Comparative study of different machine learning algorithms for time series forecasting to model stock price movements

[redacted]

April 28, 2021

Abstract

This project evaluates and compares various machine learning algorithms when used for modelling stock prices through windowed predictions. The majority of the literature in this field revolves around single time step predictions in the daily time frame. The main focus of this project is to explore different time frames and window sizes and discover which model performs the best in each scenario. Secondly, to analyse the effect that increasing the prediction window size has on the model's performance. Instead of comparing the models against a simpler machine learning algorithm, a baseline model of shifted closing prices is used. This is an unexplored evaluation method in the literature and allows the models to be compared relative to the baseline. The results show that the baseline model performs the best for 1 time step windowed predictions in all time frames. For higher time step windows in the daily time frame, the ARIMA and LSTM outperform the baseline model. In the hourly time frame, the LSTM almost always outperforms all of the other models. Also, an increase in window size results in less accurate predictions in terms of mean RMSE and MAPE. To conclude, future research will involve using technical indicators and other data to produce signals for change in direction, rather than predicting exact price.

I certify that all material in this dissertation which is not my own work has been identified.

Contents

1 Introduction

There are doubts on whether or not price movements in the financial markets can be predicted. The efficient market hypothesis (EMH) states that all available information is already factored into the current price of a stock, implying that future prices cannot be predicted as information in the future is not available yet [1]. This concept is challenged by technical analysis (TA), which is the study of predicting price movements by identifying patterns in historical price action [2]. Machine learning (ML) algorithms can be used to build models that can make future predictions by being trained on sample data [3]. A large amount of research in this field focuses on single time step predictions in the daily time frame. This paper focuses on comparing various ML algorithms for modelling stock prices across multiple time frames through windowed predictions. This allows for a deeper understanding of how each model will perform in different scenarios. An autoregressive integrated moving average (ARIMA) model will be compared against a long short-term memory (LSTM) neural network. Furthermore, as a baseline comparison for a target dataset, the historical closing prices will be shifted forwards by an amount ws, where ws represents the prediction window size. This will uncover whether or not, with a window size of 1 for example, the model's prediction for today is better or worse than simply using yesterday's price as today's prediction. The models will be trained on historical price action data, technical indicators and news data. Fundamental indicators and social media data will not be covered in this study but will be considered for future research. The research aims of this project are as follows:

- Discover which model performs the best depending on the chosen stock, time frame and prediction window size
- Analyse the effect of increasing the prediction window size on the model's performance

This report will follow a simple structure, starting with a literature review section to analyse the existing research in the field. Then, a design section will outline the overall approach that will be taken for this project, including a diagram of the system that will be developed. A development section will delve into the methodology and implementation of what was discussed in the design section. The methodology for the proposed project will begin by building the required datasets through scraping various application programming interfaces (APIs), namely, Alpha Vantage API [4] and Stock News API [5]. The Alpha Vantage API will be used for historical stock data as well as technical indicator data, whereas the Stock News API will be used for daily sentiment analysis of historical news data of a particular stock. A scraper will be built in C# to generalise the scraping process across multiple APIs while pre-processing and formatting the data into a common format. For reproducibility, the ability to select a custom date range will be implemented to ensure the same data is collected regardless of when the scraping process takes place. Next, exploratory data analysis will be performed on the collected datasets to learn more about the data and uncover any patterns. Python will then be used to develop the rest of the system, starting with any further pre-processing that might be required such as combining or splitting the datasets. Scripts will be developed to allow for the training and testing of the ARIMA and LSTM models. A pipeline system will be adopted, allowing configuration files in the form of JavaScript Object Notation (JSON) to indicate the dataset and hyperparameter combinations to be used in the experiments for optimising the models. An experiment section will be used to present the plan and design of the tests. Afterwards, an interpretation of the results from the experiments will be discussed in an evaluation section. The models will be evaluated by stock, time frame and prediction window size using the mean and median of the root mean square error (RMSE) and mean absolute percentage error (MAPE) across prediction windows. In addition to this, they will be compared against each other as well as the shifted closing price predictions. Finally, an analysis and assessment of the project will be in a critical assessment section, followed by a conclusion to discuss the research aims and the implications of the results produced.

2 Literature Review & Project Specification

2.1 Literature Review

TA is used by traders to identify trends and patterns to predict price movements in the future [6] [7]. Technical indicators are another form of TA and they can be categorised as leading or lagging indicators [6] [7]. Those in the leading category are used to anticipate future price movement based on calculations made in a shorter time frame while lagging indicators are used to determine or confirm a trend after the price action has already started to change direction [7]. Moreover, technical indicators can be further divided into the following types; trend, momentum, volatility and volume [7]. Across various studies, a few popular technical indicators include Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Commodity Channel Index (CCI) and Exponential Moving Averages (EMAs) [8] [9] [10]. These studies also show that the RSI and MACD perform poorly by themselves and that results improve when they are alongside other indicators such as the CCI and EMAs [8] [9] [10].

Fundamental analysis (FA) is used to measure an asset's intrinsic value through analysing related economic factors and determine whether it could be undervalued or overvalued, which is not possible according to the EMH [11]. One study discovers that there is an inverse relationship between the size of the company and the intensity of the price movement with regards to releases of earning reports [12]. The paper also states that when earning reports are released unexpectedly early, it causes a positive price reaction whereas unexpectedly late reports result in a negative price reaction [12].

Sentiment analysis can be performed on news data for a particular stock, to decide if it is positive, neutral or negative [13] [5]. This researcher found a strong correlation between released news articles about a stock, and the price movement of that stock 20 minutes before and 20 minutes after the articles became publicly available [14]. Other papers show that analysing news data through word embedding and deep learning techniques can improve accuracy for stock price predictions and that directional changes in the market can sometimes be predicted more accurately through news data than closing prices [15] [16].

Various researchers have used deep learning (DL) algorithms for solving time series forecasting problems [17] [18]. DL is a subfield of machine learning based on Artificial Neural Networks (ANNs), which are meant to simulate how the human brain functions [19]. DL algorithms have the ability to extract higher-level features from the raw input data [19] [20]. One researcher states that there cannot be one model that is suitable for all problems; one model might outperform another depending on the specific use case [21]. There can be a lot of variation in results when using ANNs because of the number of factors that can affect their performance [22]. Limitations include being black-box methods, meaning it is difficult to explain the relationship between inputs and outputs [22]. In addition to this, ANNs suffer from overfitting and tend to require more data and computation time for training [22].

Recurrent neural networks (RNNs) are a type of ANN that can be used to model sequence data as its outputs depend on previous computations [23]. RNNs are commonly used for temporal forecasting problems because of their natural interpretation of time series data as sequences [24] [25]. Their internal memory state acts as a summary of previous information which results in predictions that are more precise [26]. However, RNNs suffer from the vanishing gradient problem which can worsen memory formation and retrieval [27]. Long short-term memory (LSTM) networks are a type of RNN that solves the vanishing gradient problem by using gates that regulate the flow of information [28]. Gated Recurrent Units (GRUs) are another possible solution; they have a similar performance to LSTMs but are more computationally efficient [29]. In addition to these neural networks, several other models have been used for forecasting stock prices [30] [31]. One paper makes use of a convolutional neural network (CNN) which achieved consistently better results than the baseline algorithms [30]. Support vector machines (SVMs) have been used in a different study to predict the direction of price action and found that it can significantly outperform other models as it is less prone to overfitting [31].

2.2 Project Specification

The hypothesis of this project is to compare the performance of an LSTM network to that of an ARIMA model for modelling stock prices, with a dataset of shifted closing prices being used as a baseline measurement.

The data will be acquired by scraping a variety of APIs, with price action data belonging to the New York Stock Exchange (NYSE). The data can be categorised into technical, fundamental and news data. The Alpha Vantage API [4] will be used for historical price action data as well as technical indicator data. If time permits, fundamental data will also be scraped from this API. For any further technical or fundamental indicators that might not be listed in the Alpha Vantage API, the Twelve Data API [32] will be used. Moreover, the Finnhub API [33] will provide a wider variety of fundamental indicators. In addition to these, sentiment analysis on news data (including articles and videos) will be available through the Stock News API [5]; with the sentiment analysis providing a positive, neutral or negative classification for each article or video. The scraper will be made in C# whereby data will be scraped in a generic format that can be used further down the pipeline, allowing for easy integration with multiple APIs. It will be adapted to allow for reproducibility, with the output being in the format of a comma-separated values (CSV) file. A variety of stocks will be chosen and scraped in the daily and hourly time frames.

