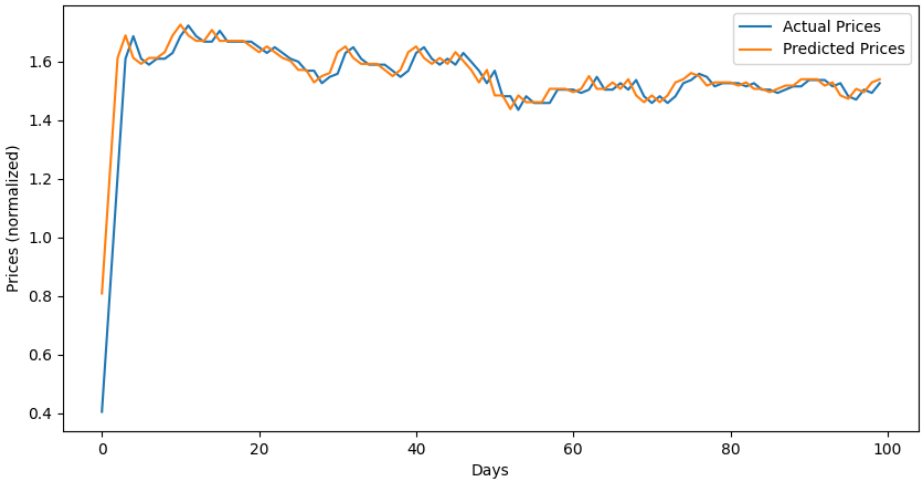

Figure 1.16: Actual vs Predicted Prices on MPI for 100 days

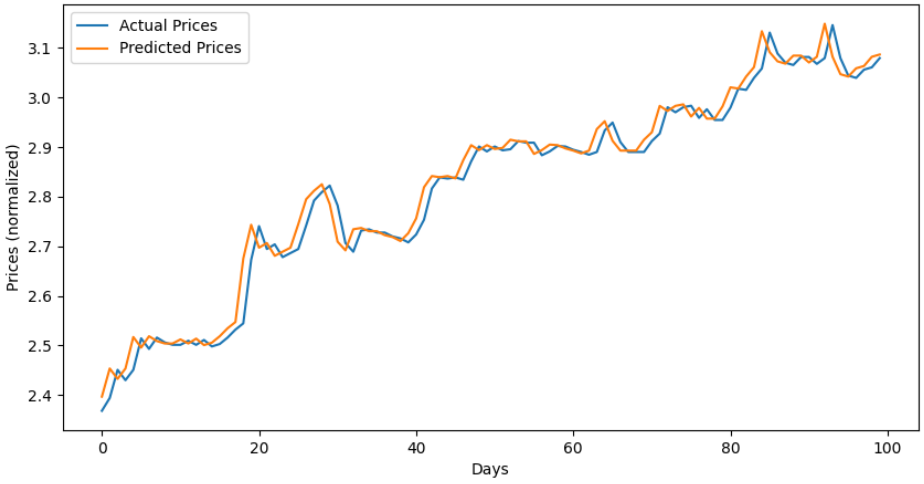

Figure 1.17: Actual vs Predicted Prices on PGOLD for 100 days

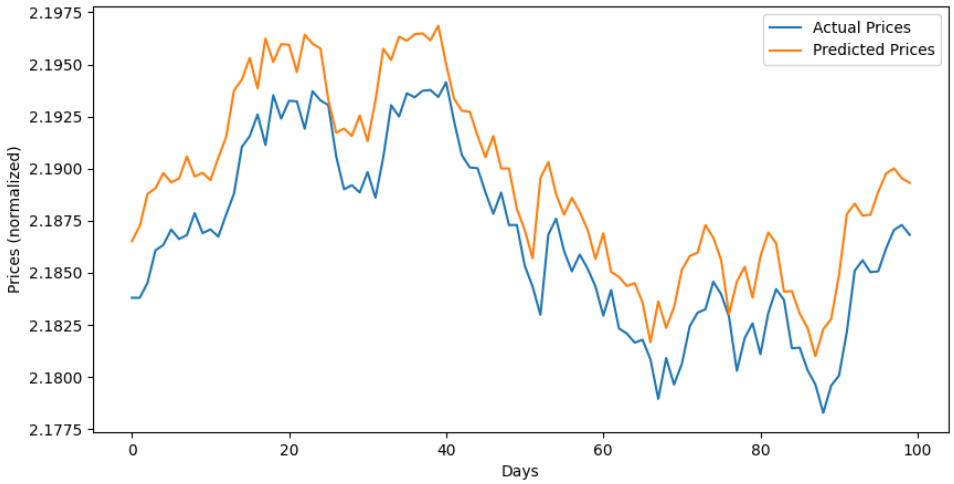

Figure 1.18: Actual vs Predicted Prices on PSEI for 100 days

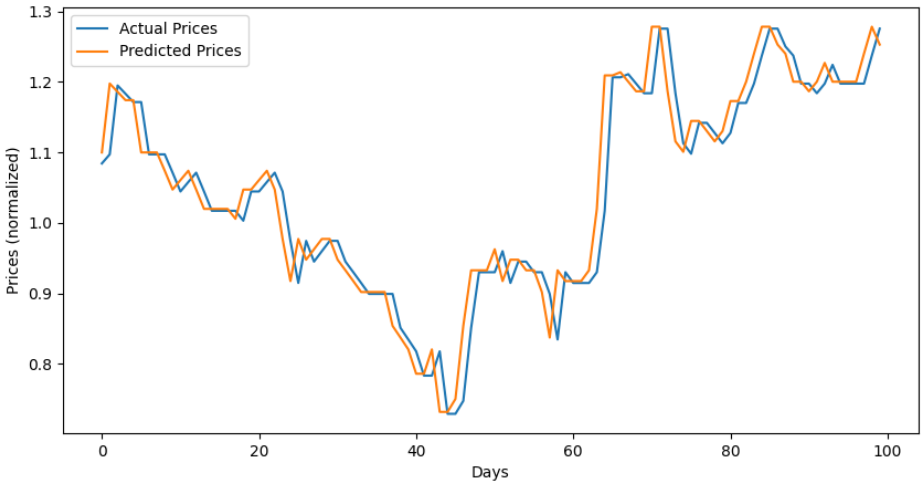

Figure 1.19: Actual vs Predicted Prices on RLC for 100 days

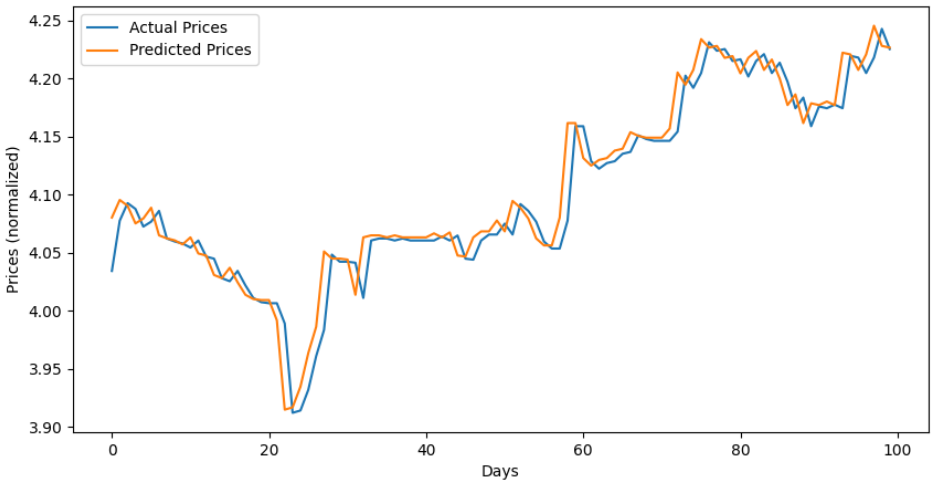

Figure 1.20: Actual vs Predicted Prices on RRHI for 100 days

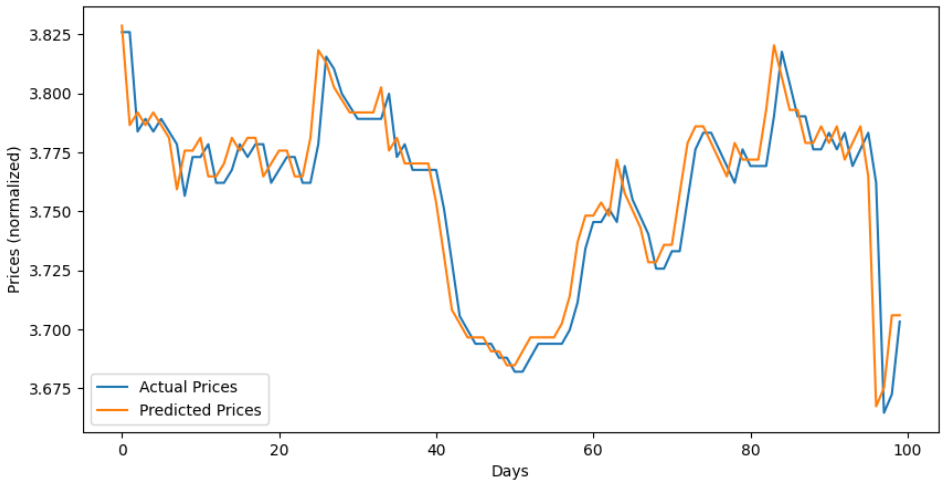

Figure 1.21: Actual vs Predicted Prices on SMC for 100 days

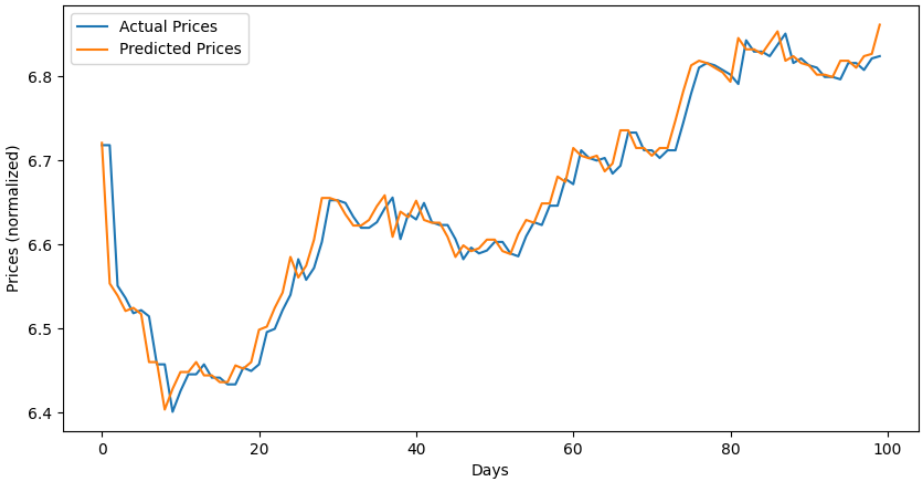

Figure 1.22: Actual vs Predicted Prices on TEL for 100 days

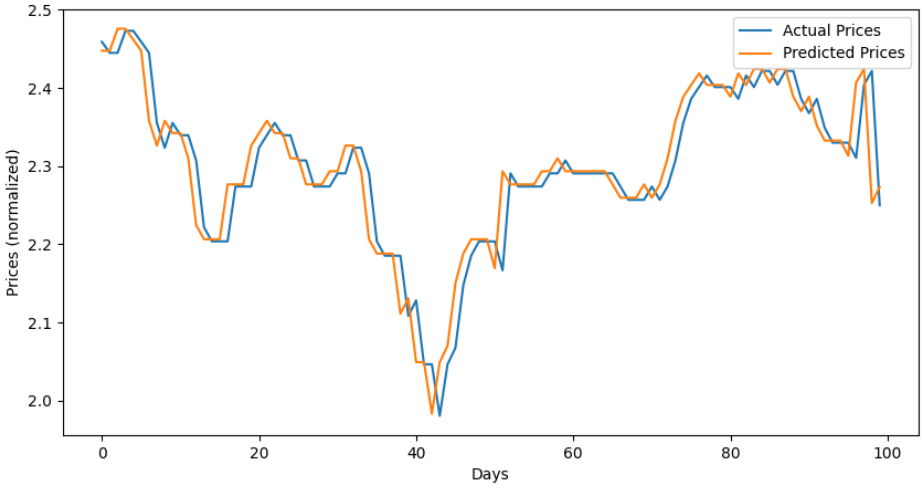

