# Data Science Mini Project

## HSBC Global Markets

Amy Gardiner, Brooke Grantham, Michael McCoubrey, Aditya Singh

***Abstract*—Two data analysis pipelines were executed in order to transform four months of raw asset data, in the form of daily LOB and tape files, into quantifiably profitable systems based on modelled price predictions. A2C reinforcement learning, trained on derived daily tape features, learnt policies of buy and sell actions for datasets 1,2 and 3 with total reward and profit determined to be (18,1.013), (0,0.861) and (94,1.479) respectively. Big data storage and processing solutions allowed for manipulation such to combine the LOB and tape data. Using this, random forest regression was implemented in order to provide price predictions for future trading. These formed the input of the decision making program that determined buy, hold and sell actions based on the spread of their mean absolute error. Trading on these actions returned a total profit of 2.238. All code can be found in the hsbcminiproj GitHub repository.**

## I. Introduction

Predicting the financial market is one of the most prominent and interesting problems for a data scientist today, with profit yielded from automated trading algorithms still much to be improved upon. Throughout the day a trader will post their price per share with the quantity of shares, known as a bid limit order. Conversely for an ask limit order, the seller will post their sale price per share and the quantity. Multiple times a second the limit orders are posted, giving an estimate of 50 million data points for any given day. A day of limit orders for an asset is the a limit order book (LOB), also known as Level-2 data, which comprises all the current quantities of units to buy or sell with the corresponding prices. The LOB data is thought to be an indicator into whether the stock price will increase or decrease, however this requires data preprocessing and a complex algorithm to make the predictions and conduct trades. Along with the LOB data there is the tape data which contains every trade including price, quantity and parties involved.

There are many approaches to creating stock predictions including time-series regression, deep learning and reinforcement learning models. A key factor is extracting features from the data such as moving averages, quantities and the spread of bids and asks. Exploring the best combination of models with these features can generate price predictions and buy/sell actions accurate enough for a working automatic trading system.

## II. Literature Review

Univariate models such as Neural Prophet and ARIMA are very well known and "can be employed to represent seasonal time series" [1]. However these are usually less performant than multivariate models. The ARIMA model could hence be used as a baseline model to compare the final data model's performance against.

Like univariate models, the ideal model would also consider the current price of the asset but, because this model is multivariate, technical indicators can be added to assist prediction. Various technical indicators were defined in [2] which allowed for the implementation of indicators such as the exponential moving average.

While the current price and the associated technical indicators will be useful for predicting performance, marketable orders (which these variables are based on) only represent 10% of all orders that reach the exchange [3]. Hence to consider only the price and technical indicators would mean much of the available data would not be utilised.

LOBs contain "bid limit orders" and "ask limit orders" - data which has been of academic interest as this trading activity influences how asset prices change. Spread and liquidity indicators can summarise the LOB at a given time. LOB indicators, described in [3], [4] and [5] were added to the dataset in order to help the model's performance.

Any data model that is used in seasonal time series should consider seasonal patterns so that the data model stays accurate throughout the year. For this reason, features were added to represent the seasonality effects discussed in [2].

Many machine learning models such as neural networks and support vector machines are known to outperform benchmarks, but the models fail to generalise to the noisy and dimensionally complex stock market data [2]. Due to this overfitting, these strategies excessively buy and sell in a short amount of time, which due to transaction costs can be expensive. These models that are overfitted also have large drawdowns (erratic profit). Fortunately overfitting has been proven to not be a problem for random forests [2].

As random forests are an ensemble network of decision trees, when more recent data becomes available decision trees can be generated and added to the random forest in an online fashion. This allows the model to adapt to market conditions, such as seasonality effects [6].

Random forests models are an example of homogeneous ensemble network or decision trees. Heterogeneous ensemble networks that use various types of underlying models (e.g. neural networks, decision trees, and logistic regression) offer better performance than homogeneous ensemble networks [2]. For this reason it was planned to design a heterogeneous ensemble network that performed well and did not suffer from overfitting.

## III. Methodology

### A. Univariate Modelling

*1) ARIMA:* In order to explore the time series data involving only the price of the asset, auto-regressive integrated moving-average (ARIMA) models were first considered. These models are beneficial within stock data analysis due to their ability to both handle non-stationary data and provide forecasting - which has the potential for great commercial value.