Due to time constraints, this project will focus on incorporating an ARIMA model and LSTM network, with the ability to incorporate more models if time permits. Price action data, technical indicators and news data will be prioritised, with fundamental indicators being incorporated as further research. Furthermore, extra computational resources might be required in which the university's ISCA facility could be used, along with other options such as virtual cloud computing services or virtual dedicated servers.

With regards to the design of the project, a scraper will be built in C# to collect data from the previously mentioned APIs. The remainder of the project will be implemented in Python. Various preprocessing functions will be developed for merging datasets, scaling the data, splitting data into training/testing and creating the 3D inputs required for the LSTM network. An ARIMA model will be incorporated, with the ability to optimise its parameters for a given dataset through a grid search. In addition to this, an LSTM neural network will be built as well as a baseline comparison model consisting of shifted closing prices. The input for the LSTM will consist of historical price data, technical indicators and/or news data. It will also consist of fundamental indicators if time permits, as well as other models being incorporated into the system. Hyperparameters will be optimised through performing grid searches in a narrowed search space. A pipeline system will be developed to allow for configuration files in the form of JSON to control the data that will be obtained through the scraper, with any preprocessing that might be necessary, along with further configuration files to setup experiments while defining the hyperparameter search space for the models. Lastly, in order to create a reproducible workflow, the following conventions will be followed [34]. Files and folders pertaining to data and code will be stored separately. Data will be further separated into raw data and processed data, while code will primarily be split into exploration, projects, scripts and functions.

The models will be set up to use a multi-step time series cross-validation mechanism, where the multi-step value represents the window size ws. To evaluate the performance of each window, various forecast error measures will be used. The root mean squared error (RMSE) is a scale-dependent forecast error that will be used to compare experiments performed on the same dataset [35]. The next type of forecast error is the mean absolute percentage error (MAPE), which is not scale-dependent and so it will be used to compare performance across multiple datasets [35]. The mean and median will then be calculated for the forecast errors across all windows. For a deeper evaluation, a box plot of the forecast errors will be produced. Lastly, all models will be compared against a baseline model consisting of closing prices taken from the same dataset and shifted by an amount ws, where ws represents the prediction window size.

3 Design

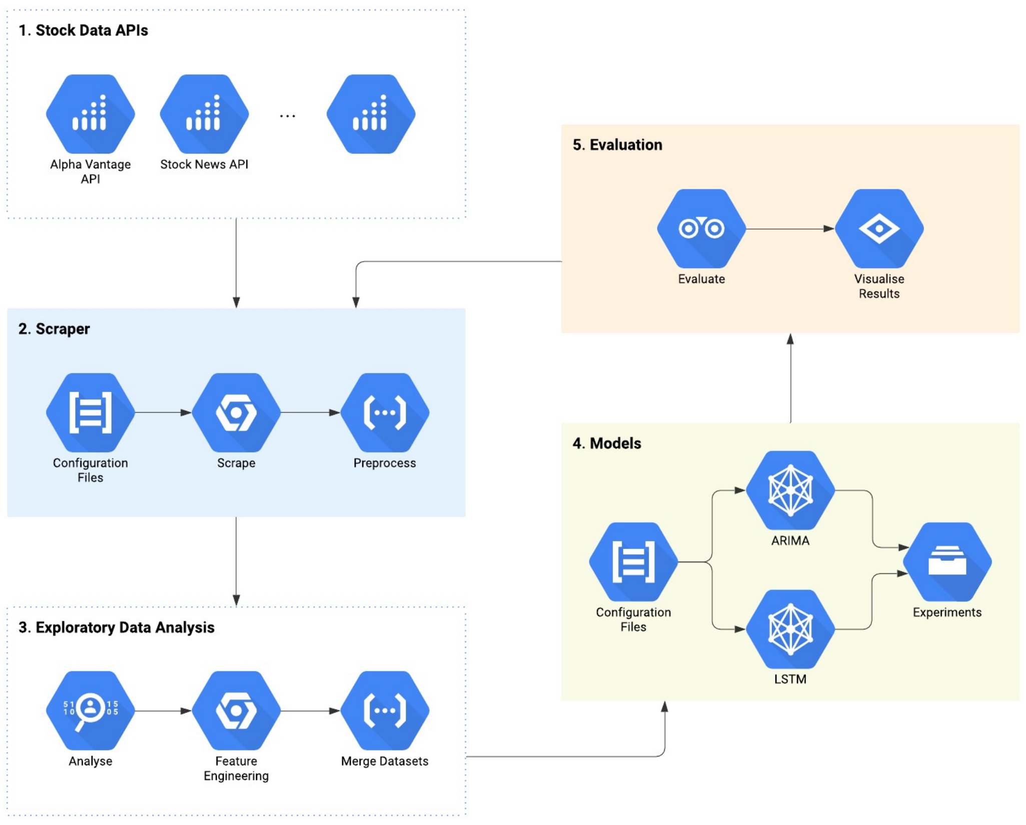


Figure 1: A diagram showing the overall design of the system.

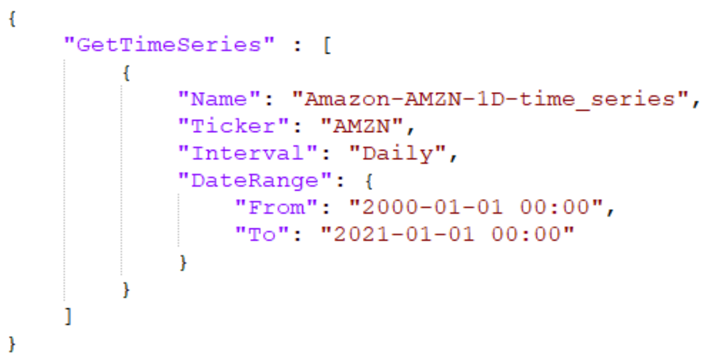
3.1 Stock Data APIs

The system pipeline begins with stage 1 in figure 1; libraries will be built for the APIs if an existing library is not available. The Alpha Vantage API [4] will be developed first due to it containing historical price data and technical indicator data which is the main focus. This API is chosen due to its wide range of historical price data, technical indicators and fundamental data, as well as being free to use. The technical indicators chosen include exponential moving averages (EMAs), stochastic oscillator (STOCH), relative strength index (RSI), moving average convergence divergence (MACD) and on-balance volume (OBV) [6] [7]. They are chosen based on the research in the literature review section as well as their popularity [6] [7]. Following this will be the Stock News API [5] which will be used for news data including sentiment analysis for both articles and videos. The Stock News API [5] is chosen for its simplicity and in-built sentiment analysis.

3.2 Scraper

3.2.1 Configuration

In stage 2, configuration files in the form of JSON will be constructed to indicate which API endpoints to call and what parameters to include, as seen in figure 2. Configuration files will be used for flexibility, and JSON for its simplicity. Additional settings will be required for each API call such as a name to save the dataset as and a date range for the data collection, for reproducibility. This allows for the same data to be collected even when the scraping process is run in the future. The scraper will function similarly to that of a plugin system, so that additional APIs could be added in the future



```
{
    "GetTimeSeries" : [
        {
            "Name": "Amazon-AMZN-1D-time_series",
            "Ticker": "AMZN",
            "Interval": "Daily",
            "DateRange": {
                "From": "2000-01-01 00:00",
                "To": "2021-01-01 00:00"
            }
        }
    ]
}
```

Figure 2: An example scraper configuration file written in JSON for the Alpha Vantage API [4], targeting the Amazon stock on the daily time frame.

seamlessly. This will be accomplished through the scraper making use of interfaces and reflection for interpreting the JSON configuration files as classes and methods to call; this is a dynamic and extendable implementation so that multiple APIs can be used. For instance in figure 2, GetTimeSeries represents the endpoint and the list of objects represent the set of parameters to use. Finally, the data will be preprocessed and organised into CSV files in an output directory specified in the scraper's settings. This is to ensure columns of the same data type such as date are all given the same name rather than multiple column headings such as 'date', 'date_time', and so on. This is important not just for clarity but also for merging datasets in the next stage of the system pipeline.