Figure 1.23: Actual vs Predicted Prices on URC for 100 days

The figures above show that the predicted prices follow the actual price trend. In addition, the discrepancy between predicted and actual prices is relatively small, as evidenced by the error metrics scores shown in Table 1.8.

However, the MAPE scores for BLOOM, ICT, JGS, LTG, and MEG range from ten billion to hundred billion. This outlier in the data is, fortunately, just the result of the applied logarithmic normalization, where some of the data in the datasets of the aforementioned stocks are in the negative range, that influence the calculation of the MAPE scores using the scikit-learn library. Because this library handles the calculation of the MAPE scores, there is no way to fix this bug. Moreover, if we take a look at the graphs of the 100 days prediction versus the actual for the aforementioned stocks in Figures 1.8, 1.11, 1.12, 1.13, and 1.14, respectively, it can still be observed that the model performs well on these stocks.

Not to mention that the other error metrics used show the same performance levels across the different stocks when the DMD-LSTM model is utilized. Meanwhile when the data normalization is removed, the MAPE scores for BLOOM, ICT, JGS, LTG, and MEG become 0.068108, 0.037207, 0.039754, 0.057332, and 0.044411 units, respectively.

Another observation from the graphs comparing actual and predicted prices over 100 days is that the predicted values appear to be higher than the actual prices. This indicates the possibility of loss because the model overestimates its prediction.