ARIMA models require 3 hyperparameters: the differencing term d, the auto-regressive term p and the moving average term q. The amount of differencing, d, allows for stationarity in the data since the linear regression aspect of the model requires each data point to be uncorrelated. The auto-regressive term, p, measures of the number of lag points to be considered within the model, and the moving average term, q, is a similar measure to p but with the lagged forecast errors [1].

*2) Neural Prophet:* Neural Prophet is a time-series forecaster which makes use of deep auto-regressive neural networks and is optimized using Gradient Descent. Neural Prophet has the ability to accurately forecast longer term time-series data due to the seasonality recognition. The seasonality is calculated using the Fourier Series and can detect periods such as yearly, daily, weekly and hourly. [7]

The forecasting is determined by 3 main hyperparameters; number of lags, number of forecasts and the number of changepoints. The number of lags and forecasts respectively dictate the quantity of time to use from the past to make the future forecasts and how many datapoints into the future to project. The number of changepoints describes how many times the data shifts direction.

### B. Random Forest

The random forest model is a multivariate technique based on decision trees. Decision trees are known for their high variance and low bias, whereas the random forest algorithm applies bootstrap aggregating (bagging). Bagging involves random sampling of the training data, allowing the uncertainty in the predictions to be measured, thus reducing variance. It is hence this bagged ensemble of decision trees that is known as the random forest [8]. Random forest hyperparameters include the number of trees (n_estimators in Sklearn), the maximum depth of the tree and the minimum number of samples required within an internal node before it can split. Tuning these hyperparameters can affect the performance of the regressor in many ways, from the increase in computation time seen with increasing the tree number, to overfitting as seen with a maximum depth value over the optimal, to underfitting by increasing the minimum number of samples, as less parameters are required.

### C. Reinforcement Learning

Reinforcement learning (RL) algorithms connect agents with their environments through the exploration of actions and reward. RL agents learn to map the sets of environmental states $S$ and actions $A$ in order to maximise a certain parameter or reward using the best learned policy $\pi$. This policy can be determined through value or policy iteration - the former being the sum of rewards in the case of a given policy being followed. [9]

*1) Advantage Actor-Critic Method (A2C):* The A2C algorithm is considered hybrid, as it combines value optimization with policy optimization [10]. A2C was implemented using the AnyTrading environments, a subset of the OpenAI Gym toolkit, with its corresponding Tensorflow dependency. While many actions exist and prove useful within the context of trading (buy/sell/hold/enter/exit), including all of these for the development of a RL agent had been found to be computationally/time intensive and far less significant than those utilising a reduced action space of buy and sell only. Hence the 'stocks-v0' AnyTrading environment implemented here contains such an action space [11].

### D. Custom Model

*1) Regression-based Decision Making:* The aim of the model is to make a profit from buying or selling assets but the underlying random forest model only predicts the current price. Therefore a model is needed to decide when to buy or sell the asset. To transform the regression output of the random forest to a transaction decision requires an appropriate strategy.

It was assumed that the model has insight of what the price should be given the asset's characteristics. Given that assumption, if the predicted price is drastically higher or lower than the actual price then this can be identified as a trading signal. If the model recognises this difference as a trading signal then an appropriate transaction decision will be made to capitalise on this opportunity.

The model needs to be given a threshold to decide whether to trade based on the variation in prediction price to the true price.The threshold to buy or sell can be adjusted but will increase the risk. If the threshold is set low, the model will make many transactions with lower likelihood of correct decisions. But if the threshold is set high, the model will make few transactions with increased likelihood of correct decisions, potentially not capitalizing for maximum profit.

The algorithm's threshold is defined by the z-score of the model; for example setting the threshold to be a z-score of 1 should mean that the model makes transactions 32% of the time. The z-score can be defined as the number of standard deviations from the mean of which to define the bounds of the threshold.

As seen in A. Booth et al. [2], models can overfit the data and make many transactions in a short amount of time which results in large drawdowns and high transaction costs. The model designed is tested for an overfit on the data to reduce this undesirable behaviour. To reduce the overfitting, the z-score threshold is set to 4 therefore only making transactions on 0.0064% percent of the time. Ultimately, the model makes decisions 1.8771% of the time suggesting the model is finding outliers and signals in the data.

