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

Abstract

Investing in the stock market can provide significant benefits to both individuals and the economy as a whole. Many people, however, are hesitant to participate due to the high risk involved. This problem is worsened by a lack of access to accurate and reliable stock market information, which is frequently reserved for those who can afford expensive research. Which, when compared to larger private institutions, there is a gap in information available to the public, creating an unfair advantage in the investing market. As such, the objective of this study was to develop alamSYS, a system that could collect and process stock market data in order to provide comprehensive stock position suggestions with the utilization of DMD-LSTM model and ALMACD. In addition, alamAPP, which showcases the main functionalities of the alamSYS, was also developed. Wherein, the results show that alamSYS, with its efficient and robust components, together with DMD-LSTM predictions along with ALMACD as a trading algorithm, has the potential to provide credible stock trading advice that yields high returns while minimizing potential risks. Moreover, the mobile-based test application's successful integration with external devices demonstrates the system's adaptability and versatility. Ultimately, the socioeconomic impact of alamSYS provide essential advantages to the stock market, individual investors or traders, and the Philippine economy as a whole by bringing about economic growth, increasing investor trust and participation, and limiting potential risks and losses.

Keywords: Systems Architecture, Computing Methodologies, Applied Computing, Human-Centered Computing

Contents

List of Figures

List of Tables

Chapter 1

Materials and Methods

This chapter discusses the materials and methods used for the design and development of the system: alamSYS. Specifically, the following are discussed in this chapter: (a) Development Tools and Software Requirements; (b) System Diagrams; (c) Hardware Requirements; and (d) Methodology.

1.1 Development Tools and Software Requirements

The development of the alamSYS utilized the following development tools and software requirements:

1.1.1 Development Tools

To develop the alamSYS, the following development tools were utilized: (a) Visual Studio (VS) code, as the project's integrated development environment; (b) MongoDB Compass, used for managing the alamDB; and (c) Github, as the version control system and project's source code repository.

1.1.2 Software Requirements

In the development of the alamSYS and its components the following software and their respective versions were utilized: (a) Python *versions 3.9.10 and 3.11.1*; (b) MongoDB; (c) Jupyter Notebook; (d) Docker; (e) Docker-Compose; (f) Dart and Flutter; (g) Git; and (h) Github.

Moreover, the following Python libraries and modules are also utilized: (a) FastAPI *version 0.85.0*; (b) mongoengine *version 0.24.2*; (c) json; (d) datetime; (e) os; (f) schedule *version 1.1.0*; (g) requests *version 2.28.1*; (h) numpy *version 1.23.5*; (i) tensorflow *version 2.11.0*; (j) matplotlib *version 3.7.0*; (k) pyDMD *version 0.4.0post2301*; and (l) pandas *version 1.5.3*.

Additionally, the following flutter packages are utilized for the development of the alamAPP: (a) http *version 0.13.5*; (b) path_provider *version 2.0.13*; (c) syncfusion_flutter_charts *version 20.4.52*; and (d) lottie *version 2.2.0*

1.2 System Diagrams

In this section, the appropriate system diagrams is shown and discussed. This shall help in the understanding of the system's features, data flow, and processes.

1.2.1 Top-Level Overview Diagram of the alamSYS and Its Interactions to External Systems

Figure 1.1 shows the top-level overview of the alamSYS and its interactions to any third-party or external applications.

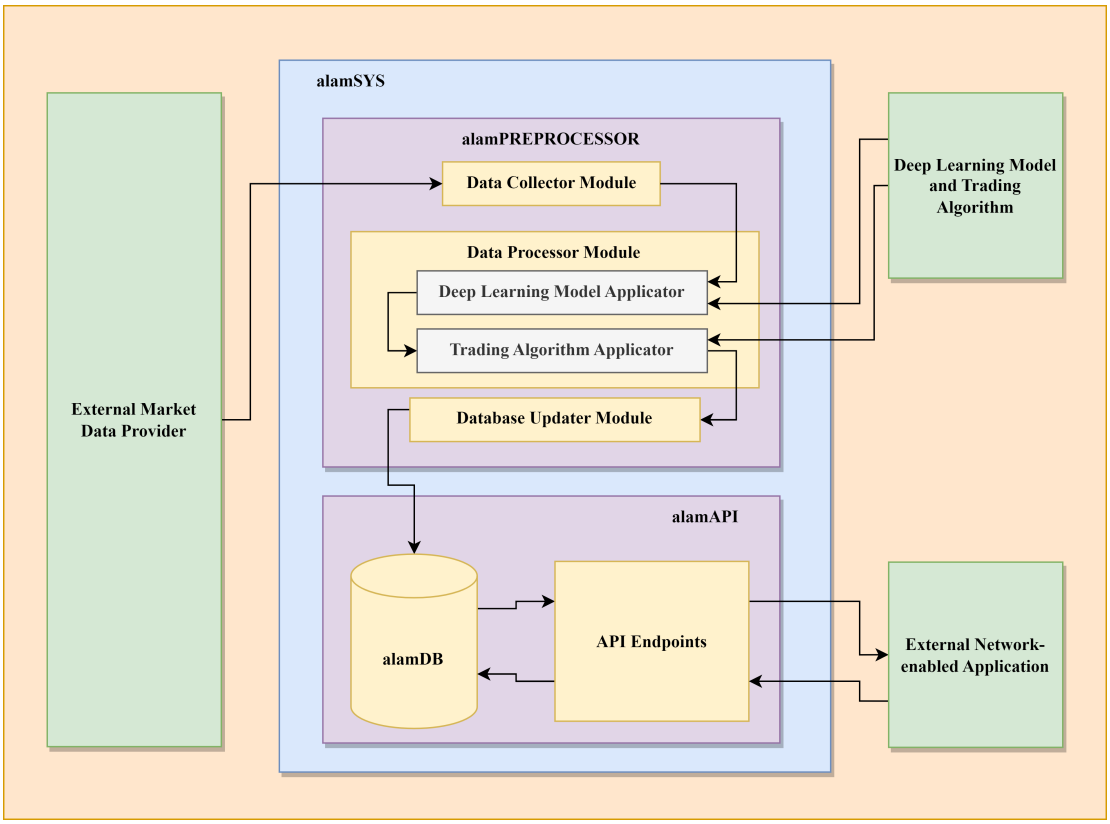

Figure 1.1: Top-Level Overview of the alamSYS and Interactions with External Applications/Systems

As shown from the figure above, the alamSYS is connected to three external entities: (1) External Market Data Provider, which provides the system with the needed historical market data; (2) Deep Learning Model and Trading Algorithm, in the case of this special problem, a deep learning model was developed and utilized by the system, however as previously discussed the system is created to accept any other machine learning models and or proprietary trading algorithms that other developers may or want to develop in the future; and (3) External Application, which can be a web-based or mobile-based application, that utilizes the functionalities provided by the alamSYS, through the API endpoints.

On the middle of the diagram the alamSYS is observed to have three main components, namely, (1) alamPREPROCESSOR, which is further divided into sub-components: (a) Data Collector Module (DCM), which collects the data from

the external market data provider; (b) Data Processor Module (DPM), which processes the historical market data collected by applying the developed machine learning model and sending it to the (c) Database Updater Module (DUM); (2) alamDB, which is based on MongoDB, which is a document-based and non-relational database; finally, the database is connected to the (3) alamAPI, which contains the API endpoints which processes the request and responses of the system to any external application connected to the API via a network.

1.2.2 Process Flow Diagram

The diagram shown in Figure 1.2 the different processes that the system undergoes, once it has been deployed in the server.

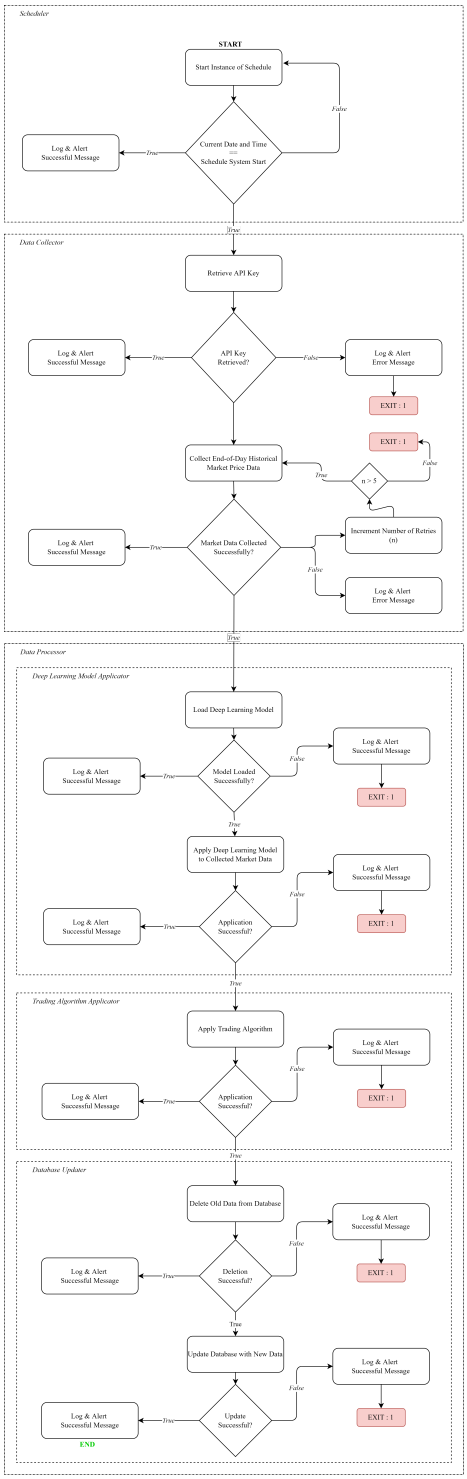

Figure 1.2: Full Overview of the Process Flow Diagram for the alamSYS

To better view and understand the flow, the processes were divided and presented in the succeeding parts.