The successive predictions for the following day and up to ten days were tested using the price data from PSEI in order to make the system's predictions more useful for actual utilization. Table 1.9 shows the MAPE scores for the successive predictions of the DMD-LSTM for each days.

Table 1.9: DMD-LSTM Successive Predictions

Successive Days Predicted	**Actual and Predicted Data Ratio**	**MAPE Score**
1	100%	0.00973
2	80%	0.13403
3	60%	0.15782
4	40%	0.15646
5	20%	0.13910
6	0%	0.12494
7	-20%	0.11283
8	-40%	0.10014
9	-60%	0.08914
10	-100%	0.08976

From the table above it must be noted that the ratio values highlighted in red is to demonstrate that, despite the fact that negative ratio values shouldn't exist, doing so simply indicates that the data used to forecast the subsequent price data was overlapping by 2 to 5 times, depending on the ratio, and no longer used any actual data.

Moreover, in the integration of the DMD-LSTM model to the alamSYS, the 5 days successive predictions was utilized. Where it is shown from the Table 1.9 that it still performs well, even if the actual and predicted data ratio is only at 20%. This is also to limit the effect of stock market volatility that might affect the accuracy of the successive predictions of the model.

However, it can also be observed that the MAPE scores for successive days with zero to negative actual and predicted data ratio outperforms the MAPE scores from successive days 2 to 5 as illustrated in Figure 1.24, shown below.

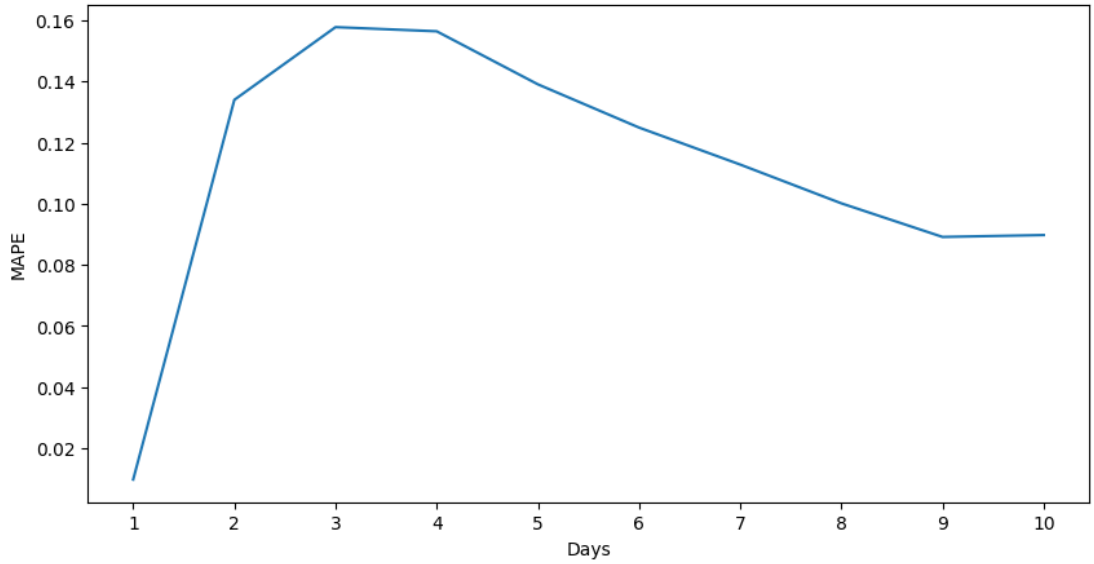

Figure 1.24: MAPE Scores for 1 to 10 (Days) Successive Predictions

Yet, since doing so might result in a poor generalization of data, they were not chosen to be the maximum consecutive days of predictions to be integrated in the alamSYS. As a matter of fact, it could be argued that these data's MAPE scores are overfitted, rendering them unreliable. On the contrary, it might also imply that the model maintains its accuracy for a longer time, even if the majority of the data used are those produced by the model itself. This could be a good thing, and may be attributed to the use of the dynamic modes, as first suggested in the study of Mann and Kutz (2015). In light of these considerations, additional testing is required to establish which of the two claims is true.

Overall, the results from the model training, evaluation, and cross-validation shows that the DMD-LSTM model developed in this special problem performs on par with the other studies that utilizes dynamic modes, as mentioned in Chapter **??** of this paper.

1.4 ALMACD Results and Discussions

The ability to predict consecutive days in the stock market is useless without a trading strategy - which allows risk mitigation and increases the probability of

positive returns over time. Trading strategies, in particular, are based on a predefined set of rules and criteria that are used to determine when to buy and sell stocks. (Hayes, 2022).

A variety of algorithmic trading, on the other hand, refers to the use of mathematical and computational techniques to determine the best position to take for a specific set of stocks. Additionally, the possibility of loss due to the influence of human emotion is eliminated. (WallStreetMojo, n.d.).