Below is the equation for deciding what transaction decision the model should make:

$$F(z) = \begin{cases} buy & \hat{p} > p + \mu + (z \times \sigma) \\ hold & p + \mu - (z \times \sigma) < \hat{p} < p + \mu + (z \times \sigma) \\ sell & \hat{p} < p + \mu - (z \times \sigma) \end{cases} \quad (1)$$

where $\hat{p}$ is the predicted price, $p$ is the true price, $\mu$ is the mean error of the model, $z$ is the z-score and $\sigma$ is the standard deviation of the absolute error. The transactions generated from this equation form the input for the trading simulator in order to generate a profit.

*2) Trading Simulator:* A trading simulator calculates the exact profit that is generated based on the predictions of a trained system. For this project a simple trading simulator has been designed. Depending upon the sell or buy prediction made by the model, the respective functionality of the simulator is called which then calculates the profit that would be generated on the data given to it over the selected period of time [12]. The parameters used and their significance are as follows:

- Amount: The initial amount which the trader intended for investment in trading.
- Portfolio: The current number of shares with the trader.
- Money end: The amount left after the trading ends.
- Investment: This list that keeps track of investments, with the last value in list giving the total investment till time.
- Transaction cost: A transaction cost charged for trading - its value may differ for different countries.

The trading simulator uses two different processes for the calculation of the money end based on the sell or buy prediction. If the prediction suggests the trader to buy, the buy method is implemented. It calculates the allocated price for the trade which is the product of price of share and the quantity of shares. This method updates the money that accumulates with the trader, the portfolio and the investment values at the end of the trading period. The updated value of money the trader obtains for Buy transactions is calculated by subtracting the sum of allocated price and the transaction cost multiplied by the allocated price from the current value of money with the trader. The portfolio is updated by adding the quantity of shares for the current trade to the existing number of shares with the trader. The investment list is updated by adding the allocated price to the last value of the list to give a cumulative value.

Conversely, If the prediction suggests the trader to sell, the sell method is implemented. It calculates the allocated price for trade which is the price of share times the quantity of shares. This method updates the money left with the trader, the portfolio and the investment values at the end of the trading period. The updated money with the trader for Sell transactions is calculated by adding the sum of allocated price and the transaction cost multiplied by the allocated price to the current value of the money with the trader. The portfolio is updated by subtracting the quantity of shares for the current trade from the existing number of shares with the trader. The investment list is updated by subtracting the allocated price from the last value of the list to give a cumulative value.

## IV. Data Description / Preparation

### A. Tape Data

*1) Tape Explanation:* Time and sales, also known as "the tape", is a record of all stock transactions throughout the trading day and typically includes trade size, price and time of trade. Traders use the tape data to identify shifts in supply and demand in markets by developing an intuition for how the tape moves when shifts are occurring. An executed trade on the tape is called a print. A print contains three vital pieces of information: the time of trade execution, the price that the trade was executed at, and the amount of shares traded [**?**].

*2) Tape Purpose:* The tape displays the moves of the market's big players. Because all transactions must be recorded on the tape, they have no place to hide if they're doing lots of volume. Some traders use that information to follow the big players, while others choose to predict when the big players are done buying or selling, and trade the reversal. Some traders solely use the tape in trading. They make relatively short term trades and allow supply and demand, as displayed on the tape, to dictate their exits and entries. Tape data helps to get the below insights [**?**].

- Speed of Transaction: Speed is one way traders utilize the tape to forecast future market direction. You can identify momentum by seeing an acceleration of executed trades in one direction.
- Rate of Volume: Most orders in the market are small and inconsequential, trading 100-200 shares per trade. It's important not to get too wrapped up in analyzing these trades. All combined, they matter, but there is so much noise in the modern market that our focus is better shifted on the less frequent large orders that really move markets.
- Trend of Transactions: A trend is simply a sustained supply and demand imbalance. When significant volume is being done on the offer, the market is telling you that buyers are getting aggressive and want to buy that market more than sellers want to sell it. If large orders are moving the price higher or lower consecutively, that's showing a supply/demand imbalance that one can take advantage of. By closely watching the trend of these transactions, trade offers may be able to identify incoming demand earlier than chart readers.

*3) Cleaning the Data:* As part of cleaning the data unwanted columns with information about buying and selling parties were dropped. Only the information about Time of Transaction, Date of Transaction, Price of Share and Quantity of Shares were retained.

*4) Processing the Data for Feature Creation:* A distributive approach using Pyspark has been used for processing the data keeping in mind how huge the size of data can get. As part of processing the first step was to provide a schema mapping each column value of the data to the correct data

type. Different operations and transformations like joining and partitioning were then applied to the data to get the below features which have been used to train the model to make sell or buy predictions.