3.2.2 Architecture

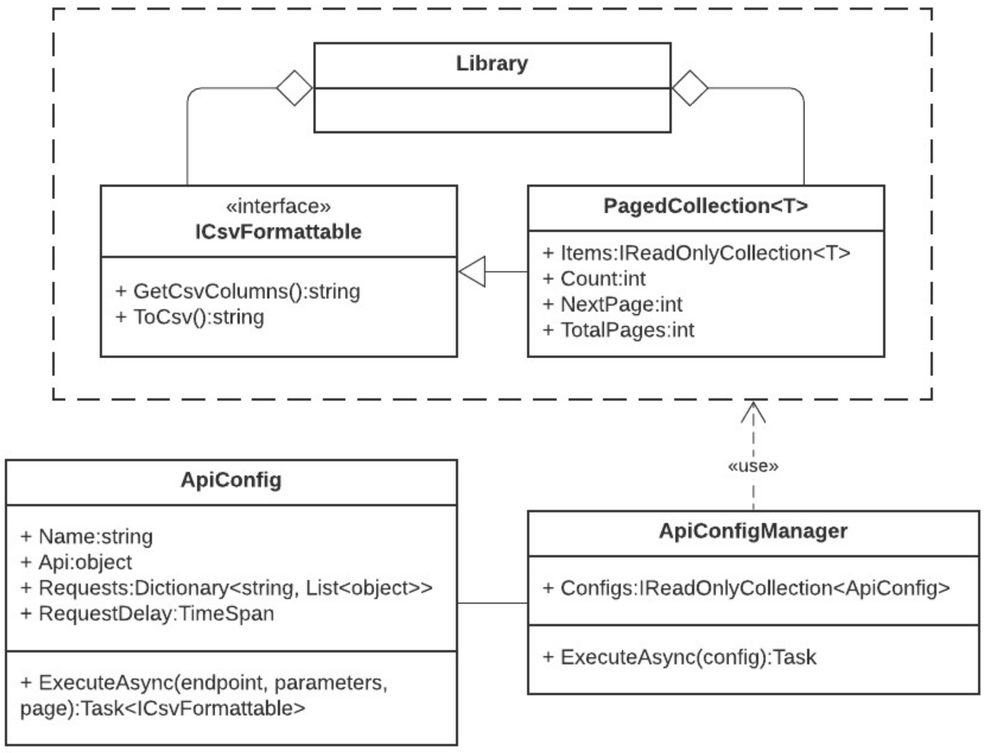


Figure 3: A UML diagram illustrating the design and inner workings of the scraper.

As seen in figure 3, a more in-depth design of the scraper is shown through a UML diagram. Firstly, a library will contain an interface ICsvFormattable which will be inherited by response objects in the API libraries, overriding methods GetCsvColumns() and ToCsv() to allow for serialising to a CSV format. In addition to this, a class PagedCollection<T> also belongs to the library and inherits ICsvFormattable; this should be returned by API calls that consist of paged responses. This means that the library will be able to deal with both single responses and paged responses. The library as a whole will allow for a variety of APIs to be developed and return data in the same, consistent format. The CSV format was chosen due to being compatible with data frames in the pandas package in Python. For the scraper itself, a class ApiConfig will hold the information pertaining to a particular API. The Name property stores the name of the API, used for saving the CSV files in an organised format. The Api property will store the instance of the API class, with the Requests property being used to hold a dictionary of endpoint functions along with a list of parameters for each endpoint. This will allow for JSON configuration files to be used to call multiple parameters for a single endpoint, for example, to scrape historical price data for 5 stocks in different timeframes. Also, a RequestDelay property will be used for any APIs that limit the rate of requests, allowing for the scraper to make a request and then wait x seconds before requesting again. The ApiConfig class will contain a function called ExecuteAsync, taking in the following arguments: endpoint, parameters and page. This will use reflection to call the corresponding API. A traditional plugin framework was not used as different APIs have different endpoints and parameters and therefore cannot easily be tied to an interface, unless the entire scraping process is done separately in each API. For this project, it was therefore preferable to use reflection in the scraper for extendibility, to simplify the development of adding APIs in the future. Finally, the ApiConfigManager class handles each config and therefore the property Configs contains a list of ApiConfigs. The ExecuteAsync function will then run the entire config, calling the target API for all parameters in each endpoint specified in the configuration file.

3.3 Exploratory Data Analysis

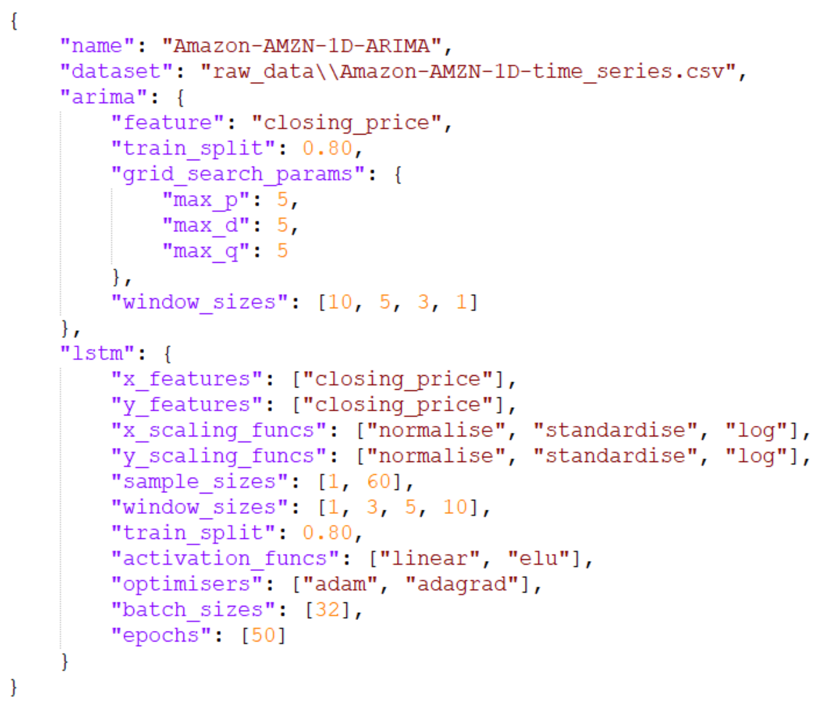
For stage 3, the acquired data will undergo exploratory data analysis, where the aim is to gain a deeper understanding of the data and see if there are any identifiable or noticeable patterns. If so, features will be engineered accordingly for use in the LSTM network and appended as another column to the dataset. In addition to this, data such as technical indicators and news will initially be separated; therefore, datasets could be merged according to any findings during the exploration of the data as well as the research discussed in the literature review section. This stage is crucial for discovering patterns that could be used as additional features to improve price prediction.

3.4 Models

3.4.1 Configuration

After exploratory data analysis comes stage 4, whereby the models will be configured through JSON files to indicate which datasets to use, set additional settings pertaining to the model and define the hyperparameter search space. Once again, JSON is used due to its simplicity. The models will be trained on each hyperparameter combination and the relevant results will be saved in the corresponding folder inside the 'data\results' directory of the project. This is to ensure experiments are kept organised and can be easily searched later on. The results produced will consist of the trained model and the associated hyperparameters that were used in that experiment, along with charts of the predictions and the MAPE of each window for both training and testing datasets. All relevant data is saved so that the same model can be used in the future for further experimentation if desired. In addition to this, another chart of the predictions split by windows will be generated if the window size is greater than 1. This is to visualise the behaviour of the model by window, to see how the predictions vary in accuracy by each time step within a window. Furthermore, for the LSTM model the training vs validation loss will be plotted.