Whereas, the author used the Arnaud Legoux Moving Average Convergence and Divergence (ALMACD) trading strategy in this special problem and integrated it into the alamSYS as the system's internal trading algorithm. ALAMCD uses predicted prices for the next 5 days, as well as 200 days of actual stock price data, to track the signals and output a simple flag indicating whether to buy or sell that stock at that time.

The compounded expected return after return backtesting is provided for each stock in Table 1.10 using the optimized parameters for the fast and slow ALMA. This was done to validate the potential returns for all stocks, not just the PSEI, from which the best ALMA parameters were derived.

Table 1.10: Optimal Alma Parameters Validation Results

Stock	**Compounded Expected Return**
PSEI	113966.8500
AC	20893.1914
ALI	1072.1418
AP	690.7100
BDO	2541.9970
BLOOM	495.4600
FGEN	581.0804
GLO	60538.0035
ICT	2815.6103

Table 1.10 continued from previous page

Stock	Compounded Expected Return
JGS	1569.8650
LTG	397.2854
MEG	149.2233
MER	8586.0306
MPI	146.0200
PGOLD	721.2700
RLC	649.4767
RRHI	1050.7000
SMC	2557.0770
TEL	72070.5000
URC	3207.5394

Based on the table of expected returns above, all stocks are expected to return a positive yield over time when these optimal ALMA parameters are used. It is also worth noting that the expected return is calculated for each unit of stock, which means that if we use the expected compounded return value of MPI at PHP 146.02, which appears to be the lowest - the actual return could be at least PHP 146,020, assuming the minimum board lot required for the stocks is 1000 shares (Pesobility, n.d.).

However, despite the high potential returns, investors should proceed with caution for two reasons. First, the expected return is based on historical price data, which may not follow the trend of future price data, potentially rendering the trading algorithm obsolete (Quantified Strategies, 2023). Second, the return calculation does not account for and compensate for the additional fees associated with buying and selling the stock, which can affect the overall actual returns. Moreover, the author investigated the potential for returns by following the alamSYS predictions, as discussed further in the succeeding section.

1.5 Results and Discussions for the Real World Application of alamSYS

Following the earlier discussions of the system's theoretical performance in terms of DMD-LSTM model performance and optimal ALMACD's expected returns for each stock. It is critical to test the alamSYS's actual performance in recommending which stocks to buy and sell in a simulated real-world application. It should be noted that this is referred to as 'simulated' because no actual money exchange or utilization occurred, but all calculations are based on real-world fees and data.

The results in Table 1.11, in particular, are based on data collected from March 24, 2023 to April 12, 2023 - a total of 10 trading days. A simple set of rules was followed to standardize the results and minimize the bias that could be introduced:

(a) When the alamSYS indicated that the stock was to be purchased for that day, buy five times the required boardlot.

(b) Similarly, if the alamSYS indicates that the stock must be sold, it must be sold at five times the required boardlot, regardless of the calculated return.

(c) The PSEI will be bought and sold on a daily basis for the baseline comparison; and

(d) All calculations are done with First Metro Securities' trading calculator.

Table 1.11: Return Performance Comparison Between alamSYS and PSEI

	Realized Profit (PHP)	**Realized Gain (%)**
alamSYS	7,839.75	1.51
PSEI	-22,788.90	-13.810

The above table displays the cumulative profit and gains over a 10-day trading period, demonstrating that the realized profit from following the stock recommendations of the alamSYS yields PHP 30,628.65 more than the baseline cumulative

returns of the PSEI. This is because trading the PSEI resulted in a loss of PHP 22,788.90 (a gain of -13.810%), whereas following the trading suggestions of the alamSYS resulted in a profit of PHP 7,839.75 (a gain of 1.51%).

Given that the PSEI was down by around 2% during this period, as shown in Figure 1.25, the positive performance of the alamSYS affirms the findings from Sections 1.3 and 1.4.

Figure 1.25: PSEI Stock Market Price Trend from March 24 to April 12, 2023

Note: The figure above was generated using the chart tool from Investagrams.com

However, the loss in PSEI may be attributed to the fact that no specific trading strategy was followed, but this is still a fair battle between the two systems - as the use of the alamSYS was based solely on its recommendations. Whereas, additional technical analysis on the stock market may result in higher alamSYS gains.

Figure 1.26 depicts the day-to-day gains of both systems to further visualize the performance of alamSYS's suggestions in comparison to the performance of PSEI.

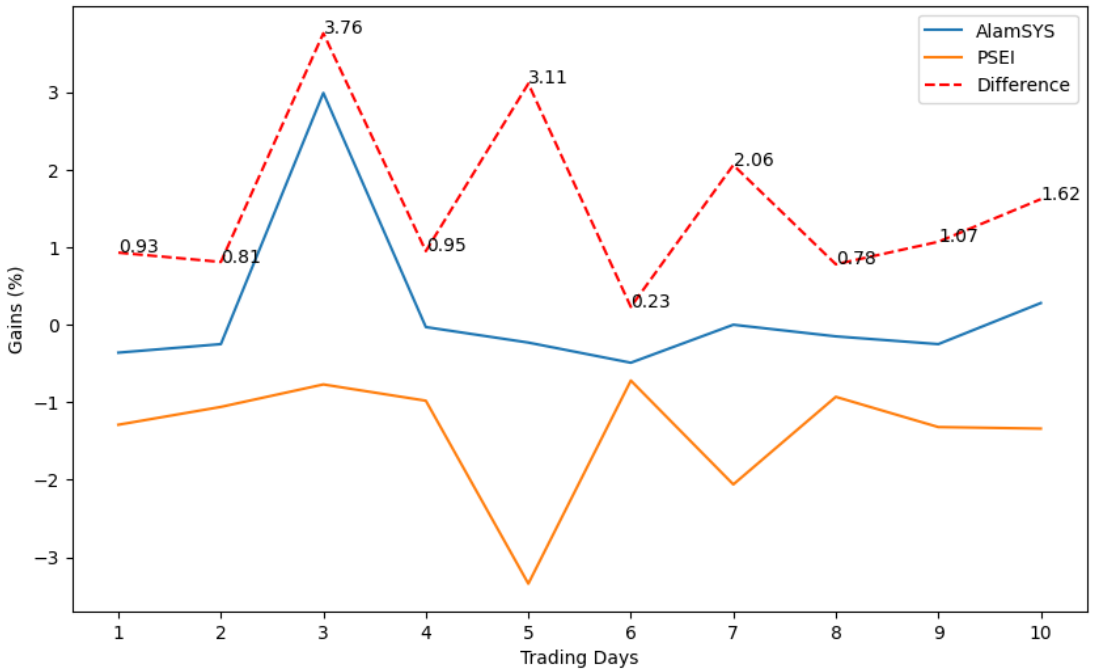

Figure 1.26: Comparison Between the Day-to-day Gains of alamSYS and PSEI

The above figure additionally demonstrates how the alamSYS performs better on a daily basis compared to the PSEI, highlighted by the 'Difference' line. Whereas the alamSYS performed better than the PSEI by 0.93%, 0.81%, 3.76%, 0.95%, 3.11%, 0.23%, 2.06%, 0.78%, 1.07%, and 1.62% over a 10-day period. Also, even though the cumulative return was positive, the alamSYS's day-to-day performance still yielded negative returns, It is worth noting, however, that it was able to reduce the risk of further losses, as can be observed in Figure 1.26.

Chapter 2

Conclusions and Future Work

Chapter 5 presents the findings from this unique problem, as well as recommendations for future research. This chapter specifically covered the following topics:

(a) Summary of the main findings discussed in Chapter 4 that are relevant to the objectives of this special problem.

(b) Contributions of the special problem were identified, emphasizing its strengths and limitations.

(c) Finally, the author provides a roadmap for future works that outlines what needs to be investigated further, additional features that can be added to the system, other external applications or systems that may connect to or be used by the current system, and so on.

2.1 Summary of Findings

The following is a summary of the key findings, in accordance with the objectives outlined in Chapter 2 of this special problem and building on the discussions of the results in Chapter 4.

(a) In connection to the development of the alamSYS:

1. When idle, the alamSYS's average CPU and memory utilization is 0.43% and 525.29 MiB, respectively. Whereas the alamPREPROCESSOR uses the least amount of CPU power, it also uses the most memory due to the scheduling program being active.
2. Both alamAPI and alamPREPROCESSOR use a small amount of CPU power when idle, which is due to the low system requirements of their linux-based base images.
3. The alamPREPROCESSOR was able to process 100 consecutive data collection and processing, indicating a 100 times higher working capacity than expected.
4. The alamPREPROCESSOR's average runtime is 48.30 seconds. Which have been affected by the internet connection speed of 52.98Mbps on average.
5. It also uses an average of 20.03% of the CPU power and 794.29 MiB of memory on load. This means that the alamPREPROCESSOR can be used with low-end devices.
6. When alamPREPROCESSOR is loaded, its CPU and memory utilization are 99.95% and 60.62% higher, respectively, than when it is idle.
7. The average deployed API response time is 1.34 seconds, which is within the acceptable range for avoiding user-perceived interruptions.
8. The delay from the tunneling service accounts for 99.30% of the response time, and the actual response time of the alamAPI is only 0.009 seconds on average; and
9. The average CPU power utilization of alamAPI and alamDB on load is 99.05% and 76.24% higher, respectively, than their idle CPU power utilizations. This allows them to process 71,000 requests in about an hour and 20 minutes. On a different note, onload memory utilization was found to be lower than idle values, but this could simply be due to dips in zero MiB memory utilization over time, as shown in Figure 1.2.