- Month and Day: These features were extracted from the date value. These values allow the model to understand the relation between the trend of trade and the day of the month.
- Price: It is important to calculate the final profit using a trading simulator for the model.
- Quantity: the quantity of shares is required to get the total transaction amount which is price times the quantity. This again helps to the profit achieved by model predictions.
- Closing Price: It is the price of a stock at the end of each trading day. It is considered the most accurate valuation of a stock or other until trading resumes on the next trading day. Comparing the closing prices over earlier days help to measure the changes in market sentiment toward a stock.
- Highest and Lowest Price: These values help the model learn when generally the price of the stock is highest or lowest. They are directly linked with the decision about sell or buy, with a high price indicating sell and a low price indicating buy.
- Volume Traded: This is an important value to calculate the total transaction amount and which is used to understand the performance of the model.
- Simple Moving Average of Closing Price over 10, 16 and 22 Days: Simple Moving Average is one of the core technical indicators used by traders and investors for the technical analysis of a stock, index or securities. The average is taken over 10, 16 and 22 days.
- Exponential Moving Average over 10, 16 and 22 Days: EMA is a weighted moving average that measures a trend of a financial security over a given period of time (10, 16 and 22 in our case). EMA is used in trading to determine whether the price of a security is going up or down. Fig.1 explores this and the previous moving average for the tape data.
- Days until Weekend: This value helps the model learn about the Weekend Effect on trading. The weekend effect is a phenomenon in financial markets in which stock returns on Mondays are often significantly lower than those of the immediately preceding Friday.
- Days until Month end: In trading it is often noticed that the buying activities are frequent toward the start of the month. This parameter helps the model learn the relation between the time of the month and the buy and sell activities.

### *B. LOB Data*

The LOB data was provided as text documents. The data was stored in a JSON-like structure that had information on the current time and the price and quantity of bids and asks in the order book. Due to this data being stored in an irregular way a custom solution was needed to convert this data into tabular data that could be stored in CSV format. These text files were roughly 2.3GB in size, which is potentially an issue as some programs may attempt to load all this data into the computers memory at once, causing it to run out of memory. For this reason this custom program to convert the data into a CSV format was required to be memory efficient.

To minimise the amount of memory being used, the program used is based around a Turing Machine model that sequentially read the text document one line at a time. Files that are empty or in a irregular format are discarded.

Similar to the tape features, the program added relevant features to the converted LOB files. Based on A. Booth et al. [3] the LOB features with the most effect and created on the data can be categorised into 2 sections:

- LOB spread: information about how the best bid and best ask compare with each other.
- LOB ask and bid liquidity: summary information about both the ask and bid sections of the order book with outliers accounted for.

The following spread features, mid price and microprice, were calculated using the following equations:

$$P_{micro} = \frac{(Q_a \times askBestPrice) + (Q_b \times bidBestPrice)}{Q_a + Q_b} \tag{2}$$

$$P_{mid} = \frac{askBestPrice + bidBestPrice}{2} \tag{3}$$

$$QuotedSpread = bidBestPrice - askBestPrice, \tag{4}$$

where micro price $P_{micro}$ in Eq.2 describes the quantity of asks and bids at best price - $Q_a$ and $Q_b$ respectively.

To evaluate the liquidity the following features are abstracted from the bids and asks:

- Median transaction price
- Interquartile range of transaction prices
- Best transaction price (least expensive bid price & most expensive ask price)
- Transaction quantity at best price
- Median transaction quantity
- Interquartile range of transaction quantities
- Number of Transactions

Finally, the LOB feature data and the tape data were combined by the time attribute using an inner join from the pandas Python library. The data combined with its features is now correctly formatted to be applied to the price prediction model.

## V. Results and Discussion

### *A. ARIMA*

In determining the hyperparameters for ARIMA, Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests were used. These allowed the null hypothesis of stationarity for d=0 to be rejected and instead accepted with d=1, through observation of associated P values. By visualising