An example configuration file for training the models is shown in figure 4, where JSON will be used for its simplicity. The name parameter specifies the name of the folder to save the results of the experiments, within the 'data\results' directory. The dataset parameter specifies the dataset to use



```json
{
    "name": "Amazon-AMZN-1D-ARIMA",
    "dataset": "raw_data\\Amazon-AMZN-1D-time_series.csv",
    "arima": {
        "feature": "closing_price",
        "train_split": 0.80,
        "grid_search_params": {
            "max_p": 5,
            "max_d": 5,
            "max_q": 5
        },
        "window_sizes": [10, 5, 3, 1]
    },
    "lstm": {
        "x_features": ["closing_price"],
        "y_features": ["closing_price"],
        "x_scaling_funcs": ["normalise", "standardise", "log"],
        "y_scaling_funcs": ["normalise", "standardise", "log"],
        "sample_sizes": [1, 60],
        "window_sizes": [1, 3, 5, 10],
        "train_split": 0.80,
        "activation_funcs": ["linear", "elu"],
        "optimisers": ["adam", "adagrad"],
        "batch_sizes": [32],
        "epochs": [50]
    }
}
```

Figure 4: An example configuration file written in JSON, targeting the Amazon stock on the daily time frame and specifying the hyperparameter search space for both the ARIMA and LSTM models.

for training and testing, and the arima and lstm parameters contain all the hyperparameter settings that will be explored. For ARIMA, the selected feature will be the closing price, with 80% of the data used as training and 20% used for testing. ARIMA has the hyperparameters p, d and q; the parameters max_p, max_d and max_q in the configuration will specify the maximum number to use for their corresponding hyperparameter. All values from 0 to the maximum number will be searched. In this case with the dataset being in the daily time frame, a window size of 5 translates to 1 week due to the stock market being closed on the weekends. For the ARIMA, all values in the window_sizes parameter will be tested. For the LSTM neural network, the x_features parameter specifies the input features from the specified dataset to train the model on, with y_features being the features to predict. In this project, the focus will be varying the input features but keeping the output features as just the closing price. The training and testing split percentage will also be defined through the train_split parameter. In addition to this, all combinations of the remaining parameters in the configuration file for the LSTM will be explored, with each combination being represented as an experiment. The x_scaling_funcs and y_scaling_funcs parameters will represent the scaling function to use for both input and output features respectively. Normalisation, standardisation and the natural logarithm will be the available scaling functions. Moreover, the sample_sizes parameter represents the values for the sample_size in the LSTM input, as shown in figure 5. Furthermore, activation functions, optimisers, batch size and epochs will be tested, along with the window sizes.

3.4.2 LSTM Architecture

Figure 5 shows the design of the LSTM network, adapted from the following paper [36]. The input to the LSTM is 3D, consisting of the input features, sample size or batch size and time steps. The input features will include a combination of historical price data, technical indicators and news data. To improve the performance of the model, overfitting is reduced by adding Dropout layers with 0.2 as the value. Lastly, a Dense layer is used where an activation function will be specified and where the number of output features is specified. The number of input features will vary throughout experiments, whereas the output feature will initially just be the closing price. The design of the system will allow for multiple output features to be specified if desired in the future.

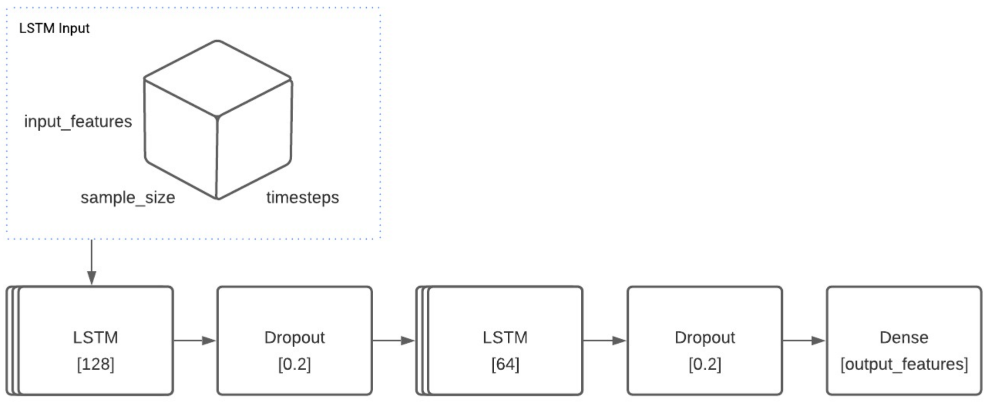


Figure 5: A diagram showing the architecture of the LSTM neural network.

3.5 Evaluation

In stage 5 of the system, evaluation tests will be run for each model's experiments for every configuration file mentioned in stage 4. Evaluation of the experiments consists of taking the mean and median of the RMSE and MAPE across all predicted windows. The RMSE and MAPE were chosen because of their wide use in the machine learning field for measuring forecast errors, with RMSE being scale-dependent [35]. The average forecast error is taken across all windows due to the aims and objectives of this project with regards to evaluating by window size. The median in particular is included due to its robustness against outliers. Experiments will be ordered lowest to highest by mean RMSE and MAPE. This allows for a full overview of all experiments ranked from best to worst. To compare models, comparison tests will be run comparing each model's best performing experiment (the hyperparameters for that model that resulted in the smallest error). To compare configuration files (different time frames, stocks, etc.), these same comparison tests will be used. In addition to this, comparisons will be made against the baseline model which consists of closing prices shifted by an amount ws, where ws represents window size. Finally, if time permits the process can be repeated from stage 2 for additional data to be collected and experimented with such as alternative technical indicators.

4 Development

4.1 Stock Data APIs

The APIs as well as the scraper were developed in C# using the .NET 5.0 Framework. The development process began by creating a dynamic link library (DLL) called 'Scraper.Common' which follows the architecture shown in figure 3. It also included an extension class to allow for arrays to be transformed into a CSV format with quoted values so that they are not taken as literals. This library is referenced by the scraper itself as well as external libraries pertaining to the stock data APIs. For the Alpha Vantage API [4], an existing library found on GitHub was modified [37]. The existing library did not support requests that returned paged responses such as intraday price action data, which was therefore a feature that had to be implemented. This was accomplished through following the library's existing structure and creating a new custom JSON serialiser and modifying a variety of the functions used to handle hypertext transfer protocol (HTTP) requests. Afterwards, a simple wrapper was made around the library to adapt it for use in the scraper. The next library that was developed was the Stock News API [5]. An external library called refit [38] was used to allow for a strongly-typed representational state transfer (REST) API architecture. Furthermore, the library Polly [39] was used for HTTP request error handling, where a jitter retry policy was setup with a wait time of 2^x where x refers to the retry attempt (maximum of 3).

4.2 Scraper

The scraper makes use of dependency injection, following the single responsibility principle (SRP) and simplifying the process of swapping dependencies. In addition to this, a logging system was put in place for any errors that might be encountered during the scraping process. An 'appsettings.json' file is used to configure the scraper, which includes settings for the data folder path and log file path. In addition to this, the options for the various supported APIs can be specified including an API key, the file path of the API's configuration file and a request delay in seconds. The architecture shown in figure 3 was adopted, with classes ApiConfig and ApiConfigManager being implemented. The API's configuration file is a representation of the Requests property in the ApiConfig class. This class also contains a method ExecuteAsync, which selects the function corresponding to the endpoint parameter by loading all available functions in the API. The parameters argument is cast to the type that is taken in as the functions's argument. If the parameter type supports pages, it will use the value from the page argument. The function will then be invoked, calling the API library which executes a HTTP request and returns the response as an ICsvFormattable type. On the other hand, the ApiConfigManager class is in charge of iterating through each request for each endpoint specified in the JSON configuration file for a specific API. Them, the ExecuteAsync method from the ApiConfig class is called for all parameters for each endpoint. The ApiConfigManager class then handles the ICsvFormattable response, depending if it is of type PagedCollection<T> or not, and whether that API requires a delay between each request. Lastly, asynchronous tasks are used in C# to run the scraping process on multiple APIs in parallel. The remainder of the project was developed in Python 3.7.9.

4.3 Preprocessing

Apart from the in-built preprocessing in the scraper, an additional helper class was developed for preprocessing the data prior to the exploratory data analysis phase. The pandas package was used to load the CSV files as a data frame object. The default index was dropped and the datetime feature is converted from a string to a datetime object and set as the index. Moreover, the remainder of the features that are numeric are converted to the float32 type. In addition to this, another function was coded for merging multiple datasets into one based on index, which is to be used in exploratory data analysis if required. Lastly, normalisation, standardisation and the natural logarithm were implemented as scaling functions, as well as their inverse so that the model's predicted values can be transformed back to the original scale. These are to be used for both exploratory data analysis as well as training the models.

4.4 Exploratory Data Analysis

The exploratory data analysis was performed on 5 selected stocks which included Alaska Air Group (ALK), Amazon (AMZN), Chevron (CVX), Nvidia (NVDA) and Walmart (WMT). These choices were based on their difference in market capitalisation, industry, amount of data available, as well as their price action after COVID-19 was declared a pandemic.

4.4.1 Price Action

Four main types of datasets were explored, starting with historical price action data. The features include opening price, lowest price, highest price, closing price and volume. Multiple new features were engineered, starting by taking the natural logarithm of both the closing price and volume to regularise the values. In addition to this, several other columns were created as an attempt to measure volatility to visualise the relationship it might have with price action. These used the equation $(\frac{h_i}{l_{i-v}} - 1) \cdot 100$ where i represents the index of the data point, v being the number of time steps to measure back from, h representing the high price and l representing the low price. These measures were taken with v being 1, 50, 100 and 200. Basic statistics were calculated for the datasets, showing the mean, standard deviation and percentiles to measure the location and dispersion of the data. A heatmap was produced using the seaborn package to show the correlation coefficients between the features. Various line charts were plotted using the matplotlib package to visualise the trend over time and to compare two or more features. Afterwards, scatterplots were generated to show any correlation between the variables. Then, histograms were plotted to show the frequency distribution of the features. Finally, boxplots were used to show the shape of the distribution. All in all, it could be seen that the volatility measures using v as 50, 100, 200 were somewhat positively correlated with the closing price.

4.4.2 Technical Indicators

Indicator	**Settings**
EMA	20
EMA	50
EMA	200
STOCH	5, 3, 3
STOCH	14, 7, 3
RSI	2
RSI	6
RSI	14
RSI	20
MACD	12, 26, 9
OBV	N/A

Table 1: A list of technical indicators that were explored during the exploratory data analysis phase, with their corresponding settings.

A variety of technical indicators were explored, the first being the EMAs with values 20, 50 and 200. Others include the STOCH using settings 5, 3, 3 and 14, 7, 3. In addition to this, the RSI was used with settings 2, 6, 14 and 20. Others that were used include the OBV indicator along with the MACD using settings 12, 26, 9. These technical indicators were merged into one dataset to view the measure of location and dispersion of the data. Line charts were generated for the EMAs to show the crossovers, as well as boxplots for the distribution. The STOCH, RSI and MACD were also visualised through line charts. The OBV also showed positive correlation with the EMAs.

4.4.3 News

Regarding news data, sentiment analysis was taken from previous news articles and videos about a particular stock, using the Stock News API [5]. Various new features were engineered which consisted of cumulative sums of the daily positive, neutral and negative sentiment analysis scores. This means they can be plotted on a line chart to show how the scores change over time. In addition to this, a score was determined based on the cumulative sum of the equation $p - n$, where p represents the amount of positive news and n represents the amount of negative news. Furthermore, histograms and boxplots were produced to show the frequency and spread of the data, in particular the sentiment score.

4.4.4 Combined

Lastly, the different types of data from the previous sub sections were combined to see how they interrelate. Line charts were produced of the closing price and volatility measures from the price action data as well as the news sentiment cumulative score, showing that there is a positive correlation between the features. Furthermore, the EMAs are plotted against the closing price, which showed how the EMA crossovers correlate to price movement, along with the EMAs acting as support.

4.5 Models

4.5.1 ARIMA

The pmdarima package in Python was used for the hyperparameter optimisation of the ARIMA model. Firstly, a class was created to include a variety of functions for training and testing. The datasets need to be split into training and testing, therefore a function is used to simply divide a given dataset by a percentage split. Then, another function was developed to train the model using a grid search of the hyperparameters, with maximum values specified in the parameters. In addition to this, further capabilities were added such as functions to save and load the model from file storage; these used the joblib package. Lastly, a function was developed to output predictions based on a given window size. This used a rolling forecast to iterate through each window and make a prediction for ws time steps, where ws represents the window size. The ARIMA model is then updated with the real values from the test dataset. This continues until it has processed all windows, whereby an array of windowed predictions are returned.

4.5.2 LSTM

The keras package in Python was used for the development of the LSTM neural network, with the numpy package being used for a range of mathematical operations. Firstly, the datasets have to be split into training and testing, in a different format to that of the ARIMA model. This is due to the LSTM requiring a 3D input and the need for the input and output data to be separate. A dataset would need to be split into training and testing, as well as input and output which can be represented as x and y. Therefore, the data is first split into x and y datasets, and then a percentage split is performed on each of them to divide them into training and testing; this results in *x_train*, *y_train*, *x_test* and *y_test*.

This data also needed to be scaled either through normalising, standardising or taking the natural logarithm. A decision was made to scale the data after splitting as it could be argued that future data could be leaked if the scaling was done beforehand, unless the scaling function was the natural logarithm. The reasoning for this is that if all of the data was normalised between 0 and 1, the highest value in the training data could be 0.75, meaning that higher values must exist in the testing data. The natural logarithm is not effected by this as it does not use any other data points in its calculations. In addition to this, a function was created to first scale the training data, and then use the same fitted scale for the testing data. This was another decision that had to be made due to leaking future data once again. In a real life scenario, the future data will be unknown and therefore the scaled output of the LSTM cannot be reversed using a scaler fitted to future data. Taking this into account, the input and output training and testing datasets are scaled accordingly. Afterwards, the data needs to be transformed into batches according to the sample size and prediction window size. The sample size, ss, represents the number of previous data points to use for the next prediction and the window size, ws, represents the number of data points to predict. For example, the price from the last 20 days can be used to predict the price of the next 5 days.

The next phase involves implementing the architecture of the LSTM as shown in figure 5. A function was created to fit the LSTM to the input and output training data, and predict the testing data. Parameters for the activation function, optimiser, batch size and epochs were included to allow for hyperparameter optimisation. The first layer of the LSTM also required an input shape to be specified, consisting of the sample size and number of input features. Therefore, this was dynamically set according to the input training data as these values will change. Finally, the output data had to be transformed and cleaned into windowed predictions. This is because of the way the data was split into batches. The input consists a rolling forecast, moving ahead 1 time step each time but the size of the prediction window could be larger, therefore, multiple predictions might be made for the same day depending on the window size. Finally, a function was developed for plotting any neural network metrics such as loss.

4.6 Evaluation

Various functions were developed for evaluation by measuring forecast errors and plotting predictions. These included the mean, median, root mean squared error (RMSE) and mean absolute percentage error (MAPE), which made use of the sklearn and statistics packages in Python. These were used in other functions to evaluate on a per window basis. In addition to this, more functions were developed for plotting the training vs testing predictions, plotting the predictions split by windows and generating boxplots to visualise the spread of the RMSE and MAPE of the windows. Furthermore, another function was used to return predictions through shifting the closing prices of a given dataset by an amount of ws, where ws represents the prediction window size.

4.7 Overall System

A script was developed to bring together all the features in the system. Firstly, support was added for a list of JSON configuration files to be used. These are read as objects and iterated through one at at time. Then, a function to create the corresponding folders for the experiments was developed, based on the name in the configuration file. The training and testing of the ARIMA model is then initialised, optimising the hyperparameters based on the inputted configuration. Each experiment is contained within a folder generated by the script, with all associated files saved to storage such as the trained model and training and testing data used. In addition to this, all the results for the experiment are saved, including boxplots, plotted predictions and predictions separated by windows. Afterwards, the LSTM model is then initialised to begin training and testing. The settings are taken from the configuration file and used to create experiments. A grid search is used to test all the hyperparameters according to the search space defined in the configuration file. Once all experiments have completed, comparison tests are run to locate the experiment that had outperformed the rest, according to the lowest mean and median windowed RMSE and MAPE. This is used to find the best experiment for a given configuration file. Finally, another script was developed to evaluate and compare the best experiments across configuration files.

One of the issues faced during development was the inability to train for a prolonged period of time, due to the matplotlib package crashing with the following exception; ”fail to allocate bitmap”. This was due to too many figures being opened in memory and therefore it would run out of memory and be unable to allocate enough space to generate new figures. Various methods to dispose of the figures did not seem to function properly for this use case, where the figures are saved to file. The solution that was discovered was to allocate memory for 1 figure and then clear the figure and its axes so that it can be reused.

5 Experiments & Evaluation

The experiments consist of training the ARIMA and LSTM models on the daily and hourly time frames, on the following stocks: Alaska Air Group (ALK), Amazon (AMZN), Chevron (CVX), Nvidia (NVDA) and Walmart (WMT). Predictions will be made in 1, 3, 5 and 10 time step windows. The experiments will be evaluated through the mean of the root mean squared errors (RMSEs) and mean absolute percentage errors (MAPEs) of the predicted windows. In addition to this, the ARIMA and LSTM models will also be compared against the baseline model of shifted closing prices. Furthermore, the difference in performance between the model and the baseline will be taken into account. This will be accomplished through calculating the percentage difference of the RMSE as shown in equation 1, where $RMSE_m$ is the RMSE of the best performing model, and $RMSE_b$ is the RMSE of the baseline.

$$\frac{2 \cdot |RMSE_m - RMSE_b|}{RMSE_m + RMSE_b} \cdot 100 \tag{1}$$

Regarding the first research objective, the following experiments were carried out to discover which model performs the best in different situations such as changing stock, time frame and prediction window size. The first set of experiments focuses on 1, 3, 5 and 10 time step windowed predictions on the daily time frame, with the second set being in the hourly time frame. For the second research aim, the effect of increasing window size on the model's performance will be evaluated. The results shown in the tables below contain the best performing experiment for each model in each of the given situations. Note that the values in bold represent the experiments that resulted in the lowest mean RMSE and MAPE for that particular stock.