Scheduler

Using Python's Schedule Library, an instance of a scheduled task is initialized upon the startup of the alamSYS. The scheduled task shall execute every Mondays to Fridays at exactly 6 P.M. Where the whole scheduling process is illustrated in Figure 1.3.

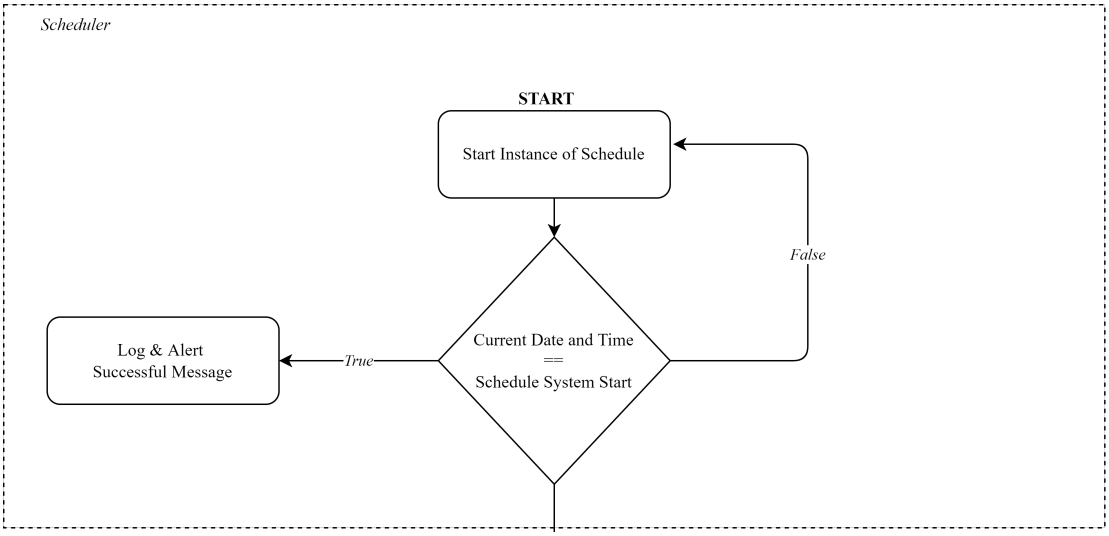

Figure 1.3: Overview of the Process Flow Diagram for the Scheduler

Data Collector

The collection of end-of-day (EOD) market data is the first process in the scheduled task. To collect EOD market data, the system first looks for the EODHD API key, which is stored in system variables or provided by the user in the tools directory. If the system is unable to locate an API key, it logs the error and notifies the user before exiting the program.

Once the API key is obtained, the system connects to the EOD market data provider and attempts to collect all market data five times. If it fails to collect data for the fifth time due to errors (i.e. incomplete payments, unstable network, and no established internet connection), the system logs the error, sends an alert to the user, and exits the program.

All successful processes are also logged and sent to the command line interface to notify the user. Figure 1.4 depicts this, as well as the entire data collection process. The flow of processes discussed above can be observed in Figure 1.4.

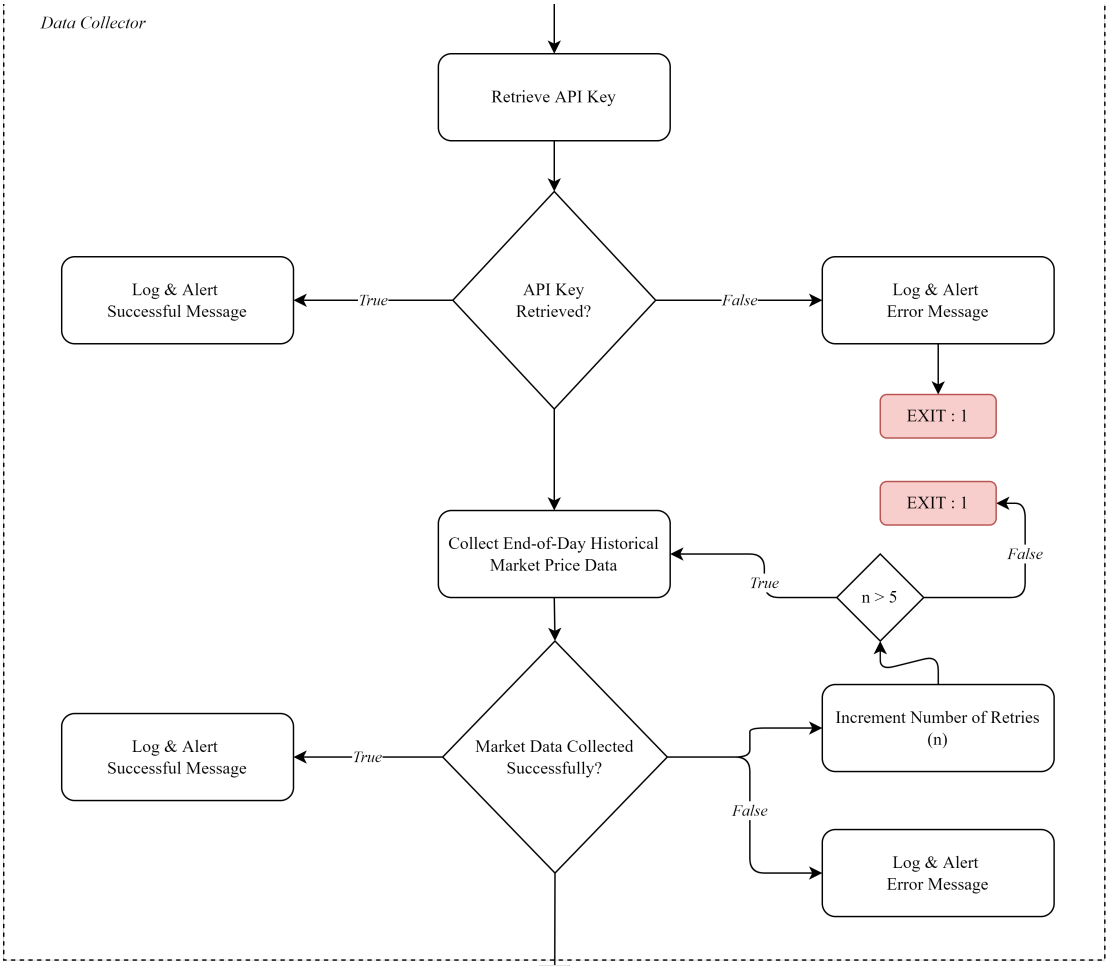

Figure 1.4: Overview of the Process Flow Diagram for the Data Collector

Data Processor

Data Processor is a process that is divided into three subprocesses, as shown in Figure 1.5.

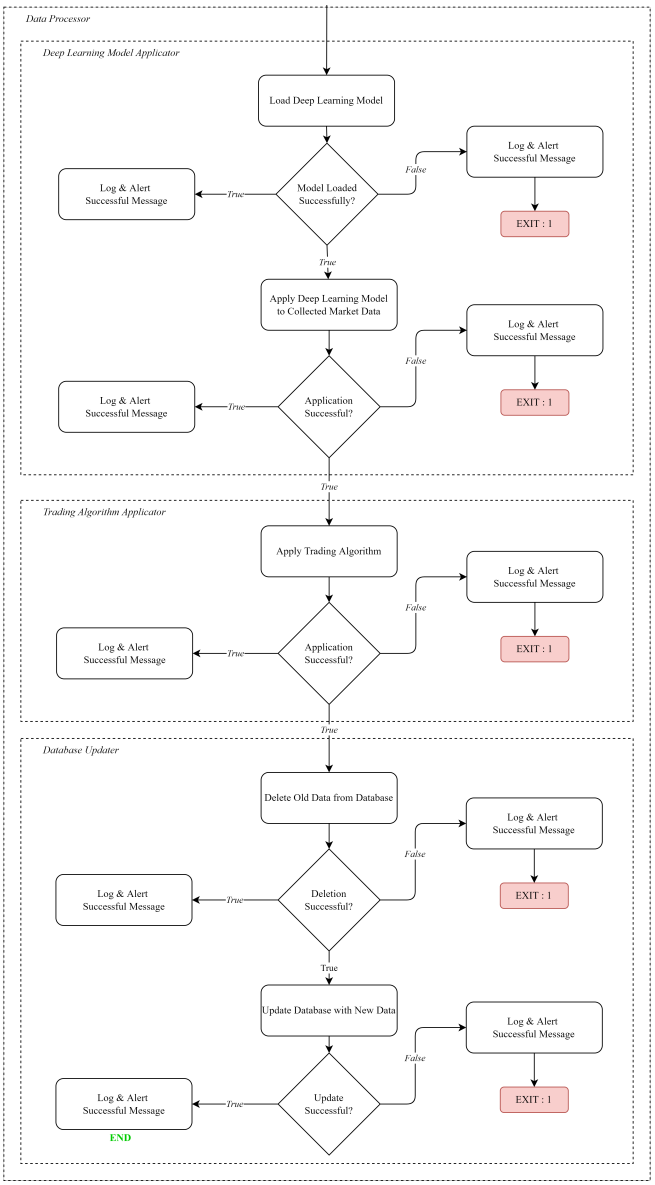

Figure 1.5: Overview of the Process Flow Diagram for Data Processor

Where the three subprocesses are as follows:

(a) Deep Learning Model Applicator - This subprocess applies the deep learning model to the collected EOD market data for each stock. Specifically, alamSYS applies the DMD-LSTM model developed as part of this special problem. This is done as illustrated in Figure 1.6.

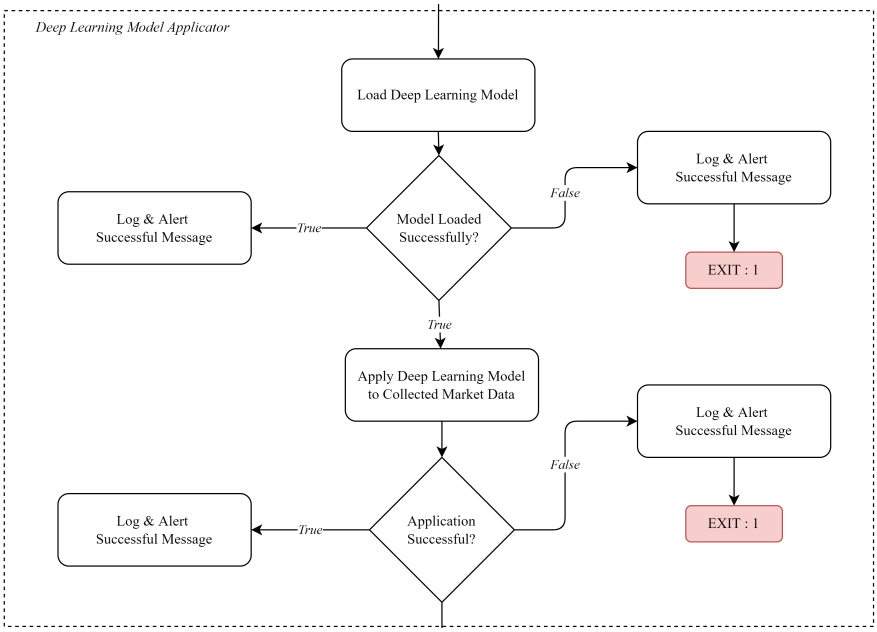

Figure 1.6: Overview of the Process Flow Diagram for the Deep Learning Model Applicator

(b) Trading Algorithm Applicator - This subprocess applies the trading algorithm to the output data from the deep learning model applicator. Specifically, alamSYS applies ALMACD algorithm developed as part of this special problem. This is done as illustrated in Figure 1.7.

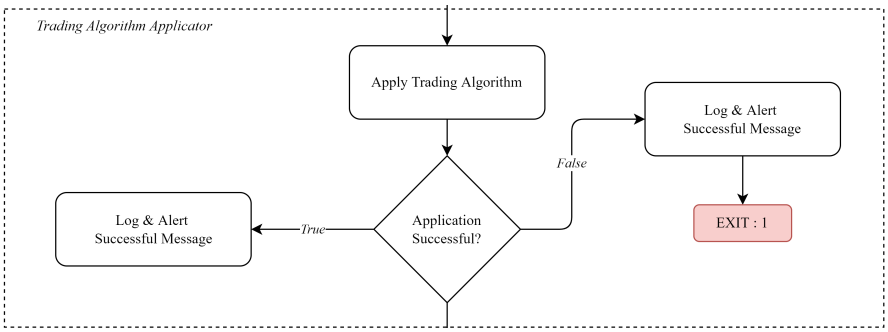

Figure 1.7: Overview of the Process Flow Diagram for the Trading Algorithm Applicator

(c) Database Updater - This subprocess updates the database with the output data from the trading algorithm applicator. This is done as illustrated in Figure 1.8.

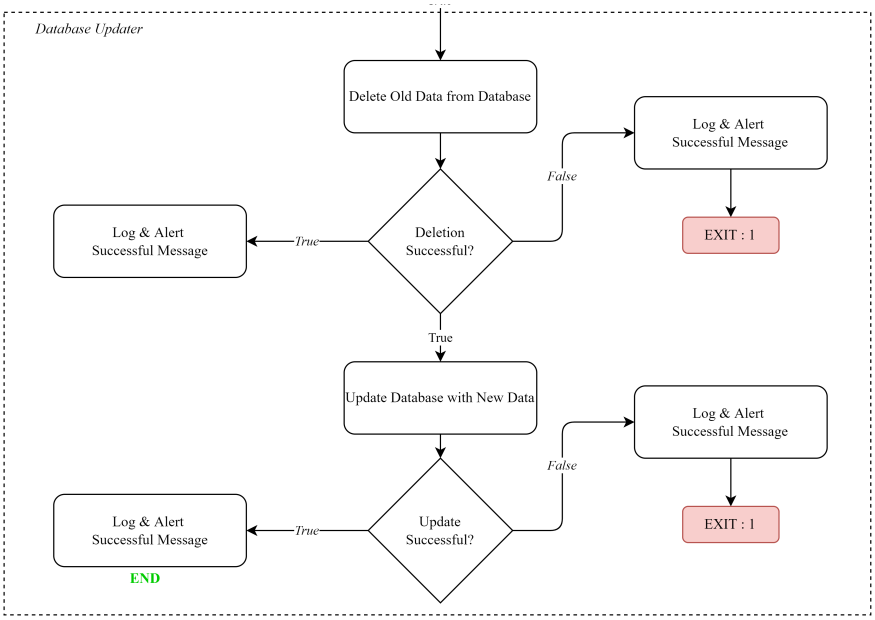

Figure 1.8: Overview of the Process Flow Diagram for the Database Updater

1.2.3 Data-Flow Diagram (DFD)

A data-flow diagram (DFD) helps to understand how processes work and how data flows from one process to the next. This is especially important because it provides an overview of the data's security by demonstrating how it can be accessed. In the case of alamSYS, the only publicly accessible data is the listed stock to buy and sell, as well as other functions as provided in its database and as permitted by the API endpoints.

Furthermore, the DFD paradigm used in the diagrams in this section adheres to the Gane-Sarson DFD symbols, which employ four basic symbols: (1) Entity / External Entity; (2) Data Flow; (3) Process; and (4) Data Store (VisualParadigm, n.d.)

Context Diagram

The overview of the entire process is depicted in a context diagram of the system, labeled process 0, as shown in Figure 1.9.

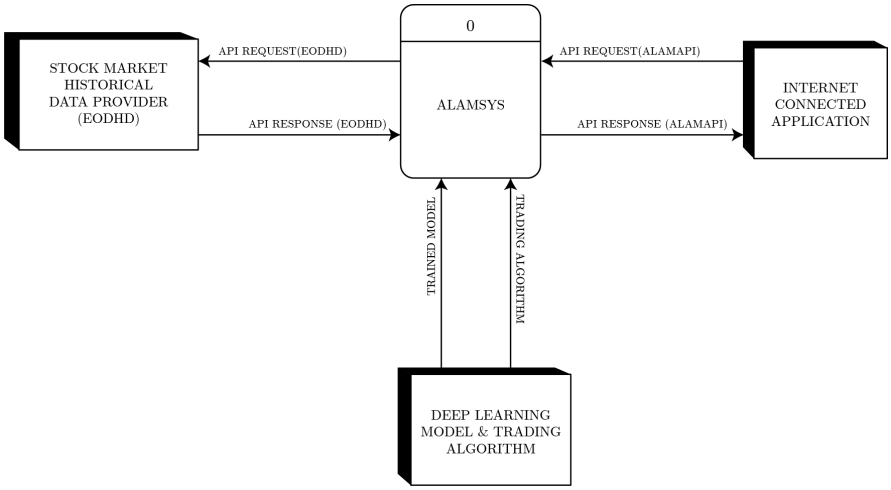

Figure 1.9: Context Diagram of the alamSYS

The diagram above depicts the root process (0), which is the alamSYS it-

self, and is linked to three external entities: (a) Stock Market Historical Data Provider, which was provided by EODHD; (b) Deep Learning Model & Trading Algorithm,these were developed alongside the alamSYS, specifically the DMD-LSTM Model and ALMACD, respectively; and (c) Internet Connected Applicatio, these are any type of applications with internet access. Other applications may include a web-based application, a smart home speaker, and so on.

The data flow lines shown Figure 1.9 include the following: (a) API Request(EODHD), this is the request sent to the EODHD API to collect the stock market historical data; (b) API Response(EODHD), this is the response received from the EODHD API, which contains the end of day stock market data; (c) API Request(alamAPI), this are the requests sent by any internet connected application to the alamSYS via the alamAPI; (d) API Response(alamAPI), this are the responses sent by the alamSYS to any internet connected application via the alamAPI; (e) Trained Model, this is the trained model, referring to the DMD-LSTM, that was used to predict the price movement of the stocks in the Philippine Stock Market; and (f) Trading Algorithm, this is the deployed trading algorithm, referring to the ALMACD, that was used to determine the entry and exit signals of the stocks in the Philippine Stock Market.

In connection, the following are the API requests that can be utilized: (a) Home, this API endpoint returns a greeting message. Which should notify the user that they have connected to the alamAPI successfully; (b) Stocks to Buy, these API endpoints return a json data of recommended stocks to buy based on the current market price, the predicted price uptrend, and the entry signal of the trading algorithm in use; (c) Stocks to Sell, these API endpoints return a json data of recommended stocks to sell based on the current market price, the predicted price downtrend, and the exit signal of the trading algorithm in use; (d) ML Model Info, these API endpoints return a json data of the Machine Learning Models used in the alamSYS, as well as their associated information; and (e) Stock Risks Profile, these API endpoints return a json data of stocks in the alamSYS as well as their risk values based on value at risk (%), volatility (%), and drawdown (%).

DFD of Diagram 0

To better understand how each data stream entering and exiting the root process is processed, it is essential to look inside the inner workings of the root process, which is illustrated in the DFD of Diagram 0, as shown in Figure 1.10.

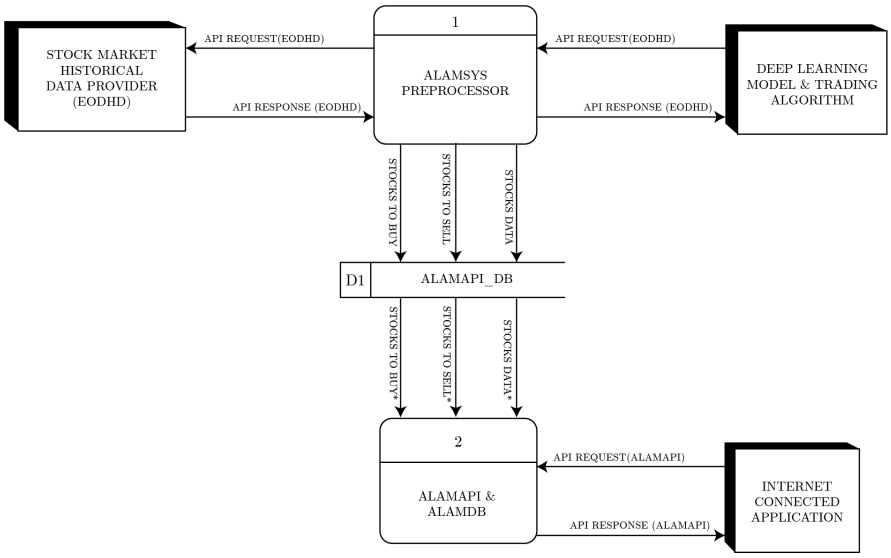

Figure 1.10: DFD of Diagram 0

From the figure above, the root process, has two main processes: (a) alamSYS Preprocessor, which is the system's stock market data processing unit, which deploys the deep learning model (DMD-LSTM), and the trading algorithm (ALMACD) to predict the price movement of the stocks in the Philippine Stock Market; and (b) alamAPI & alamDB, which is the system's API and database unit, which is responsible for processing the API requests and responses, as well as storing the data of the system, respectively.

DFD of Diagram 1

To better understand the internal workings of the Process 1, it is useful to check the DFD of that process, which is illustrated in Figure 1.11

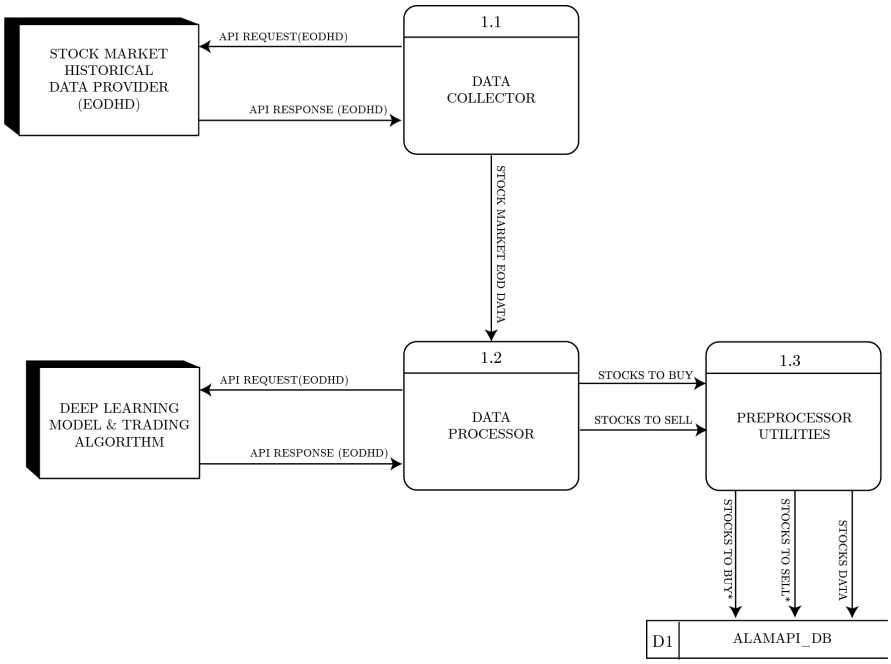

Figure 1.11: DFD of Diagram 1

From the figure shown above, it can be observed that Process 1 is composed of three internal processes, which are as follows: (a) Data Collector, which is the main process responsible for collecting the historical market data using EODHD End of Day Market Data API; (b) Data Processor, which is the main process responsible for processing the collected stock market data using the DMD-LSTM Model and the ALMACD Trading Algorithm; and (c) Preprocessor Utilities, these are a set of tools that contains the utilities used by the data processor, such as the initialization of the database, database related actions, database models, stock symbols, and logs and alerts module.

DFD of Diagram 1.2

This shows the processes inside the process 1.2, which is the data processor.

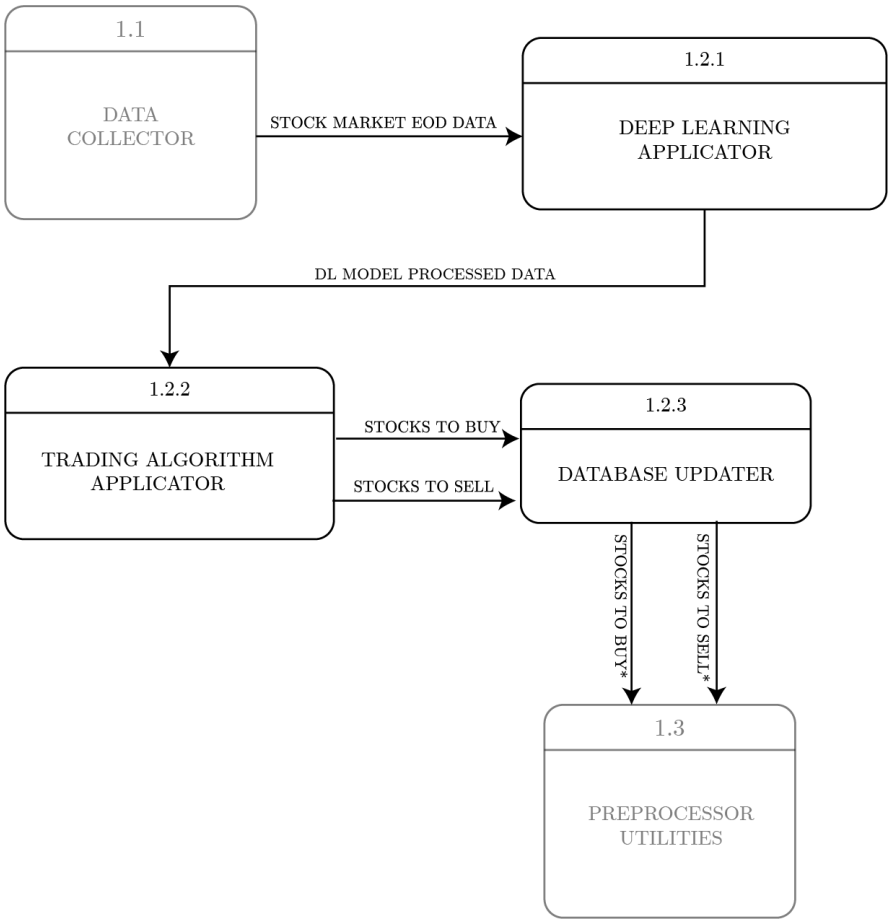

Figure 1.12: DFD of Diagram 1.2

The data processor is further composed of three processes, which are as follows: (a) Deep Learning Applicator, this subprocess applies the DMD-LSTM model to the collected stock market end-of-day (eod) data; (b) Trading Algorithm Applicator, the eod data alongside the list of predicted stock prices composes the "DL Model Processed Data", which is sent to this subprocess. This subprocess applies the ALMACD to better determine the entry (buy or hold), and exit (sell) signals for each stocks; and (c) Database Updater, a subprocess responsible for updating the contents of the database, based on the data processed by the Deep Learning Applicator and Trading Algorithm Applicator.

DFD of Diagram 2

Figure 1.13 shows the inner processes of the Process 2 (alamAPI & alamDB).

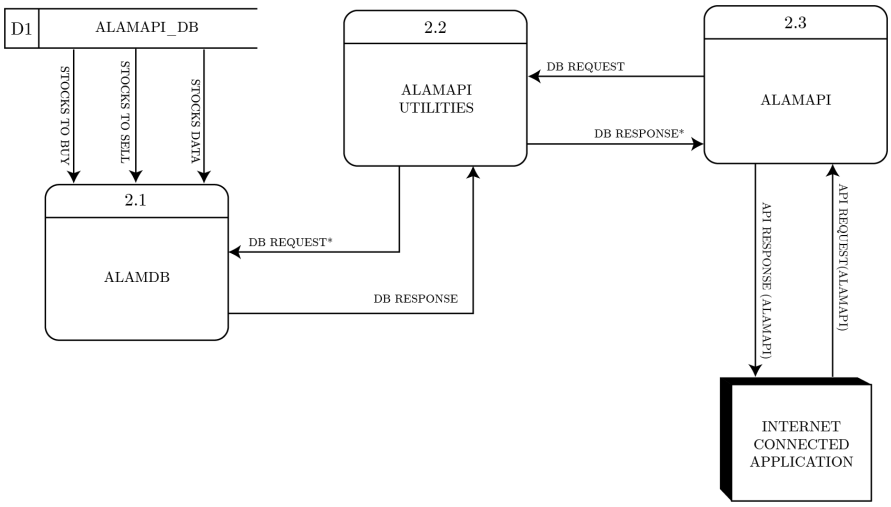

Figure 1.13: DFD 2: Data-Flow Diagram for the alamSYS

The figure above shows three internal processes of the Process 2, namely: (a) alamDB, this process, processes the database request sent by the alamAPI as requested by the Internet Connected Application through the utilization of alamAPI utilities. It also sends the database response to the alamAPI via the same utilities module; (b) alamAPI Utilities, this process serves as a mediator of database request and responses between the alamDB and alamAPI; and (c) alamAPI, this process contains all the API endpoints for the alamAPI, which is responsible for processing the requests and API responses from and to the connected users.

1.2.4 Object Document Mapper (ODM) Diagram

Because the system's database is non-relational, an Object Document Mapper (ODM) diagram rather than an Entity Relationship Diagram (ERD) is shown in this section. The ODM diagram is shown in Figure 1.14:

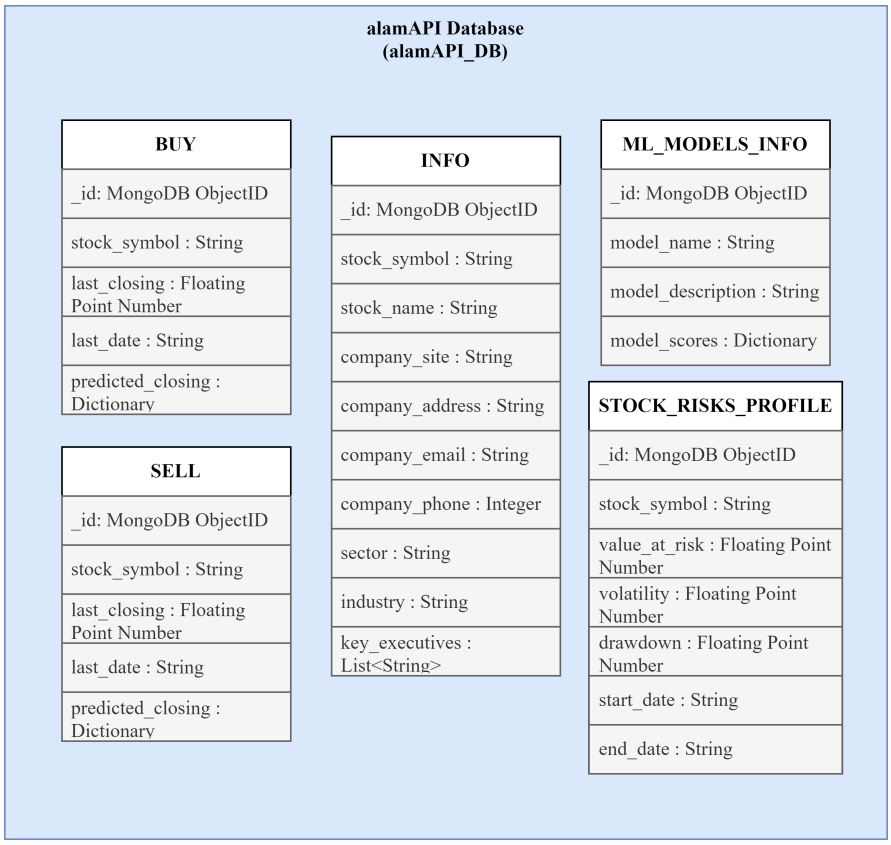

Figure 1.14: Object Document Mapper for the alamSYS Database

As shown from the Figure 1.14, the "alamAPI_DB" is the database name of the system. Where it is composed of five collections namely:

(a) Buy – this collection contains all the stocks that the data processor predicted and classified as a stock to buy. A sample of which is presented in Figure 1.15.

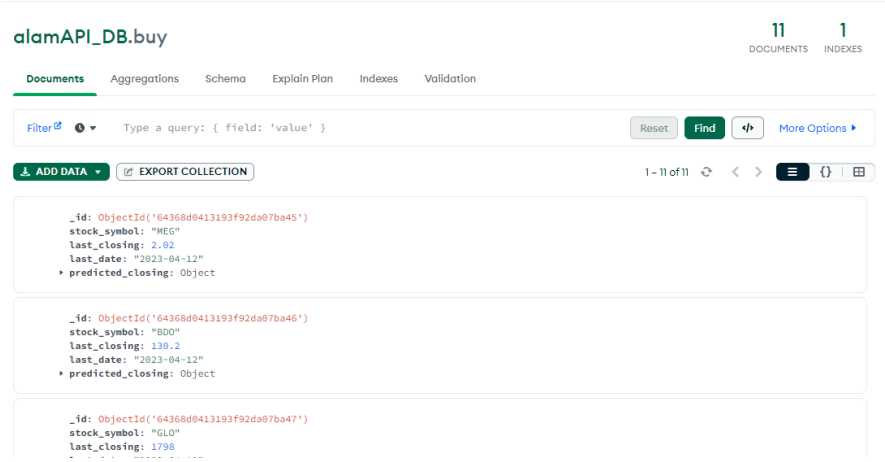

Figure 1.15: Sample Buy Collection from the alamSYS Database

(b) Sell – this collection contains all the stocks that the data processor predicted and classified as a stock to sell. a sample of which is presented in Figure 1.16.

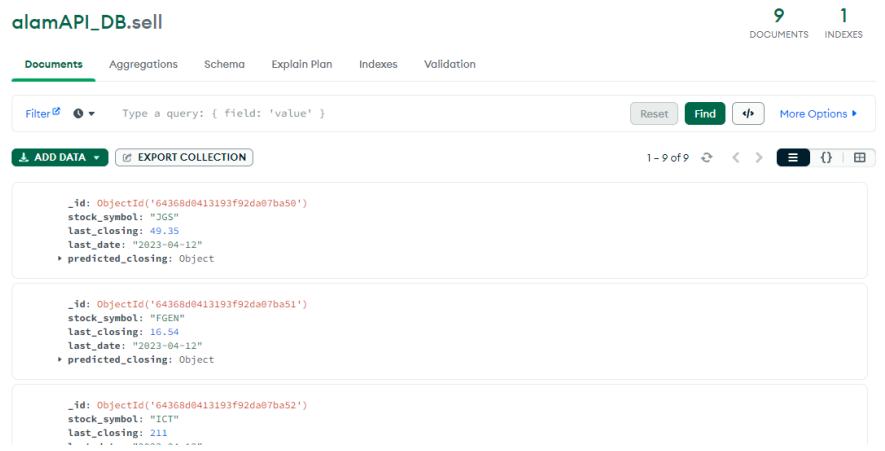

Figure 1.16: Sample Sell Collection from the alamSYS Database

(c) Info – this collection contains the general and relevant information about a stock, or the general company information. Such as the stock symbol, stock name, company site, company address, company email, company phone number, sector, industry, and the company's key executives. Where all of this information are gathered from their official listing accessed in the database of the Philippine Stock Exchange (PSE). A sample of which is

presented in Figure 1.17.

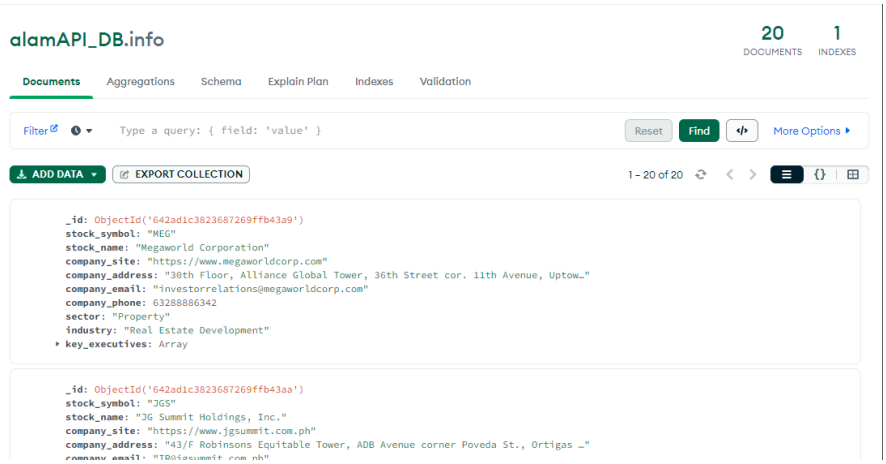

Figure 1.17: Sample Info Collection from the alamSYS Database

(d) Machine Learning (ML) Models Info – this collection contains the details about the Machine Learning Model/s deployed in the system. For the current alamSYS, only one model is deployed, which is the DMD-LSTM model. A sample of which is presented in Figure 1.18.

Figure 1.18: Sample ML Models Info Collection from the alamSYS Database

(e) Stock Risks Profile - this collection contains the details about the risk profiles for each stock. A sample of which is presented in Figure 1.19.

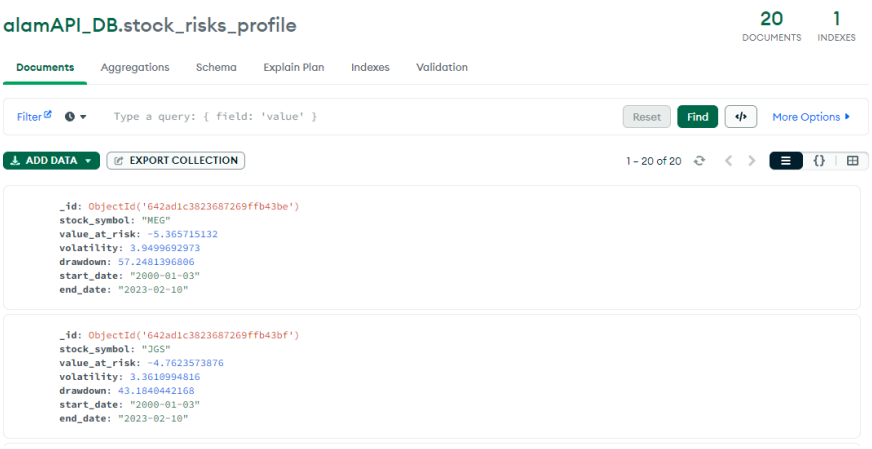

Figure 1.19: Sample Stock Risks Profile Collection from the alamSYS Database

Note that each collection are their own separate entities, hence the database is called non-relational, as the documents are not in any way related to each other.

1.2.5 Deep Learning Model Diagram

In this section, the process on how the deep learning model was developed is shown in Figure 1.20. Wherein, the process overview is based on the Fine-Tuned Support Vector Regression Model for Stock Predictions by Dash and Dash (2016).

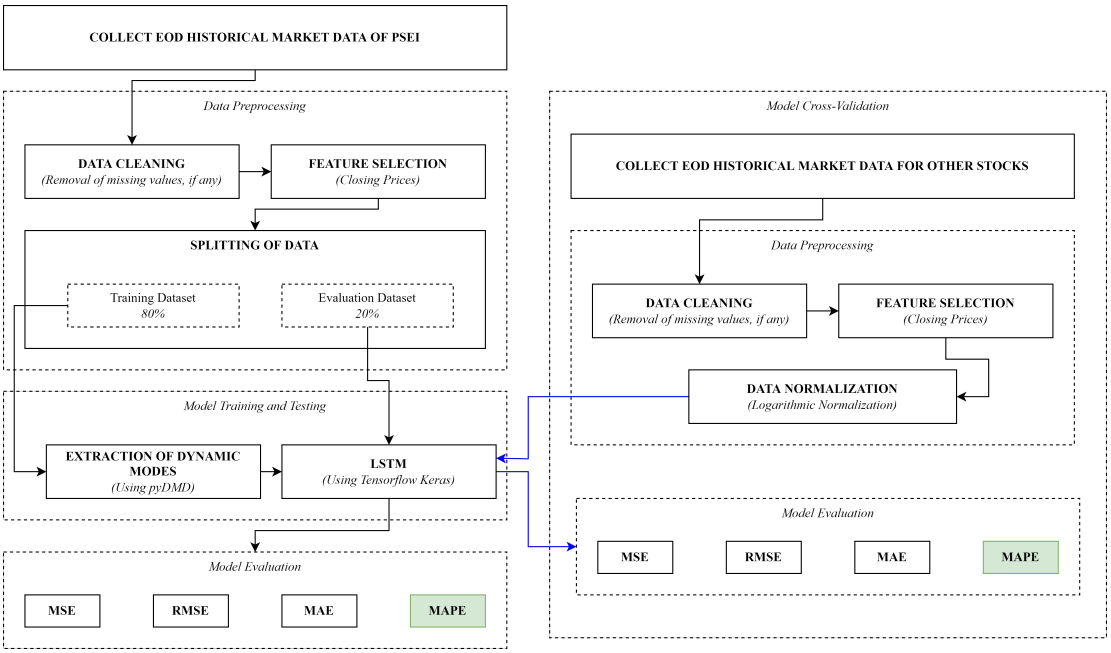

Figure 1.20: DMD-LSTM Model Development Methodology for alamSYS

Data Collection

The market data used to develop the DMD-LSTM model was obtained through EODHD's end-of-day market data API. While PSEI market data was used for model training and testing, market data from other stocks was used for cross-validation of the DMD-LSTM model. The following are the specifics of the stocks gathered:

Table 1.1: Collected Market Data Details

Stock	Data Count	Start Date	End Date
AC	6809	June 27, 1994	February 10, 2023
ALI	6789	June 27. 1994	February 10, 2023
AP	3795	July 16, 2007	February 10, 2023
BDO	5041	May 22, 2002	February 10, 2023
BLOOM	3033	October 30, 2000	February 10, 2023
FGEN	4142	February 02, 2006	February 10, 2023
GLO	6707	January 03, 1995	February 10, 2023

Table 1.1 continued from previous page

Stock	Data Count	Start Date	End Date
ICT	6805	January 03, 1995	February 10, 2023
JGS	6525	June 27, 1994	February 10, 2023
LTG	3774	February 06, 1995	February 10, 2023
MEG	6751	January 03, 1995	February 10, 2023
MER	6799	June 27, 1994	February 10, 2023
MPI	3888	December 18, 2006	February 10, 2023
PGOLD	2762	October 06, 2011	February 10, 2023
PSEI	5675	January 03, 2000	February 10, 2023
RLC	5879	June 27, 1994	February 10, 2023
RRHI	2253	November 11, 2013	February 10, 2023
SMC	6799	June 27. 1994	February 10, 2023
TEL	6814	June 27, 1994	February 10, 2023
URC	6135	June 03, 1995	February 10, 2023

Data Preprocessing

Data preprocessing before model training and testing is composed of three main processes which are as follows: (a) Data Cleaning, this was done to clean any missing values from the data; (b) Feature Selection, closing prices was selected as the main feature of the model; and (c) Splitting of Data, data was split in the ratio of 80:20 for testing and training data, respectively.

Meanwhile for the data preprocessing for cross-validation, the following processes were done: (a) Data Cleaning, this was done to clean any missing values from the data; (b) Feature Selection, closing prices was selected as the main feature of the model; and (c) Data Normalization, using logarithmic normalization method, the data was normalized Logarithmic normalization was utilized to help solved the problem with the data having extreme ranges, which affects the evaluation of the cross-validation. In essence it was used to enable data stability, and increase interpretability (Baeldung, 2022; Tuychiev, 2021; Andrew, 2019).

Model Training and Testing

Using pyDMD the dynamic modes was extracted from the training and testing data, these extracted values alongside the actual closing price data were utilized for the training of an LSTM model using Tensorflow Keras Library.

There are a total of eight model variations trained and tested, which are as follows: (1) Baseline LSTM with window size of 5; (2) Baseline LSTM with window size of 10; (3) Baseline LSTM with window size of 15; (4) Baseline LSTM with window size of 20; (5) DMD-LSTM with window size of 5; (6) DMD-LSTM with window size of 10; (7) DMD-LSTM with window size of 15; and (8) DMD-LSTM with window size of 20. Where the best performing model was used for the cross-validation and was deployed to the system.

Model Evaluation

The DMD-LSTM Model was evaluated using the following error metrics:

(a) Mean Squared Error (MSE) - a well-known metric for assessing regression models. It calculates the average of the squared differences between predicted and true values. MSE is useful because it penalizes large errors more severely than small errors, which is important in some applications. A lower MSE indicates that the model performed better (Glen, n.d.-a).

(b) Root Mean Squared Error (RMSE) - another popular metric for evaluating regression models is the RMSE, which is the square root of the MSE. It, like MSE, computes the average of the differences between predicted and true values. Because it is in the same unit as the target variable, RMSE is easier to interpret. A lower RMSE indicates that the model is performing better (Glen, n.d.-b).

(c) Mean Absolute Error (MAE) - it calculates the average of the absolute differences between predicted and true values. MAE is advantageous because it is more resistant to outliers than MSE and RMSE. A lower MAE indicates that the model is performing better (Secret Data Scientist, 2023).

(d) Mean Absolute Percentage Error (MAPE) - it computes the average percentage difference between predicted and true values. MAPE is useful because it provides a relative measure of error, which makes comparing model performance across different target variable scales easier. A lower MAPE indicates that the model is performing better (Allwright, 2022). Furthermore, this is the primary error metric used to select the final model deployed to alamSYS.

Model Cross-Validation

The model selected for the cross-validation was the DMD-LSTM model with a window size of 5, due to it being the highest performing model based on having the lowest MAPE scores compared to the other models, this is further discussed on Chapter 4 of this paper.

Model cross-validation was conducted using the stock market data from the other stocks aside from the training data from PSEI. Where cross-validation of an LSTM model must be done before deployment to assess the model's generalization performance. LSTM models are well-known for their ability to capture long-term dependencies in sequential data, but their performance varies greatly depending on the dataset and model hyperparameters used. And a correctly performed cross-validation helps to provide a more accurate estimate of the model's performance on unseen data, which is critical for ensuring that the model performs well in real-world scenarios (Mellema, 2020; Scherzinger, Roennau, & Dillmann, 2019).

Model Deployment

After determining that the DMD-LSTM model works well with non-training stock market data, it was deployed to the alamSYS as a '.keras' file.

1.2.6 ALMACD Development Diagram

This section discusses the development and integration of the Arnaud Legox Moving Average Convergence and Divergence (ALMACD) indicator as a trading algorithm. The ALMACD indicator tells traders when to buy and sell securities based on the convergence and divergence of two ALMA (fast and slow). Specifically, when the fast ALMA crosses above the slow ALMA, the indicator suggests buying the stock. Conversely, when the fast ALMA crosses below the slow ALMA, the indicator suggests selling the stock.

The process of the development and integration of the ALMACD is shown in Figure 1.21.