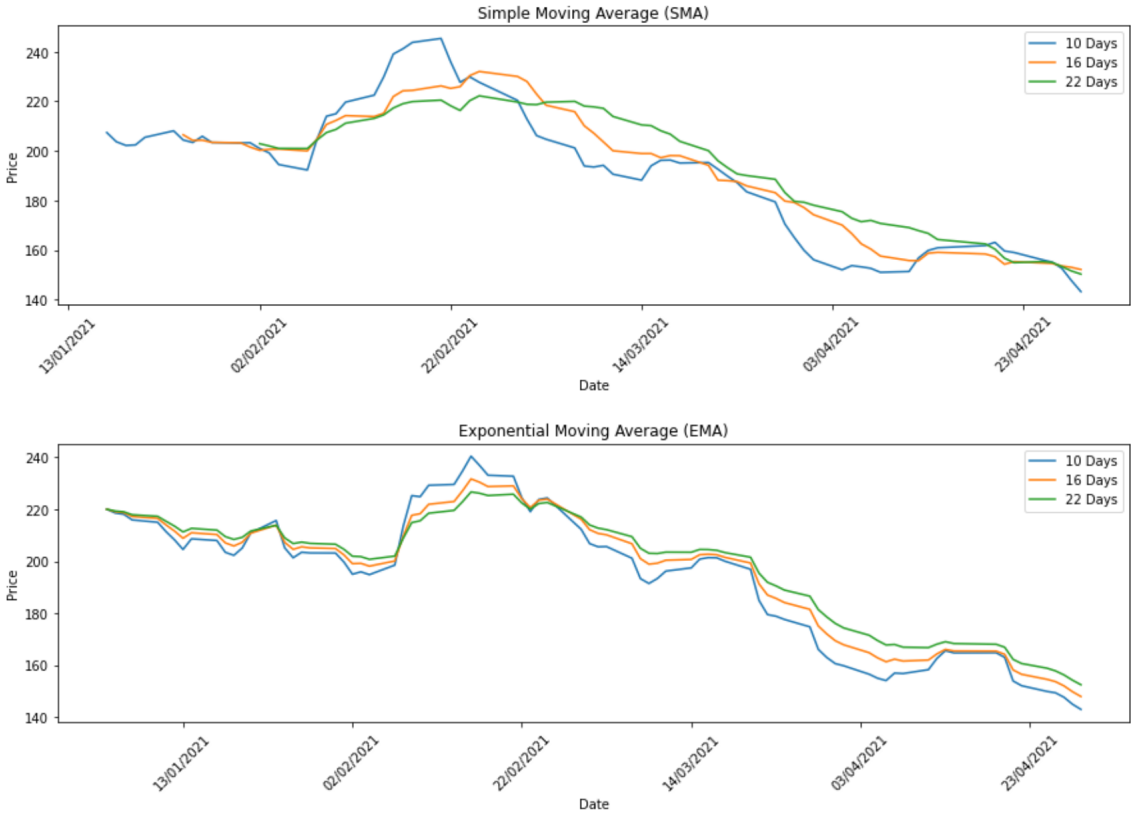


Fig. 1. Graph showing Simple Moving Average and Exponential Moving Average for 10, 16 and 22 days spans.

the associated auto and partial correlation functions for the differenced data, optimal hyperparameters (p,q) were determined to be (1,1) respectively. Hence the ARIMA(1,1,1) model was implemented, with statistics shown in Table I and graphs in Fig. 2. Forecasting was found to be extremely limited for this dataset, with convergence in prices occurring after approximately 4 steps. Hence worthwhile metrics determining the model performance could not be accurately determined for any time interval greater than two minutes.

TABLE I
ARIMA(1,1,1) STATISTICS.

| | coef | std err | z | P $>\|z\|$ |
|---|---|---|---|---|
| ar.L1 | 0.6671 | 0.096 | 6.921 | 0.000 |
| ma.L1 | -0.8328 | 0.076 | -10.933 | 0.000 |
| sigma2 | 25.5260 | 0.903 | 28.256 | 0.000 |

### B. Neural Prophet

Neural Prophet ultimately did not model the short term time series data well. This is predominantly due to the lack of seasonality or pattern within a short time frame. Shown in Fig. 3 over the same time frame as ARIMA, Neural Prophet forecasts a continued increase where ARIMA remains stationary.

However forecasting predictions on the daily closing prices showed a more appropriate prediction as shown in Fig. 4. This is primarily due to the seasonality being recognised within the deep learning aspect of the model.

### C. Random Forest

The features used for the random forest implementation were all the valid columns of the combined tape and LOB files - bar the price labels. A train-test split of 75% and 25% was applied, with random state 0 for reproducibility. The regressor used was the Sklearn Ensemble library's 'RandomForestRegressor', and was instantiated with default hyperparameters. The fit method was applied to the training features and their associated labels, before applying the predict method on the test features, returning the array y_pred containing price predictions. Associated metrics of this model can be seen in Table II. These metrics reflect a very good fit of the model to the data, and justify the default hyperparameter approach used.