5.1 Daily Time Frame

5.1.1 1 Time Step Window

Stock	**ARIMA**		**LSTM**		**Baseline**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	1.1065	2.0021	1.0186	1.828	0.9922	**1.7991**
AMZN	26.248	1.4377	66.1884	3.1748	24.3421	**1.3242**
CVX	1.3666	1.3488	1.2839	1.2663	1.2555	**1.2453**
NVDA	5.2503	2.3106	22.3906	6.4629	4.8329	**2.1128**
WMT	1.0253	0.9330	1.5479	1.359	0.9125	**0.8855**

Table 2: The mean RMSE and MAPE of the 1 time step windowed predictions on the daily time frame for the ARIMA, LSTM and baseline models.

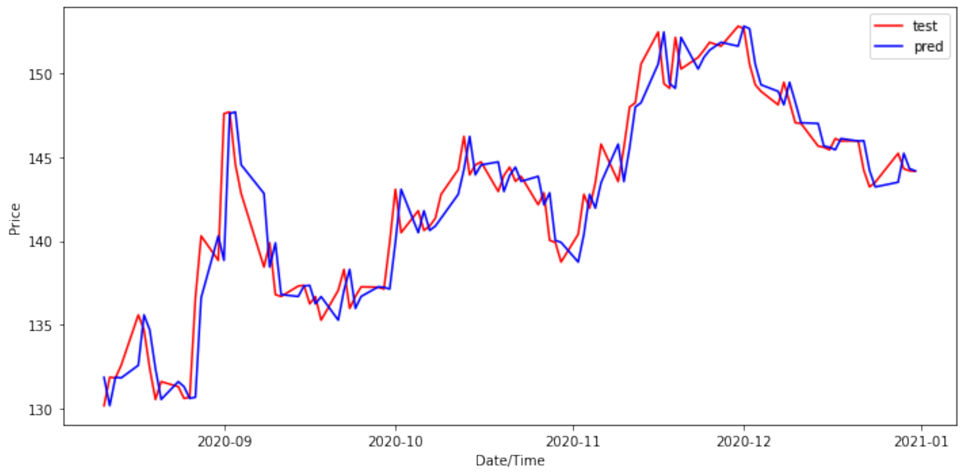


Figure 6: A plot of the real vs predicted values for the WMT stock on the last 100 days of the daily time frame, with a 1 time step prediction window using the baseline model.

This experiment uses a 1 time step windowed prediction in the daily time frame, with the results shown in table 2. To visualise the model's performance, the plotted predictions of the experiment for the baseline model on the WMT stock can be seen in figure 6. From the results in table 2, it is evident that the baseline model outperforms the ARIMA and LSTM across all 5 stocks. This clearly shows that in this study, the baseline model outperforms all other models using a 1 time step prediction window on the daily time frame. Therefore, using the baseline model in this scenario would lead to more accurate predictions.

5.1.2 3 Time Step Window

Stock	**ARIMA**		**LSTM**		**Baseline**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	1.7011	2.7733	1.5414	**2.5028**	1.9729	3.2405
AMZN	40.3534	**1.9798**	122.0217	5.0038	47.5127	2.3461
CVX	2.0821	1.8278	1.914	**1.6959**	2.4242	2.0994
NVDA	7.6547	**3.0581**	33.4747	9.5995	8.9958	3.6058
WMT	1.5824	**1.3701**	1.8803	1.5727	1.731	1.5118

Table 3: The mean RMSE and MAPE of the 3 time step windowed predictions on the daily time frame for the ARIMA, LSTM and baseline models.

Stock	**% Diff**
ALK	**24.56**
AMZN	16.30
CVX	23.52
NVDA	16.11
WMT	8.97

Table 4: The percentage difference rounded to 2 d.p. of the mean RMSE between the best performing model and the baseline model for the 3 time step windowed predictions on the daily time frame.

This experiment uses a 3 time step windowed prediction in the daily time frame, with the results shown in table 3. They show that the baseline is no longer the best performing model. The ARIMA and LSTM both outperform the baseline model under these conditions across all stocks. The most accurate predictions for AMZN, NVDA and WMT are predicted using the ARIMA model. The LSTM then outperforms the ARIMA and baseline model for ALK and CVX. Using equation 1, the percentage difference of the RMSE between the best performing model and the baseline is calculated for each stock; this is shown in table 4. ALK has the largest percentage difference at 24.56%. This shows that the model's predictions for ALK could be more significant than for example, those for WMT with a percentage difference of 8.97%.

5.1.3 5 Time Step Window

Stock	**ARIMA**		**LSTM**		**Baseline**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	2.0736	3.2315	1.8926	**2.9696**	2.5847	4.2223
AMZN	48.263	**2.3342**	138.6537	5.7171	60.3656	2.9662
CVX	2.6128	2.2825	2.3331	**2.0102**	3.2051	2.8025
NVDA	9.7074	**3.7734**	48.4949	14.8624	11.9825	4.7967
WMT	1.8310	**1.5433**	2.0864	1.7421	2.2878	1.9520

Table 5: The mean RMSE and MAPE of the 5 time step windowed predictions on the daily time frame for the ARIMA, LSTM and baseline models.

Stock	% Diff
ALK	30.92
AMZN	22.28
CVX	**31.49**
NVDA	20.98
WMT	22.18

Table 6: The percentage difference rounded to 2 d.p. of the mean RMSE between the best performing model and the baseline model for the 5 time step windowed predictions on the daily time frame.

This experiment uses a 5 time step windowed prediction in the daily time frame, with the results shown in table 5. The ARIMA and LSTM both outperform the baseline model; this can be seen in table 5. The ARIMA performs the best on AMZN, NVDA and WMT, whereas ALK and CVX are best predicted by the LSTM. These results are consistent with those in table 3. However, table 6 shows CVX to have the highest percentage difference at 31.49% whereas before it was ALK. In both experiments the percentage differences for ALK and CVX are in close proximity. In addition to this, the percentage difference has increased across all stocks.

5.1.4 10 Time Step Window

Stock	ARIMA		LSTM		Baseline	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	2.8421	4.5188	2.7163	**4.3209**	3.7375	5.9732
AMZN	66.3242	**3.2307**	175.5557	7.4112	87.1401	4.2404
CVX	3.5747	3.1044	3.0655	**2.5888**	4.6358	4.1021
NVDA	13.5457	**5.4344**	69.9989	22.7549	16.9004	6.9161
WMT	2.4697	2.0860	2.3921	**2.0644**	3.2515	2.7675

Table 7: The mean RMSE and MAPE of the 10 time step windowed predictions on the daily time frame for the ARIMA, LSTM and baseline models.

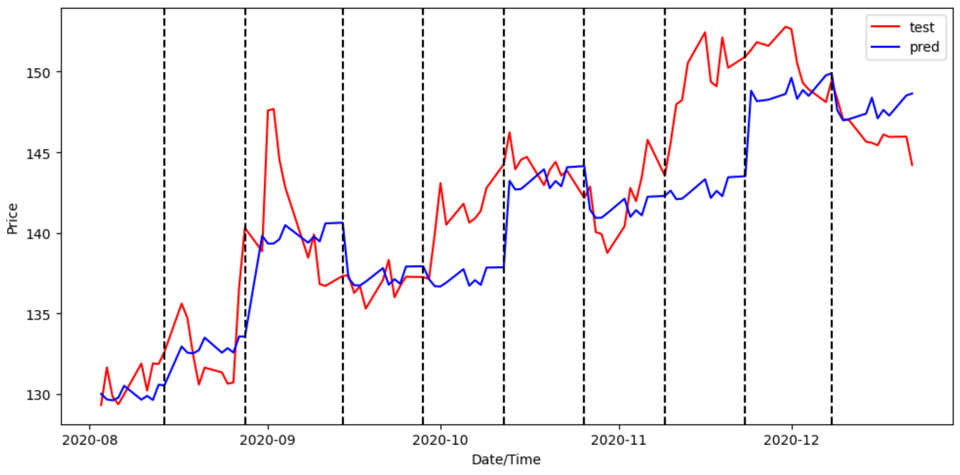


Figure 7: A plot of the real vs predicted values for the WMT stock on the last 100 days of the daily time frame, with a 10 time step prediction window using the LSTM model. Note that each window is separated by a dotted vertical line where the width can vary due to the stock market being closed on weekends.

Stock	**% Diff**
ALK	31.65
AMZN	27.13
CVX	**40.78**
NVDA	22.04
WMT	30.46

Table 8: The percentage difference rounded to 2 d.p. of the mean RMSE between the best performing model and the baseline model for the 10 time step windowed predictions on the daily time frame.

This experiment uses a 10 time step windowed prediction in the daily time frame, with the results shown in table 7. An alternative view of the model's performance can be seen in figure 7 through plotted predictions of the LSTM model on the WMT stock. The results from table 7 show that the ARIMA and LSTM both outperform the baseline model. In this experiment AMZN and NVDA are most accurately predicted by the ARIMA model with ALK, CVX and WMT performing the best on the LSTM model. In addition to this, it can be seen from table 8 that the most significant percentage difference is for CVX, in line with the 5 time step windowed predictions in the daily time frame. Once again, the percentage difference across all stocks has increased.

5.2 Hourly Time Frame

5.2.1 1 Time Step Window

Stock	**ARIMA**		**LSTM**		**Baseline**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	0.3449	0.7914	0.3086	0.7066	0.3016	**0.6935**
AMZN	13.4293	0.422	13.3982	0.4175	11.4773	**0.3605**
CVX	0.3903	0.4988	0.3666	0.4717	0.3365	**0.4305**
NVDA	2.8364	0.5392	3.8425	0.7223	2.449	**0.4655**
WMT	0.4248	0.2978	0.4746	0.3309	0.3716	**0.2606**

Table 9: The mean RMSE and MAPE of the 1 time step windowed predictions on the hourly time frame for the ARIMA, LSTM and baseline models.

This experiment uses a 1 time step windowed prediction in the hourly time frame, with the results shown in table 9. The results are in line with those of the 1 time step windowed prediction in the daily time frame, where the baseline model outperformed the others. This shows that the ARIMA and LSTM make worse predictions using a 1 time step windowed prediction in the daily as well as hourly time frames.

5.2.2 3 Time Step Window

Stock	**ARIMA**		**LSTM**		**Baseline**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	0.5155	1.0584	0.4581	**0.9337**	0.5771	1.1605
AMZN	21.5991	0.6053	21.0226	**0.5981**	24.6689	0.6898
CVX	0.6211	0.7123	0.5582	**0.6419**	0.6991	0.7924
NVDA	4.4836	**0.7628**	6.6324	1.1392	5.1200	0.8637
WMT	0.6245	0.3925	0.5817	**0.3641**	0.6992	0.4340

Table 10: The mean RMSE and MAPE of the 3 time step windowed predictions on the hourly time frame for the ARIMA, LSTM and baseline models.

Stock	**% Diff**
ALK	**22.99**
AMZN	15.96
CVX	22.41
NVDA	13.25
WMT	18.35

Table 11: The percentage difference rounded to 2 d.p. of the mean RMSE between the best performing model and the baseline model for the 3 time step windowed predictions on the hourly time frame.

This experiment uses a 3 time step windowed prediction in the hourly time frame, with the results shown in table 10. The baseline is no longer the best performing model. Predictions made using the ARIMA model are best suited for the NVDA stock, whereas ALK, AMZN, CVX and WMT are most accurately predicted by LSTM. It is important to note that the LSTM has mostly outperformed the other models in this scenario. Furthermore, the percentage differences shown in table 11 show that ALK holds the most significant percentage difference.

5.2.3 5 Time Step Window

Stock	**ARIMA**		**LSTM**		**Baseline**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	0.6546	1.3500	0.5491	**1.0641**	0.7480	1.5266
AMZN	26.8178	0.7252	24.6427	**0.6749**	32.6843	0.9084
CVX	0.8055	0.8999	0.7107	**0.7974**	0.9555	1.0769
NVDA	5.6093	**0.9285**	5.5504	0.9351	6.7311	1.1471
WMT	0.7348	0.4528	0.7094	**0.4336**	0.8790	0.5376

Table 12: The mean RMSE and MAPE of the 5 time step windowed predictions on the hourly time frame for the ARIMA, LSTM and baseline models.

Stock	% Diff
ALK	**30.67**
AMZN	28.06
CVX	29.38
NVDA	18.18
WMT	21.35

Table 13: The percentage difference rounded to 2 d.p. of the mean RMSE between the best performing model and the baseline model for the 5 time step windowed predictions on the hourly time frame.

This experiment uses a 5 time step windowed prediction in the hourly time frame, with the results shown in table 12. The LSTM has outperformed all other models for ALK, AMZN, CVX and WMT. NVDA is then best predicted by the ARIMA model. These results are similar to that of the 3 time step windowed predictions in the hourly time frame. Furthermore, the percentage differences in table 13 show ALK to hold the most significant result. Moreover, the percentage differences have increased throughout all stocks from the previous experiment of 3 time step windowed predictions in the hourly time frame. This is a common occurrence throughout time frames as this was also observed in the daily time frame.

5.2.4 10 Time Step Window

Stock	ARIMA		LSTM		Baseline	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ALK	0.8821	1.7455	0.7351	**1.4244**	1.0873	2.1679
AMZN	37.4387	0.9972	35.1002	**0.9479**	47.8818	1.3286
CVX	1.1418	1.2559	1.0469	**1.1562**	1.4380	1.6036
NVDA	8.1089	1.3368	7.5439	**1.2516**	9.6926	1.6345
WMT	0.9458	**0.5758**	0.9747	0.5854	1.2052	0.7296

Table 14: The mean RMSE and MAPE of the 10 time step windowed predictions on the hourly time frame for the ARIMA, LSTM and baseline models.

Stock	% Diff
ALK	**38.65**
AMZN	30.81
CVX	31.48
NVDA	24.93
WMT	23.12

Table 15: The percentage difference rounded to 2 d.p. of the mean RMSE between the best performing model and the baseline model for the 10 time step windowed predictions on the hourly time frame.

This experiment uses a 10 time step windowed prediction in the hourly time frame, with the results shown in table 14. These show that the stocks ALK, AMZN, CVX and NVDA performed the best using the LSTM model, with WMT being predicted more accurately by the ARIMA model. Considering the percentage differences in table 15, it can be seen that ALK has the highest percentage difference. The percentage differences have yet again increased across all stocks.

5.3 Overview

All in all, a model's performance can widely vary depending on the given situation. With regards to the first research aim, it was found that for 1 time step windowed predictions in both the daily and hourly time frames, the baseline model always outperforms the ARIMA and LSTM. For 3, 5 and 10 time step windowed predictions, the baseline model is no longer the most accurate. In the daily time frame, the best performing model is the ARIMA or LSTM depending on the stock. In both time frames, stocks ALK and CVX achieved the highest percentage difference from the baseline model. It is important to note that ALK and CVX have the smallest market capitalisation out of the 5 stocks and both experienced a drop in price after COVID-19 was declared a pandemic. For the hourly time frame, the LSTM was the best performing model the majority of the time. This supports the fact that the performance of neural networks increases with more data. Considering the second research objective regarding the effect of window size on the model's performance; in both time frames, the mean RMSE and MAPE worsens as the window size is increased across all stocks and therefore the model's accuracy decreases. However, the percentage difference between the baseline model increases as the window size is increased.

6 Critical Assessment

Overall, the project introduces a couple of concepts that have not been heavily explored in the field; windowed predictions as well as evaluating against shifted closing prices. Windowed predictions allow for a new way that the model's predictions can be viewed and its performance can be evaluated. The shifted closing prices provide a stronger baseline for the models to be compared against. However, there are multiple improvements that could be made to this project. The first limitation is the amount of data available for the lower time frames, as these only date back to the last 2 years. Aside from this, there should be further exploratory data analysis of the technical indicator data; to learn and understand additional technical indicators and how each one is used in trading so that signals can be plotted for changes in direction. Then, multiple indicators can be used simultaneously to see if they agree with each other; this would lead to stronger signals. More features can then be engineered based on the findings. Another improvement that can be made to this project is further training of the models. There are a large number of hyperparameters and input features that can be tested. Not to mention that Bayesian Optimization should be used to optimise the hyperparameters, instead of using a grid search. The structure and architecture of the LSTM should also be explored. Furthermore, a wider variety of different models should be tested apart from ARIMA and LSTM. Finally, statistical tests should be performed to solidify the evaluation of the project.

7 Future Research

For future research, the 15 minute as well as other lower time frames will be explored. Further exploratory data analysis will be performed on price action, technical indicator and news data for feature engineering. In addition to this, other types of data such as fundamental indicators and social media data will be explored. There are a wide variety of social media platforms such as Twitter and Reddit that could be beneficial towards predicting stock price. In addition to these, there are various other communities more focused on trading and investing such as Stocktwits which could also be of interest. A larger number of models will also be evaluated including GRUs, CNNs and SVMs to be compared against the current ones. In addition to this, producing buy and sell signals to represent changes in direction will also be explored, as this could be more beneficial than predicting exact price values.

8 Conclusion

Throughout the literature in this field, it is common for studies to use single time step predictions and to evaluate their models against a simpler machine learning model. This project's main focus is on windowed predictions to gain a deeper understanding of the model's performance. More importantly, this study uses a baseline model of shifted closing prices for evaluation and comparison against other models. This arguably provides a better outlook on the model's performance as the model's predictions should be more accurate than shifted closing prices. Therefore, the performance of the models can be compared relative to the baseline. Considering the first research objective to discover which model performs the best in given situations, the baseline model outperforms the others on 1 time step windowed predictions in all time frames. Then across the daily time frame for 3, 5 and 10 time step windowed predictions, the stocks ALK and CVX are on average better predicted by the LSTM, whereas the ARIMA's predictions are more accurate for AMZN, NVDA and WMT. For the hourly time frame using 3, 5 and 10 time step windowed predictions, the LSTM generally performed better for the stocks ALK, AMZN, CVX and WMT, with the ARIMA having the most accurate predictions for NVDA on average. Regarding the second research aim, the mean RMSE and MAPE is increasingly worse as the window size is increased for all models. However, the percentage differences between the models and the baseline improve as the window size increases. Out of the 5 selected stocks, ALX and CVX had the highest percentage difference. They also have the smallest market capitalisation out of the chosen stocks and both suffered a drop in price after COVID-19 was declared a pandemic. Future research will include further optimising the LSTM inputs through feature engineering as well as testing other models such as GRUs, CNNs and SVMs. Altogether, this project builds on the existing literature showing that for 1 time step windowed predictions on all time frames, the baseline outperforms the rest and therefore the models may not be beneficial in this scenario. However, there are other situations where the model is significantly more accurate than the baseline; for instance, the 3, 5 and 10 time step windowed predictions.

References

[1] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970. [Online]. Available: http://www.jstor.org/stable/2325486

[2] J. Chen, "Technical analysis of stocks and trends," Mar 2021. [Online]. Available: https://www.investopedia.com/terms/t/technical-analysis-of-stocks-and-trends.asp

[3] "Machine learning," Apr 2021. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[4] "Alpha vantage." [Online]. Available: https://www.alphavantage.co/

[5] "Stock news api." [Online]. Available: https://stocknewsapi.com/

[6] R. W. Colby, *The Encyclopedia of Technical Market Indicators.* McGraw-Hill, 2003.

[7] S. B. Achelis, *Technical Analysis from A to Z.* Equis International, 2003.

[8] M. Dempster and C. Jones, "The profitability of intra-day fx trading using technical indicators," 01 2000.

[9] W.-K. Wong, M. Manzur, and B.-K. Chew, "How rewarding is technical analysis? evidence from singapore stock market," *Applied Financial Economics*, vol. 13, no. 7, pp. 543–551, Jan. 2003. [Online]. Available: https://doi.org/10.1080/0960310022000020906

[10] M. Tanaka-Yamawaki and S. Tokuoka, "Adaptive use of technical indicators for the prediction of intra-day stock prices," *Physica A: Statistical Mechanics and its Applications*, vol. 383, no. 1, pp. 125–133, Sep. 2007. [Online]. Available: https://doi.org/10.1016/j.physa.2007.04.126

[11] Fundamental analysis definition. [Online]. Available: https://www.investopedia.com/terms/f/fundamentalanalysis.asp

[12] A. E. Chambers and S. H. Penman, "Timeliness of reporting and the stock price reaction to earnings announcements," *Journal of Accounting Research*, vol. 22, no. 1, p. 21, 1984. [Online]. Available: https://doi.org/10.2307/2490700

[13] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng, "News impact on stock price return via sentiment analysis," *Knowledge-Based Systems*, vol. 69, pp. 14–23, Oct. 2014. [Online]. Available: https://doi.org/10.1016/j.knosys.2014.04.022

[14] G. Gidófalvi, "Using news articles to predict stock price movements," 2001.

[15] Y. Peng and H. Jiang, "Leverage financial news to predict stock price movements using word embeddings and deep neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, 2016. [Online]. Available: https://doi.org/10.18653/v1/n16-1041

[16] A. Atkins, M. Niranjan, and E. Gerding, "Financial news predicts stock market volatility better than close price," *The Journal of Finance and Data Science*, vol. 4, no. 2, pp. 120–137, Jun. 2018. [Online]. Available: https://doi.org/10.1016/j.jfds.2018.02.002

[17] P. Yu and X. Yan, "Stock price prediction based on deep neural networks," *Neural Computing and Applications*, vol. 32, no. 6, pp. 1609–1628, Apr. 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04212-x

[18] Y. Zhai, A. Hsu, and S. K. Halgamuge, "Combining news and technical indicators in daily stock price trends prediction," in *Advances in Neural Networks – ISNN 2007.* Springer Berlin Heidelberg, pp. 1087–1096. [Online]. Available: https://doi.org/10.1007/978-3-540-72395-0_132

[19] A. G. Josh Patterson, *Deep Learning - A Practitioner's Approach.* O'Reilly, 2017.

[20] Y. Bengio, G. Guyon, V. Dror, G. Lemaire, D. Taylor, and D. Silver, "Deep learning of representations for unsupervised and transfer learning," vol. 7, 01 2011.

[21] R. O. Bruce L. Bowerman, *Forecasting and Time Series: An Applied Approach, 3rd ed.* Duxbury Press, 1993.

[22] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, Mar. 1998. [Online]. Available: https://doi.org/10.1016/s0169-2070(97)00044-7

[23] I. Sutskever, "Training recurrent neural networks," Ph.D. dissertation, CAN, 2013.

[24] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, Jul. 2020. [Online]. Available: https://doi.org/10.1016/j.ijforecast.2019.07.001

[25] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 7785–7794. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf

[26] B. Lim and S. Zohren, "Time series forecasting with deep learning: A survey," *ArXiv*, vol. abs/2004.13408, 2020.

[27] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain," *Current Opinion in Neurobiology*, vol. 55, pp. 82–89, Apr. 2019. [Online]. Available: https://doi.org/10.1016/j.conb.2019.01.011

[28] A. Graves, "Generating sequences with recurrent neural networks," 08 2013.

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 12 2014.

[30] E. Hoseinzade and S. Haratizadeh, "CNNpred: CNN-based stock market prediction using a diverse set of variables," *Expert Systems with Applications*, vol. 129, pp. 273–285, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.03.029

[31] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," *Computers & Operations Research*, vol. 32, no. 10, pp. 2513–2522, Oct. 2005. [Online]. Available: https://doi.org/10.1016/j.cor.2004.03.016

[32] Twelve data. [Online]. Available: https://twelvedata.com/

[33] Finnhub. [Online]. Available: https://finnhub.io/

[34] Reproducibility workflow. [Online]. Available: https://github.com/rlbarter/reproducibility-workflow

[35] Evaluating forecast accuracy. [Online]. Available: https://otexts.com/fpp2/accuracy.html

[36] M. Roondiwala, H. Patel, and S. Varma, "Predicting stock prices using lstm," 2017.

[37] Kirlut, "kirlut/alphavantage.net." [Online]. Available: https://github.com/kirlut/AlphaVantage.Net

[38] Reactiveui, "reactiveui/refit." [Online]. Available: https://github.com/reactiveui/refit

[39] App-vNext, "App-vnext/polly." [Online]. Available: https://github.com/App-vNext/Polly