Overall, the findings show that the alamSYS and its three major components (alamPREPROCESSOR, alamAPI, and alamDB) were successfully devel-

oped and integrated. Its idle and on-load performance were guaranteed to be dependable, stable, and efficient.

(b) In connection to the development of DMD-LSTM Model, and ALMACD Trading Algorithm:

1. The DMD-LSTM model with a window size of 5 outperformed the other eight LSTM models that were trained and tested. Where the MSE, RMSE, MAE, and MAPE values are 0.000037, 0.006106, 0.004175, and 0.000001, respectively.
2. Cross-validation of the selected DMD-LSTM model reveals acceptable performance outside of its training data.
3. Up to ten days of consecutive predictions show that the DMD-LSTM model still produces a range of acceptable MAPE scores. However, to avoid the possibility of extrapolation problems affecting the predictions, the alamSYS only integrated up to 5 days of successive predictions. The extrapolation problem on LSTM models were also demonstrated in the studies by Jing, Li, and Peng (2019); and Guo, Sun, Pei, and Li (2023).
4. The integration of optimized ALMACD on top of the DMD-LSTM model ensures that the alamSYS recommendations return positive yields for all stocks.
5. The DMD-LSTM and ALMACD applications in alamSYS were tested in real-world trading for a total of 10 trading days, outperforming the PSEI on cumulative return. Furthermore, the alamSYS is capable of mitigating risk and reducing potential losses in its position due to its ability to react quickly on potential price changes attributed to its 5 days in advance predictions and reactive ALMA parameters.

Overall, the combination of DMD-LSTM predictions and ALMACD as a trading algorithm ensures that the alamSYS suggests stock positions that, in theory, yield the highest positive returns while mitigating potential risks associated with market price volatility.

(c) In connection to the development of the mobile-based test application. The

mobile-based application demonstrated the potential of the alamSYS with other external internet-connected devices, such as a smartphone device.

2.2 The Implications of alamSYS to the Society

Based on the findings summary presented in the previous section, the combination of several key components, including the alamSYS, DMD-LSTM model with the optimized ALMACD, and mobile-based test application (alamAPP), has the potential to revolutionize the Philippine financial stock market.

The alamSYS was discovered to be dependable, stable, and efficient, with quick response times and the ability to handle a huge number of queries. While, through cross-validation, the DMD-LSTM model performed within an acceptable error range and proved acceptable performance outside of its training data.

Furthermore, the incorporation of optimized ALMACD confirms that all alamSYS suggestions generate positive returns, limiting potential risks and lowering potential losses. Also, the alamAPP illustrated how the alamSYS might be used with other internet-connected devices. Overall, it positions itself as an accessible stock market investment tool for all Filipinos, with the goal of supporting and empowering investors to make informed decisions and navigate the stock market confidently, even amid difficult market situations.

To put it simply, the sociological implications of a system such as alamSYS have a substantial impact on the potential benefits it could offer to the stock market, individual investors or traders, and the Philippine economy as a whole. The system has the potential to drive economic growth by providing jobs and pushing market advances. The system could boost investor trust and involvement in the stock market by offering credible stock trading advice and limiting potential risks and losses, resulting in a more healthy and prosperous economy.

2.3 Recommendations on Future Works

This section presents a road map for future research based on the insights gained in the preceding sections of this chapter. The roadmap outlines key areas for additional research, potential system enhancements, opportunities for integration with external applications or systems, and other applicable considerations.

(a) In connection to the development of the alamSYS:

1. Automated Model and Trading Algorithm Evaluator and Trainer - Given the possibility of changes in market price trend data over time, the patterns on which the current DMD-LSTM is based may become obsolete. As a result, it may be advantageous to incorporate an additional component that would revalidate the deep learning model as well as the trading algorithm, and if it performs worse than 10% of the current performance, it would automatically rerun the training with the new data until it finds a model and optimal parameters that performs inline or even better than the current one.
2. Alternative System Deployment Methods can be Explored - The tunneling service was found to be responsible for the majority of the delay in the alamSYS response time after deployment. As a result, it may be beneficial to investigate better alternatives, such as deploying the system on a server with a fast and reliable internet connection, though this may necessitate a larger budget. Another low-cost option is to deploy the system on cloud computing platforms such as AWS and Google Cloud.
3. Integration of other Models and Trading Algorithm - Initially, it was planned to integrate multiple combinations of machine learning models and trading algorithms into alamSYS, but due to time constraints, this was not pursued. However, the system was designed with this in mind, and the source code already allows for this feature to be integrated. Future developers, for example, can investigate the integration of ARIMA-based models in alamSYS. This provides users with multiple sources of data-driven information on the market's stocks. Where their decision is not limited to a single piece of information.

4. A Closer Examine on the Stock Suggestions of the System - Because the alamSYS currently only logs its daily predictions, it may be useful for future research to examine the trend of these predictions and try to see the accuracy of this suggestion, as well as potentially add improvements to the system; and
5. System Maintenance and Additional Features or Improvements to the Current System - The completion of this paper does not indicate the end of the alamSYS's development lifetime; it is expected that the developer, and other developers in the future, will continue to work on improving the system from its initial deployment, using the continuous integration, continuous delivery, and deployment - CI/CD pipeline (RedHat, 2022).

(b) In connection to the development of DMD-LSTM Model, and ALMACD Trading Algorithm:

1. Solving for the Observed Prediction Offset - The figures comparing predicted and actual price data for each stock revealed a distinct offset pattern. With this in mind, it may be worthwhile to investigate further and add potential additional steps to the predictions to compensate for the offset and improve the model's performance. This could be done during training or as a post-processing technique.
2. Exploration of Other Machine Leaning Models - There are numerous machine learning models that can be investigated for their potential to predict stock market movement. As a result, it is recommended that future developers or researchers be able to devise novel approaches to creating and integrating other models based on the findings and limitations of this special problem.
3. Redefine the Optimal Fast, and Slow ALMA Parameters - The current optimal ALMA parameters, which do not account for the additional fees for buying and selling stock positions, could possibly not yield to the highest return potentials. As a result, future developers must investigate the possibility of including these fees from the Philippine Stock Exchange in order to better compute the optimal ALMA parameters, and then compare this results on the results of the current optimized ALMA parameters deployed in the alamSYS; and

4. Exploration of Other Trading Strategies and Algorithms - The stock market strategies is not confined on the utilization of moving averages such as Arnaud Legoux Moving Average in creating simple yet effective trading strategies. Hence, it would be beneficial if other trading strategies were explored being integrated alongside the machine learning models to be developed as well.

(c) In connection to the development of the mobile-based test application, future developers are encouraged to use other internet-connected or Internet-of-Things (IoT) devices such as smart speakers, smartwatches, smart glasses, and so on. To create an application that makes effective use of the alamSYS. Making certain that it is widely used.

References

Guo, S., Sun, N., Pei, Y., & Li, Q. (2023). 3d-unet-lstm: A deep learning-based radar echo extrapolation model for convective nowcasting. *Remote Sensing*, *15*(6). Retrieved from https://www.mdpi.com/2072-4292/15/6/1529 doi: 10.3390/rs15061529

Hayes, A. (2022). *What is a trading strategy? how to develop one.* Retrieved April 22, 2023, from https://www.investopedia.com/terms/t/trading-strategy.asp

Jing, J., Li, Q., & Peng, X. (2019). Mlc-lstm: Exploiting the spatiotemporal correlation between multi-level weather radar echoes for echo sequence extrapolation. *Sensors*, *19*(18). Retrieved from https://www.mdpi.com/1424-8220/19/18/3988 doi: 10.3390/s19183988

Juviler, J. (2022). *Api response time, explained in 1000 words or less.* Retrieved April 22, 2023, from https://blog.hubspot.com/website/api-response-time

Mann, J., & Kutz, J. N. (2015, 8). Dynamic mode decomposition for financial trading strategies.

Pesobility. (n.d.). *Board lot of philippine stock exchange.* Retrieved April 22, 2023, from https://www.pesobility.com/ref/boardlot

Quantified Strategies. (2023). *Why trading strategies are not working (how to know when trading systems stopped performing?).* Retrieved April 22, 2023, from https://www.quantifiedstrategies.com/why-trading-strategies-are-not-working/

RedHat. (2022). *What is ci/cd?* Retrieved April 23, 2023, from https://www.redhat.com/en/topics/devops/what-is-ci-cd

S.Gopal Krishna Patro, K. K. s. (2015). Normalization: A preprocessing stage. *International Journal of Advanced Research in Science, Engineering and*

Technology, *205*. Retrieved from 10.17148/IARJSET.2015.2305

WallStreetMojo. (n.d.). *Algorithmic trading.* Retrieved April 22, 2023, from https://www.wallstreetmojo.com/algorithmic-trading/

Zwetsloot, R. (2019). *Raspberry pi 4 specs and benchmarks.* Retrieved April 22, 2023, from https://magpi.raspberrypi.com/articles/raspberry-pi-4-specs-benchmarks