### D. A2C

The dataframe used contained daily entries of features calculated from the asset tape files. It contained 5 columns: date, open, high, low, close and volume. A random initialisation of the environment yielded a total reward of $-191.0$ and total profit ratio of $0.266$. After training the A2C model using the

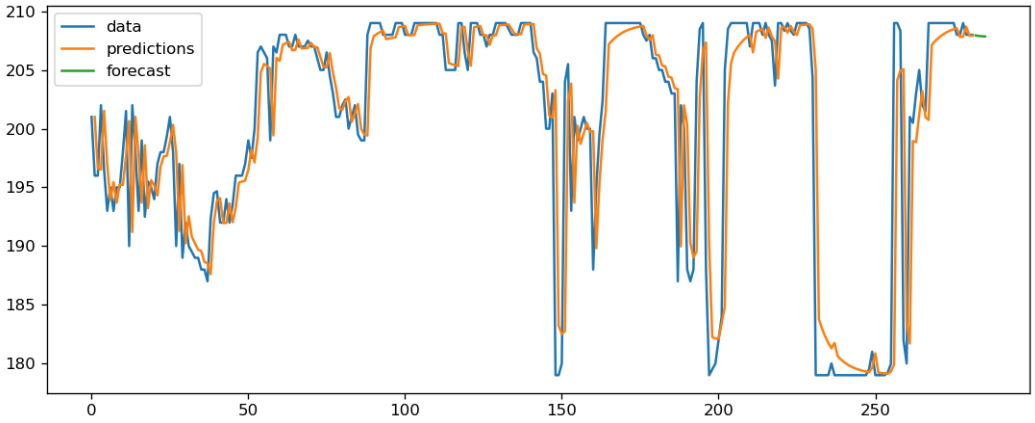


Fig. 2. Graph showing the ARIMA(1,1,1) predictions (orange) alongside the original data (blue) for the time interval t=8.912s to t=138.144s within the tape data from 04/01/2021. Forecast points can also be seen (green).

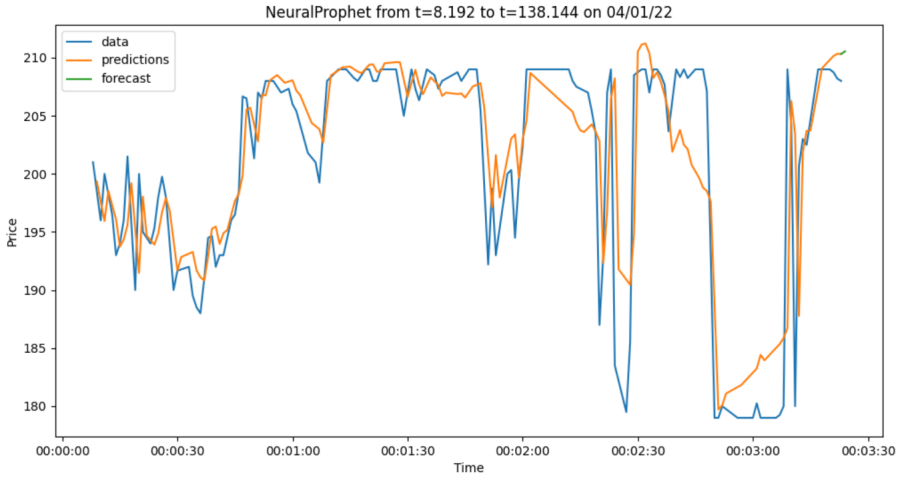


Fig. 3. Graph showing the NeuralProphet predictions (orange) alongside the original data (blue) for the time interval t=8.912s to t=138.144s within the tape data from 04/01/2021. Forecast points can also be seen (green).