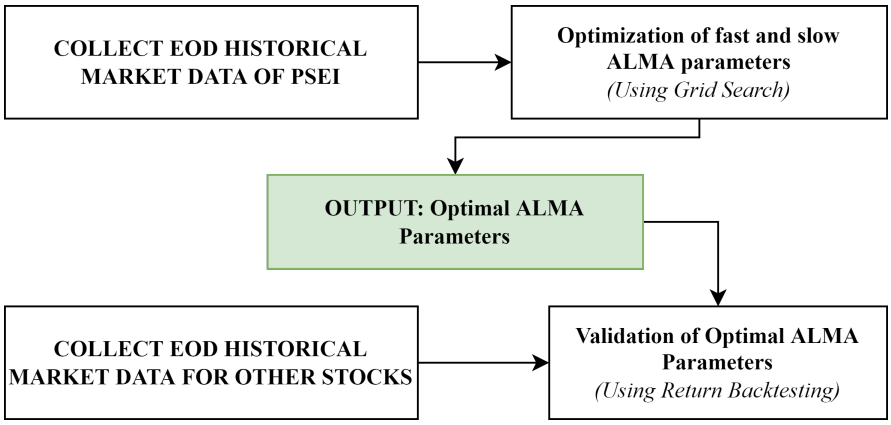

Figure 1.21: ALMACD Development Methodology for alamSYS

Optimization of ALMA Parameters using Grid Search Approach

The ALMACD indicator utilizes two ALMA indicators, one with a fast period and one with a slow period. Where each of them have their own set of optimal parameters, such as their window size, offset, and sigma. In order to find this optimal parameters, a grid search approach was used.

Grid search is a machine learning technique used to fine-tune the hyperparameters of a model. It is a brute force approach that tries every combination of

hyperparameters and evaluates the model performance for each combination. The combination with the best performance is then selected as the optimal hyperparameters (Joseph, 2018). However in the case of the optimization of ALMACD, the grid search approach was utilized to find the optimal parameters for the fast and slow alma, such that it yields a favorable performance by computing the expected return of the trading algorithm. Shown in Table 1.2 are the parameters used for the grid search:

Table 1.2: ALMA Grid Search Parameters

	WINDOW SIZE	**SIGMA**	**OFFSET**
FAST ALMA	1 to 10	1 to 20	0.85, 0.90, and 0.95
SLOW ALMA	10 to 20		

Validation of ALMACD Parameters Using Return Backtesting Approach

Before integrating the optimal ALMACD parameters to the alamSYS, it was first validated using the data from other stocks through backtesting, and calculating the potential returns. Once the returns on the validation stocks were computed to be positive, the ALMACD indicator was then integrated to the alamSYS.

Integration of ALMACD to the alamSYS

The ALMACD indicator was integrated to the alamSYS as a trading algorithm. It was implemented to run after the deep learning model to better identify which stocks to buy and sell.

1.2.7 Docker-Compose Layer Diagram

This section illustrates the different layers of the docker-compose containers based on the way it was used in the deployment of the system.

Figure 1.22 shows a diagram based on Docker documentation, to better understand the containers and layers of the alamSYS. *Note that in the diagram the lowest level is the "Server Infrastructure" and the highest level is the "Docker-Compose" layer.*

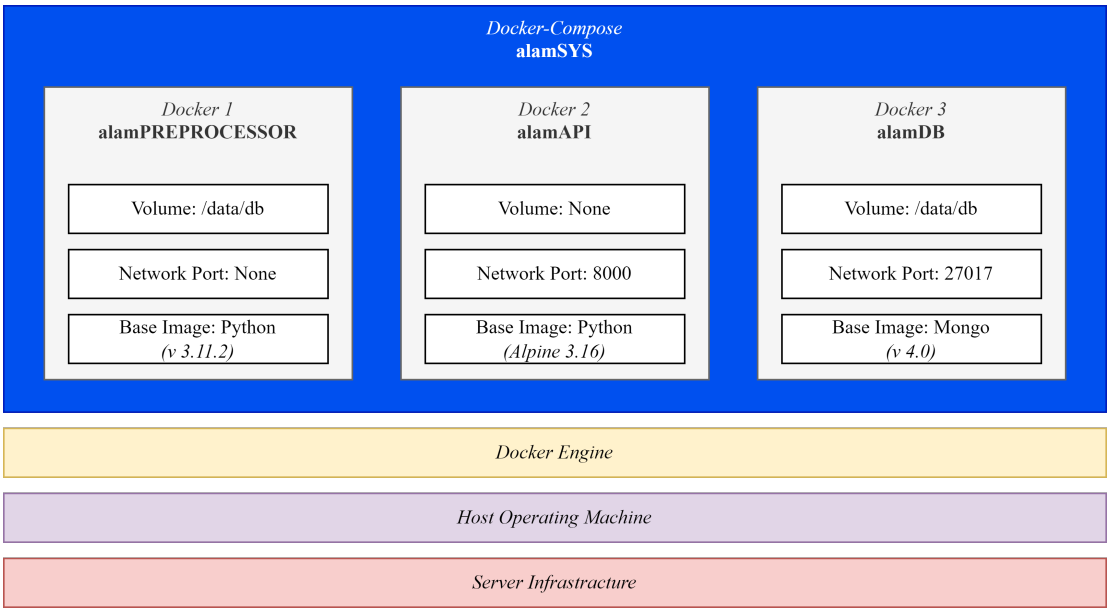

Figure 1.22: Docker-Compose Layer Diagram for the alamSYS

From the figure above, it can be observed that the docker-compose layer is composed of three docker containers, which are as follows: (a) Docker 1 : alamPREPROCESSOR, this is the docker container that contains the alamPREPROCESSOR application. It uses Python 3.11.2 as its based image. Moreover, this container is connected to the data/db volume, and there is no network port exposed; (b) Docker 2 : alamAPI, this is the docker container that contains the alamAPI application. It uses a Python image as well, but is running on top of an Alpine 3.16 image. This was done as Alpine images runs on minimal resources than other Linux based images. Furthermore, it is not attached to any volume, and its network port 8000 is exposed; and (c) Docker 3 : alamDB, this is the docker container that contains the alamDB application which runs on top of the Mongo version 4.0 image. Similarly to alamPREPROCESSOR, it is also connected to the data/db volume, while its network port 27017 is exposed.

1.3 Hardware Specifications

This section discusses the hardware specification utilized in the development of the different components of the alamSYS, as well as other devices used for testing.

1.3.1 For the Development of the alamSYS, Deep Learning Model, and Mobile-Based Test Application

The development of the alamSYS and its components, as well as for the development and testing of the deep learning model and mobile-based test application, utilized a laptop device with the following hardware specifications:

(a) CPU - Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz to 3.10 GHz. *Note that only the CPU was utilized for the development, however the device has a dedicated GPU: NVIDIA GeForce GTX 1050*

(b) Memory - 16GB DDR4 @2666Mhz

1.3.2 For Deployed System Testing

In order to test the deployment capacity of the alamSYS, other computers were used to connect to the server. In this case the aforementioned laptop device was used as a deployment server. Meanwhile listed below are the specifications of the desktop devices used for the deployment testing. *Note that a total of 10 desktop devices were utilized.*

(a) CPU - Intel Core Pentium

(b) Memory - 4GB DDR4 RAM

1.3.3 For the Test Application

Moreover, the mobile device used to showcase the main features of the alamSYS has the following specifications:

(a) CPU - 8-core 3.2GHz (Snapdragon 870 5G)

(b) Memory - 8GB RAM

1.4 Methodology

This section of the Chapter 3 is divided into two sections:

(a) Software Development Process; and

(b) Procedures

1.4.1 Software Development Process

Because of the expected tight time constraints during system development, the author of this paper chose an Agile Software Development Methodology. Agile Sprints were primarily used to manage time efficiently during the software development process (JavaTPoint, n.d.; Milne, 2021). Moreover, a modified Agile Development Methodology as shown in Figure 1.23 was followed in the system development procedures for this special problem.

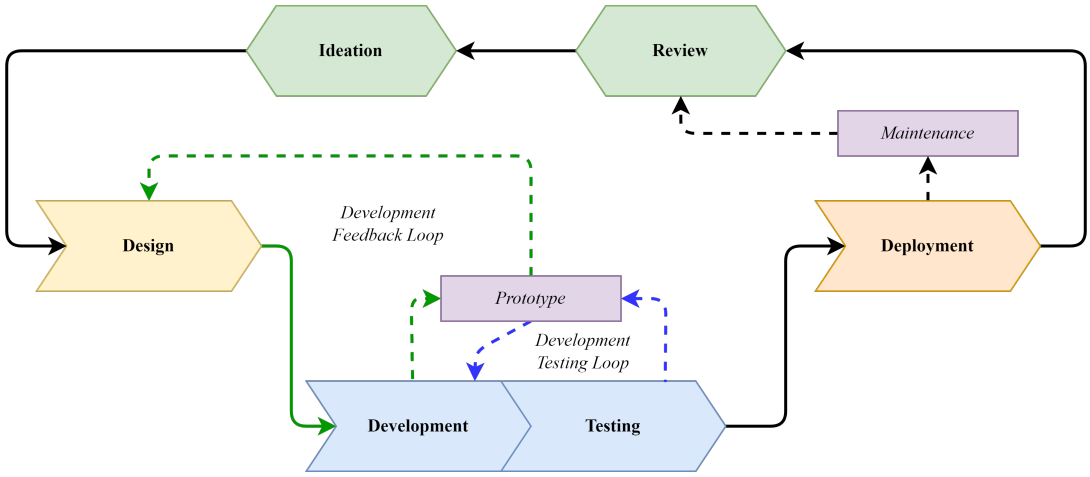

Figure 1.23: Modified Agile Development Methodology

The modified agile development methodology shown above includes additional and specific "feedback loops" such as a development feedback loop that focuses on the continuous design and development process; and a development testing loop that ensures that the software being developed works as intended or better than expected.

The Table 1.3 shows the list of sprints and sub-activities that were followed:

Table 1.3: Summary of Sprints and Activities

Sprint Number	**Activities**	**Completion Date**
1	**Main Activity:** Requirement Analysis, System Planning, and Evaluation **Sub-Activities:** • Topic Proposal • Drafted Chapters 1 to 3 for the Special Problem Proposal • System Architecture and User Requirement Analysis	September 15, 2022 to December 9, 2022

Table 1.3 continued from previous page

Sprint Number	Activities	Completion Date
2	**Main Activity:** Development of Initiasl Sytem Prototype **Sub-Activities:** • Build the different component of the alamSYS as indicated in the top-level overview diagram of the system, the following prototype were initially developed: [1.] API endpoints [2.] Database [3.] Preprocessor • Tested the build prototype.	**12 Weeks** September 30, 2022 to December 23, 2023

Table 1.3 continued from previous page

Sprint Number	Activities	Completion Date
3	**Main Activity:** Development and Testing of DMD-LSTM Model **Sub-Activities:** • Stock Market Data until February 10, 2023 was collected. • Development of the Deep Learning Model. following the procedures as previously discussed on section 1.2.5. • Development of Baseline LSTM model for comparison against DMD-LSTM Model. • Deep Learning model training, testing, cross-validation, and evaluation. • Revised of Chapters 1 to 3.	January 1, 2023 to February 21, 2023

Table 1.3 continued from previous page

Sprint Number	Activities	Completion Date
4	**Main Activity:** Integration of DMD-LSTM Model to the alamSYS and System Testing **Sub-Activities:** • Integrated DMD-LSTM to alamSYS. • Initial tests of the system. • Created System Tests and Loggers. • System Testing, Results Logging and Summarization.	January 15, 2023 to March 7, 2023

Table 1.3 continued from previous page

Sprint Number	Activities	Completion Date
5	**Main Activity:** System Testing, Analysis, Refactoring, and Deployment **Sub-Activities:** • Development test loop was conducted. • System Refactoring. • System Update and inclusion of additionl features. • Results Gathering and Summarization. • Started the development of the test application (for showcasing of the system features).	March 7, 2023 to April 13, 2023

Table 1.3 continued from previous page

Sprint Number	Activities	Completion Date
6	**Main Activity:** Finalization of Paper, System Defense, and Presentation **Sub-Activities:** • Revised Chapter 1 to 3. • Drafted Chapter 4 to 5. • Finalization of the mobile-based test application. • Reviewed the Content of this paper. • Revised and Finalized the special problem paper. • Created presentation slide deck for the presentation of the special problem.	March 14, 2023 to May 12, 2023

From Table 1.3, it was shown that it took a total of 34 weeks (from September 15, 2022, to May 12, 2023) to develop the alamSYS and all its components, as well as to write the contents of this paper. This was 5 weeks better than the expected total weeks allotment for the completion of this whole project, as projected in the Gantt Charts shown in section **??**.

1.4.2 Procedures

This section provides a general overview of the steps conducted in the development of the different components developed in the whole duration of this special problem.

Procedures for the Development of the alamSYS and its Components

The following step-by-step procedures were conducted in the development of the alamSYS and its components (alamAPI, alamDB, and alamPREPROCESSOR):

(a) User Requirement and Analysis

(b) System Design and Architecture using DFD

(c) System Prototyping and Development.

(d) System Testing and Continuous Development and Integration. Each of the components of the alamSYS were developed into smaller chunks, and each chunks were tested and integrated with the other smaller chunks until a component was functionally developed. Afterwards the same process for the other components were conducted, until a whole working system was developed.

(e) System Deployment, Review, and Maintenance

(f) Steps (a) to (e) were repeated until the system was deemed to be stable and ready for deployment within the scopes of this special problem. Moreover, the system shall be maintained even after the submission of this special problem such as the addition of new features.

Procedures for the Development of the DMD-LSTM Model

The following step-by-step procedures were conducted in the development of the DMD-LSTM model:

(a) Collection of end-of-day Philippine Stock Market Data from the 20 selected stocks from the Philippine Stock Exchange (PSE) for the periods until February 10, 2023.

(b) Data Preprocessing, Model Training, Testing, Cross-validation, and Evaluation was conducted based on the DMD-LSTM methodology shown in Figure 1.20.

(c) The best performing DMD-LSTM model was deployed and integrated as part of the alamSYS, specifically used in the alamPREPROCESSOR component.

Procedures for the Development of the ALMACD Trading Algorithm

The following step-by-step procedures were conducted in the development of the ALMACD trading algorithm:

(a) Collection of end-of-day Philippine Stock Market Data from the 20 selected stocks from the Philippine Stock Exchange (PSE) for the periods until February 10, 2023.

(b) Using a Grid Search approach, the best ALMA parameters (*i.e.* window size, sigma, and offset) for both slow and fast were determined.

(c) Using the best ALMA parameters, a trading algorithm was developed which takes in the convergence and divergence of the slow and fast ALMA as basis for entry and exit signals. When the slow ALMA crosses above the fast

ALMA, a buy signal is generated. When the slow ALMA crosses below the fast ALMA, a sell signal is generated.

(d) The trading algorithm was deployed and integrated as part of the alamSYS, specifically used in the alamPREPROCESSOR component.

Procedures for the Development of Mobile-based Test Application

The following step-by-step procedures were conducted in the development of the mobile-based test application:

(a) User Interface Design and Development using Figma.

(b) Creation of Initial Prototype using Flutter.

(c) Continuation of Feature Development and Testing.

(d) Integration of HTTP methods with the alamAPI.

Note that the purpose of this application is to showcase the main feature of the alamSYS, as such it is not fit for user deployment. However it is further discussed in the recommendations on how it can be potentially deployed by future developers.

Procedures for the System Testing

The following step-by-step procedures were conducted in the system testing:

(a) Development of tester application, which are as follows:

1. System Statistics Logger - using the docker stats stream command, the system CPU, memory, and network utilization stats were recorded. This logger was used to determine the idle and on load performance of the alamSYS.
2. Deployment Tester - this tester application was used to test the deployment reliability of the alamSYS. The test was conducted by deploying the alamSYS in a server while then other computers requests for different functions to the alamSYS over the internet using the HTTP methods. Further the tester also logs the overall response time to process 10, 100, and 1000 requests in a row, as well as the success rate of the requests.
3. Internal System Stress Tester - this tester application was used to test the internal system reliability of the alamSYS. The test was conducted to run the processes of the alamSYS in a loop for 100 consecutive times, and logs the overall response time as well as success rate for each processes.

(b) Initial testing of the tester applications, to see if they are working as intended

(c) Actual testing was done using these tester applications, and all results and system logs were recorded.

(d) The results of the testing were then analyzed, compared, and summarized. Further details regarding this are discussed on Chapter 4.

References

Allwright, S. (2022). *What is a good mape score?* Retrieved April 15, 2023, from https://stephenallwright.com/good-mape-score/

Andrew. (2019). *You should (usually) log transform your positive data.* Retrieved April 15, 2023, from https://statmodeling.stat.columbia.edu/2019/08/21/you-should-usually-log-transform-your-positive-data/

Baeldung. (2022). *Normalization inputs for an artificial neural network.* Retrieved April 15, 2023, from https://www.baeldung.com/cs/normalizing-inputs-artificial-neural-network

Dash, R., & Dash, P. K. (2016, 3). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, *2*, 42-57. doi: 10.1016/j.jfds.2016.03.002

Glen, S. (n.d.-a). *Mean squared error: Definition and example.* Retrieved April 15, 2023, from https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/

Glen, S. (n.d.-b). *Rmse: Root mean square error.* Retrieved April 15, 2023, from https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

JavaTPoint. (n.d.). *Agile model.* Retrieved April 16, 2023, from https://www.javatpoint.com/software-engineering-agile-model

Joseph, R. (2018). *Grid search for model tuning.* Retrieved April 17, 2023, from https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e

Mellema, G. R. (2020). Improved active sonar tracking in clutter using integrated feature data. *IEEE Journal of Oceanic Engineering*, *45*(1), 304-318. doi: 10.1109/JOE.2018.2870234

Milne, A. (2021). *The agile development methodology explained.* Retrieved

April 16, 2023, from https://www.netsolutions.com/insights/agile-development-methodology/

Scherzinger, S., Roennau, A., & Dillmann, R. (2019). Contact skill imitation learning for robot-independent assembly programming. *CoRR*, *abs/1908.06272*. Retrieved from http://arxiv.org/abs/1908.06272

Secret Data Scientist. (2023). *What is mae (mean absolute error)?* Retrieved April 15, 2023, from https://secretdatascientist.com/mae-mean-absolute-error/

Tuychiev, B. (2021). *How to differentiate between scaling, normalization, and log transformations.* Retrieved April 15, 2023, from https://towardsdatascience.com/how-to-differentiate-between-scaling-normalization-and-log-transformations-69873d365a94

VisualParadigm. (n.d.). *Gane-sarson data flow diagram tutorial.* Retrieved November 15, 2022, from https://online.visual-paradigm.com/knowledge/software-design/gane-sarson-dfd-tutorial/