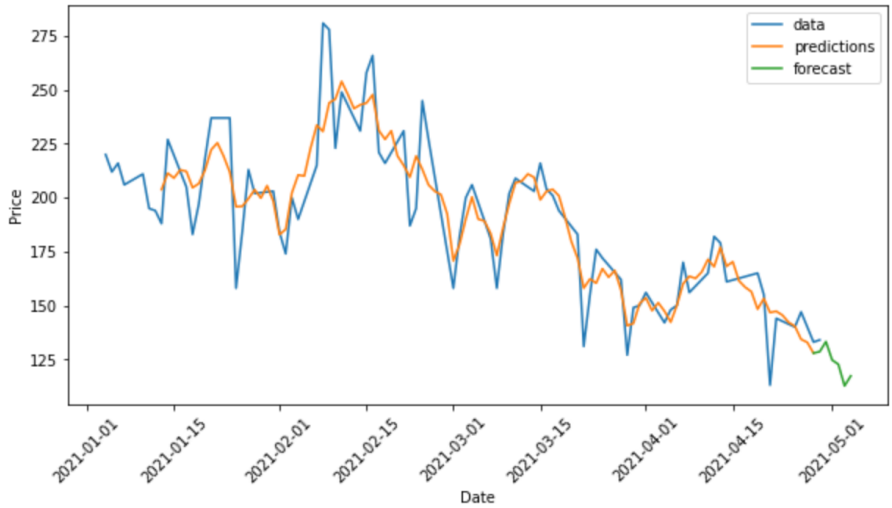


Fig. 4. Graph showing the NeuralProphet predictions (orange) alongside the original data (blue) for the daily data from 04/01/2021 to 04/05/2021. Forecast points can also be seen (green).

TABLE II
RANDOM FOREST METRICS FOR THE COMBINED LOB AND TAPE-DERIVED FEATURES.

| Metric | Score |
|---|---|
| Mean Absolute Error | 1.1382 |
| Mean Squared Error | 8.3366 |
| Root Mean Squared Error | 2.8873 |

MLP LSTM policy - a deep neural network containing a long short-term memory (LSTM) layer, the results from the testing subsection of the data were a total reward of 18.0 and total profit ratio of 1.013. This policy can be seen in Fig. 5.

This model was then implemented for datasets 2 and 3, where the tape files were cleaned in a similar fashion as with the original dataset in order to produce the necessary daily features. Where dataset 2 produced a total reward of 0.0 and total profit of 0.861, the strategy shows an initial selling action followed by all buying actions, suggesting that the model correctly identified a strong feature signal and predicted the asset price would be staying lower in the test period. Reproduction of the implementation with dataset 3 yielded a total profit of 1.479 and reward of 94.0 with an equivalent action space. While the agent manages to create profit for each dataset, there is limited justification of the interpretation of reward.

### *E. Custom Model*

The data used for the performance analysis of the custom model had information about the price of the share, the number of shares for each transaction and the trading signal i.e. either buy or sell. These signals were predicted by the custom model and used for the final profit calculation resulting from the custom model. This data is fed to the trading simulator which, depending on the trading signal associated with each transaction, uses the buy or sell method to calculate the investment, the number of shares and the outstanding amount with the trader. On feeding the data over a period of 4 months to the trading simulator a significant profit ratio of 2.238 was gained.

## VI. FURTHER WORK AND IMPROVEMENT

Since this project involved several avenues of data cleaning, exploration, modelling and trading, there are a plethora of options for developing the work. In particular, the transaction decision algorithm has proved to be successful but it could be improved. Adjusting the transaction decision algorithm in order for the purchase quantity to vary proportionally to the standard deviations from the mean would allow the random forest-based model to make many safe transactions and a few risky transactions, in the hope to make more profit in a strategically safe way. This model performance could additionally be improved through experimentation with a heterogeneous ensemble. As explored by A. Booth et al. [2], ensemble methods that use various data models (known as heterogeneous ensemble methods) have proven to perform better than ensemble methods that only use one type of data model (homogeneous ensemble methods). In future experiments a heterogeneous ensemble method could be tested that utilises various data models such as: neural networks, decision trees, logistic regression, expert based systems and more.

Improvements that focus on the tape data and its applications include extending the A2C model and creating more tape features. For example, exploring the potential of multiple independent actor-learners in trading an asset for profit could be achieved with the Asynchronous Advantage Actor Critic (A3C) algorithm. Ideally, an environment that could utilise both the LOB and tape-derived features would be most promising. For the tape features used in the random forest model, more could be added to aid training:

- Momentum: used by investors to trade stocks in an uptrend by going long (or buying shares) and going short (or selling shares) in a downtrend. In other words, a stock can be exhibit bullish momentum, meaning the price is rising, or bearish momentum where the price is steadily falling.
- Acceleration: This is an oscillator indicator which shows when the price is reducing the dynamics of the current trend, and the trader then has to estimate its acceleration to enter or exit the market.
- Rate of Change: The Rate of Change indicator, which is also referred to as simply Momentum, is a pure momentum oscillator that measures the percent change in price from one period to the next. The ROC calculation compares the current price with the price "n" periods ago.
- Relative Strength Index: This is a momentum oscillator that measures the speed and change of price movements. The RSI oscillates between zero and 100. Traditionally the RSI is considered overbought when above 70 and oversold when below 30.

The current trading simulator is designed in a way where it is specific to a particular asset. So, asset specific datasets should be fed to it in order to calculate the profit value and margin. Further improvements can be made in the trading simulator so that it is automatically able to differentiate between assets in a common dataset and calculate their respective profit values and produce a segregated result.

## VII. CONCLUSION

Stock prices and trade signals were successfully predicted of a given asset through implementation of random forest and feature engineering. Univariate (ARIMA and Neural Profit) and RL (A2C) models were used against the custom model to test and compare the performance. The final result obtained from the custom model produces a profit ratio of 1.209 higher than the A2C model. Though the implementation of these models differ in data granularity and test splitting, hence resulting in profits gained over different time periods, they both provide useful insight into the nature of Level2 data and its trading potential.

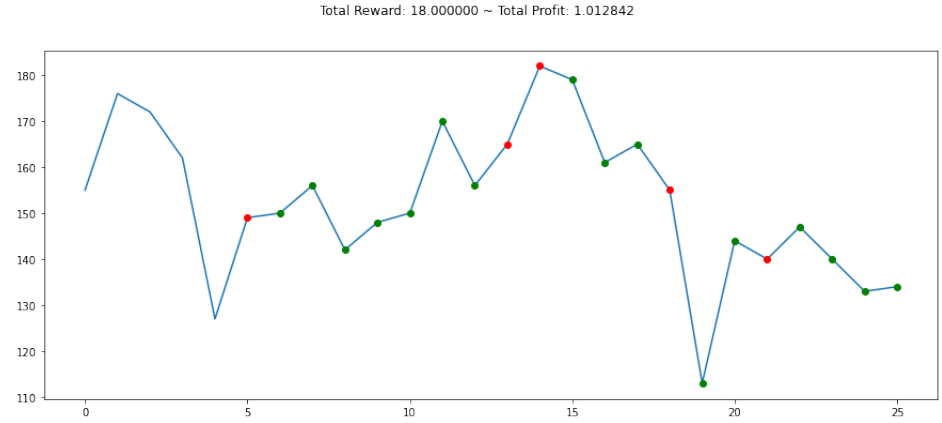


Fig. 5. Actions performed by the trained A2C model on the daily tape feature data for the original dataset. Here green marks represent the buy action and red marks represent sell.

## REFERENCES


[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control. Wiley, 2016.

[2] A. Booth, E. Gerding, and F.McGroarty. "Automated trading with performance weighted random forests and seasonality" Expert Systems with Applications, vol. 41, no. 8, pp. 3651–3661, 2014. [Online]. doi: j.eswa.2013.12.009

[3] A. Booth, E. Gerding, and F. McGroarty, "Predicting Equity Market Price Impact with Performance Weighted Ensembles of Random Forests", in IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr), 2014, pp. 1–8.

[4] Á. Cartea, S. Jaimungal, and J. Penalva, Algorithmic and high-frequency trading. Cambridge, UK: Cambridge University Press, 2015, pp. 16.

[5] Á. Cartea, S. Jaimungal, and J. Penalva, Algorithmic and high-frequency trading. Cambridge, UK: Cambridge University Press, 2015, pp. 46.

[6] A. Both, E. Gerding, and F.McGroarty. "Performance-weighted ensembles of random forests for predicting price impact" Quantitative Finance vol. 15, no. 11, pp. 1823-1835, 2015.

[7] O. Triebe, H. Hewamalage, P Pilyugina et al. NeuralProphet: Explainable Forecasting at Scale, CoRR, 2021.

[8] G. Hackeling, "Bagging," in Mastering machine learning with scikit-learn: Learn to implement and evaluate machine learning solutions with scikit-learn, Packt Publishing, 2017.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," 1996.

[10] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[11] "Gym-anytrading." https://awesomeopensource.com/project/AminHP/gym-anytrading.

[12] "WariorTrading" https://www.warriortrading.com/day-trading-terminology/

[13] "Investopedia" https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp