Short-term Price Prediction using Limit Order Book Data for Profitable High Frequency Trading in Markets with Low Liquidity

Morgan Wynne
MSc Data Science
University of Bristol
Bristol, United Kingdom
vq22301@bristol.ac.uk

Tim Chen
MSc Data Science
University of Bristol
Bristol, United Kingdom
zc16388@bristol.ac.uk

Reza Rahimi
MSc Data Science
University of Bristol
Bristol, United Kingdom
lv22609@bristol.ac.uk

***Abstract*—Traders are increasingly applying price prediction algorithms that use limit order book data to generate profit from high frequency trading. This report addresses two key issues in the current literature. Firstly, current state-of-the-art algorithms make unrealistic assumptions about an asset's execution price that inhibit their practical application to active trading scenarios – especially in markets characterized by low liquidity. This report proposes a new target variable that overcomes this assumption and uses it to achieve consistently profitable outcomes in active trading. Secondly, models are becoming more complex, trading interpretability and explainability for better performance. This report investigates this trade-off by comparing the performance of two differing approaches to limit order book modelling. 1) A decision tree with handcrafted features, and 2) a significantly more complex artificial neural network that uses convolutional and long short-term memory layers on multiple states of the limit order book to predict trading signals.**

I. Introduction

Most of the trading volume in major markets is executed automatically using algorithms. These algorithms use diverse data types and trading strategies to predict asset price movements, to trade, and to ultimately generate profit. Level 2 financial market, or Limit Order Book (LOB), data documents the full set of limit orders posted by traders at any given point in time. Used effectively, this data contains price signals that enable profitable High-Frequency Trading (HFT).

The use of LOB data for price prediction is an active field of study and many contemporary methods make unrealistic assumptions and consequently are unprofitable when applied in active stock trading scenarios. The first assumption is the absence of trading costs and the second is that trades can be executed at the mid-price (Zaznov et al., 2022). Whilst academics argue that the second assumption is irrelevant when limit orders are used to lock in the mid-price for a given trade, this makes the further assumption that, for each trade, both limit orders will be fulfilled by the end of trading. This assumption is invalidated in markets characterised by frequent liquidity shortages that cause the best bid (ask) price at a given time to frequently fall well below (rise well above) the lowest (highest) transaction price in any given day.

This report begins by briefly reviewing contemporary methods and models for predicting stock prices using LOB data, with a focus on four aspects: 1) current model performance on directional price prediction tasks and their practical limitations, 2) the importance of model interpretability, 3) limit order book modelling, and 4) supervised feature extraction. Using these findings, the report describes the extraction of useful features from LOB data for an asset with low liquidity and explores the problems that low market liquidity provides. From discussion of potential solutions, it was concluded that both engineering a suitable target variable and applying liquidity-related trading rules is necessary for ensuring profitability when used in active trading. These principals were then applied using a decision tree model with hand-crafted features and a Convolutional Neural Network (CNN) with a Long Short-Term Memory (LSTM) layer that used unsupervised feature extraction. Each model was carefully evaluated by measuring it against multiple performance criteria and assessing its profitability in active trading.

II. Literature Review

The study of stock price prediction using LOBs is an active field of study, with developments facilitated by continuous improvements in machine learning techniques that allow models to better handle the high cardinality and frequency of LOB data. In 2022, Zaznov et al. produced a survey, which synthesises contemporary methods and results for predicting stock price movements using LOB data, and summarises the limitations pertaining to data input, modelling, and experimental setup (Zaznov et al., 2022).

From 2005 to 2022, models of increasing complexity were applied to price prediction using trading data, from simple linear regression and Hidden Markov Models (HMM) to deep neural networks that utilise multiple customised hidden layers and transformer blocks. Zaznov et al. evaluates the models and results of each study in terms of its performance, practicality, and experimental reproducibility, and draws several conclusions: 1) As model complexity increases, the tendency to overfit the training data increases, consequently reducing generalisation performance. 2) Reproducibility and comparability between studies is difficult. Studies use varying performance metrics and different datasets; many of which are not publicly available. 3) All make two assumptions which limit their practicality when applied to active stock trading: there are no trading costs and trades can be executed at the mid-price.

The first benchmark LOB dataset was published in 2017, improving comparability between subsequent studies (Zaznov et al., 2022, para. 21). The dataset contains 10 trading days of

millisecond-by-millisecond data for five stocks traded on the Helsinki Stock Exchange in 2010. The data includes 10 levels of LOB data and five indicator variables representing whether the mid-price increased (=1), did not change (=2), or decreased (=3) over five increasing intervals. Zaznov et al. highlight several problems with this dataset, including class imbalance, that the dataset may be unrepresentative of modern trading patterns, and that the five target variables limited models once again to making the unrealistic assumption that trades may be executed at the mid-price.

Zaznov et al. proceed to evaluate the practicality of the second-best performing model, DeepLOB, which attained an F1 score of 83.40% on the benchmark LOB dataset. When applied to active stock trading for ten well-known tickers using a simple trading algorithm, the model achieved an average profit per trade of GBX 0.01. During the same period, the average spread (the gap between the best ask price and the best bid price) was GBX 0.1 – ten times the average profit per trade. If it was assumed that trades may be only executed as market orders at the best ask and bid prices, the model would have certainly made a loss. This highlights the inapplicability of the mid-price assumption made in contemporary experimental set ups to markets with low liquidity and is the first major motivation for the discussions and experiments described in this report.

Another major problem with applying these models in active training is their interpretability. As models increase in complexity, there is a trade-off between performance and explainability. Deloitte described the need for explainable models as "a top priority for many banks" (Deloitte, 2022, para. 9). Exacerbated by the 2008 crisis and the failure of LTCM, Chief Data Officers (CDO) at major banks are constrained by tight MiFID regulations that assess the risk parameters inherent in their algorithms (European Central Bank, 2019). If banks are unable to interpret the decisions that their algorithms make, it is very difficult to justify their deployment to active trading. Deep learning techniques like DeepLOB and TransLOB consist of many hidden layers, yielding a black-box model of prediction and low interpretability. One data science technique for increasing interpretability whilst retaining accuracy is to use less complex models and supervised feature extraction to transform the model input into explainable features that aim to capture the important information in the data. This fundamental choice between supervised or unsupervised feature extraction forms the second key factor explored in this report.

To understand which features can be extracted from LOB data to explain price movements, it is useful to consider the LOB as a modelling task. Gould et al. published a survey examining the findings from previous attempts to model a LOB and discusses how these models provide insights into certain aspects of the mechanism (Gould et al., 2013, p. 1). Several key features of LOBs emerge, including long memory, the effect of order flow imbalance on permanent price impact, depth profiles and patterns, and market liquidity. Three key aspects that have potential to effect price movements were selected to be examined in more detail: 1) Order Flow Imbalance (OFI) over a given period, 2) volume-related features, and 3) the bid-ask spread.

OFI expresses the net order-flow imbalance at the best quotes, considering market orders, limit orders, and limit order cancellations (Xu et al., 2019, p. 5). Cont et al. hypothesised that given a short period of length *t*, the difference between the net flow of orders at the best bid price and the best ask price reflects demand and supply pressures that impact contemporaneous price movements (Cont et al., 2013, p. 1). The authors used one month of trading data from April 2010 for 50 randomly chosen stocks from the S&P 500 index (Xu et al., 2019, p. 6). The OFI and contemporaneous price change were estimated for intervals of 10 seconds and grouped into 30-minute windows. For each window, mid-price (*m(t)*) changes over each interval were regressed on the corresponding OFI. The regression coefficient on OFI was discovered to be statistically significant in 98% of cases, implying a strong positive correlation between OFI and mid-price movements.

Gould et al. observes that price and market impact is a concern for traders wishing to trade a volume larger than the depth at the best quotes (Gould et al., 2013, p. 14). Price impact describes the effect of a market order on the best ask (*a(t)*) and best bid prices (*b(t)*), whilst market impact describes the overall effect of a trade on the LOB (*L(t)*). For example, if a trader wishes to submit a buy market order 20 times the lot size of an asset and the depth at *a(t)* is 7 times the lot size, the price impact would be the change in *a(t)* that would result if the full trade was executed at time *t*. The market impact would be the corresponding effect on both *a(t)* and the volume at the next highest ask price. Future price movements are therefore likely affected by the relative volumes at the best quotes, and the overall volumes on each side of the order book. Furthermore, Cartea et al. discovered that when the LOB is buy-heavy, i.e. there is greater volume at *b(t)* than *a(t)*, then it is much more likely that the next market order will be a buy order than a sell order (Cartea et al., 2015, p. 2). Cartea et al. quantify the imbalance between order volume at the best quotes using Order Book Imbalance (OBI), defined as:

$$OBI_t = (V_t^b - V_t^a) / (V_t^b + V_t^a) \quad (1)$$

where V_t^b represents the order volume at the best bid price V_t^a represents the best ask price at time *t* (Cartea et al., 2015, p. 4).

Studies on LOBs for numerous exchanges observed that traders placed more orders with a price above (below) the best bid price (best ask price) when the bid-ask spread (*s*) was larger than its median value (Gould et al., 2013, p. 12). Biais et al. (1995) hypothesised that this was because, when *s* is larger, market orders become less attractive. Gould et al. further argue that it is explainable within a zero-intelligence model – if limit order prices are chosen randomly, then when *s* is large, there is a higher probability that an incoming limit order will fall between *b(t)* and *a(t)* (Gould et al., 2013, p. 12). Regardless, this pattern implies that changes in *b(t)* and *a(t)*, and consequently *m(t)*, are affected by the size of the bid-ask spread at time *t*.

III. Methodology

A. The Dataset and Exploratory Data Analysis

The datasets used in the study were provided by HSBC Global Markets and constitute the "tape" and LOB data for a single tradeable asset over a 6-month period. Each item in the LOB dataset corresponds to the state of the LOB following a change to limit orders quoted. This change could be caused by an incoming market order or limit order, or a cancellation of a limit order. The dataset for each trading day consists of three attributes: the timestamp *t* corresponding to the LOB at time *t*

($L(t)$), the bid-side depth profile (p, n^b (p, t)) at t, and the ask-side depth profile (p, n^a (p, t)) at t. Each item in the tape dataset represents a market order with corresponding timestamp, execution price, and volume traded.

Data for one trading day, chosen arbitrarily, was used to explore patterns and features of the LOB for this asset. First, *b(t)*, *a(t)*, *s(t)*, and *m(t)* were extracted and aggregated to attain early signals for asset price volatility and market liquidity. Fig. 1 shows that *a(t)* is highly volatile with a Coefficient of Variation (CV) of 56.11% for the trading day selected. Consequently, *m(t)* is also highly volatile with its range over three times the mean value.

Statistic	***b(t)***	***a(t)***	***s(t)***	***m(t)***
Mean	103.69	164.28	60.60	133.99
Standard Deviation	8.54	92.18	91.80	46.66
Minimum	1.00	90.00	1.00	45.50
Maximum	110.00	800.00	703.00	454.00

Fig. 1. Summary Statistics for *b(t)*, *a(t)*, *s(t)*, and *m(t)*.

Fig. 2 visualises *b(t)* and *a(t)* for 1000 arbitrarily chosen consecutive timestamps. *a(t)* is characterised by frequent liquidity shortages on the sell-side of the LOB that cause it to increase sharply at regular intervals before returning to a stable price.

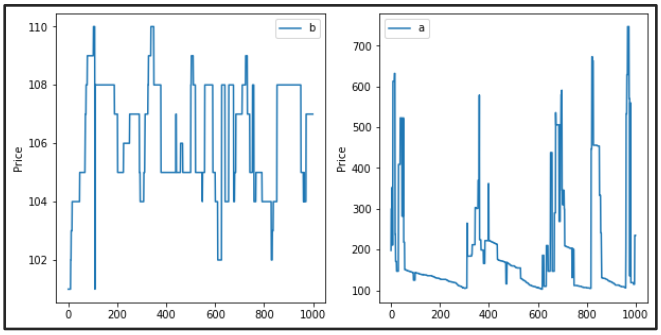

Fig. 2. *b(t)* and *a(t)* for 1000 arbitrarily-selected consecutive timestamps.

B. Baseline ARIMA and MACD Models

As a performance baseline, an ARIMA model was applied in conjunction with a simple trading algorithm. In alignment with the literature, mid-price was used as a target variable, and the ARIMA model was designed to perform a one-step forecast of the mid-price based on the historic mid-prices. A 10-second slide window is applied to compute the average mid-price for each period. In this way, the computational intensity is reduced, and the prediction reflects the mid-price for the prior period, rather than a single timestep in the LOB. Predictions are made using a rolling forecast method. To do this, the ARIMA model is recreated after each new observation is received.

Rather than the solely considering the price, the Moving Average Convergence/Divergence (MACD) of the mid-price was investigated. The MACD indicator gives more information about the trend of the price. The MACD measurement is calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. EMA places more weight on the latest data and is thus, more responsive to the latest price changes. A positive MACD value indicates an upward trend whereas a negative MACD value indicates a downward trend.

The trading algorithm assumes that only one share can be held each time. When there is no inventory, it compares the next predicted price with the mean price from the previous 3 time periods, and checks if the mean of the previous 3 MACD values is negative. If both conditions are met, it will place a buy market order. Under the scenario that there is a positive inventory, it does the same comparison. If the predicted price is lower than the mean price, but higher than the mid-price, and the MACD mean is positive, it will then sell the share. Otherwise, a random number (0-1) will be generated, and the algorithm will sell or keep the share with a 50/50 probability.

Under the assumption that trades can be executed at the mid-price, the trading algorithm generates a small profit in most cases.

C. Supervised Feature Extraction

It was hypothesised that model performance may be improved by extracting handcrafted features proven to be statistically significant in the academic literature when used to predict price movements. Each feature is described in turn.

1) Order Flow Imbalance

OFI was extracted from the data in alignment with the equations used by Cont et al. Let $q^1(\tau_n)$ represent the total order volume at the best ask price and $r^1(\tau_n)$ the total order volume at the best bid price at timestamp τ_n. Let both represent values *after* applying the effect of the n^{th} order arrival or cancellation (Xu et al., 2019, p. 5). For a given interval (t_{k-1}, t_k],

$$OFI(t_{k-1}, t_k) = \sum_{\{n|t_{k-1}<\tau_n\leq t_k\}} e_n, \tag{2}$$

where

$$e_n := \Delta W(\tau_n) - \Delta V(\tau_n), \tag{3}$$

where

$$\Delta W(\tau_n) = \begin{cases} r^1(\tau_n), & \text{if } b^1(\tau_n) > b^1(\tau_{n-1}), \\ r^1(\tau_n) - r^1(\tau_{n-1}), & \text{if } b^1(\tau_n) = b^1(\tau_{n-1}), \\ -r^1(\tau_{n-1}), & \text{if } b^1(\tau_n) < b^1(\tau_{n-1}); \end{cases} \tag{4}$$

and

$$\Delta V(\tau_n) = \begin{cases} -q^1(\tau_{n-1}), & \text{if } a^1(\tau_n) > a^1(\tau_{n-1}), \\ q^1(\tau_n) - q^1(\tau_{n-1}), & \text{if } a^1(\tau_n) = a^1(\tau_{n-1}), \\ -q^1(\tau_n), & \text{if } a^1(\tau_n) < a^1(\tau_{n-1}). \end{cases} \tag{5}$$

The OFI was extracted for 10 second intervals and the experiment carried out by Cont et al. described in Section II was replicated for data on one full trading day. For 88% of non-overlapping 30 minute windows, the 10 second OFI was positive and statistically significant at the 95% level. Fig. 3 shows the relationship for the full trading day.

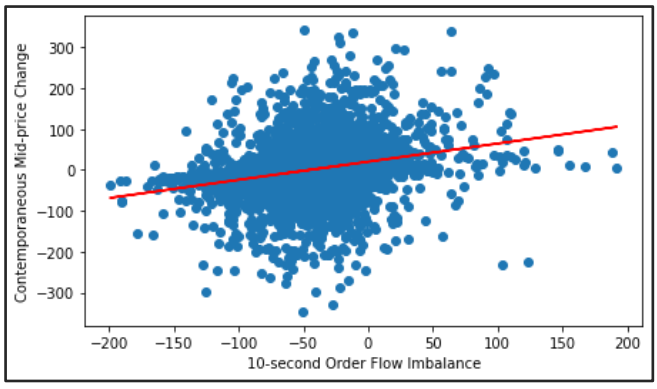

Fig. 3. 10-second contemporaneous mid-price change regressed on OFI.

2) Order Book Imbalance

The OBI at each timestamp was extracted by applying (1) as in Cartea et al.

3) All-Level Order Volume Difference (AVD)

Although the OBI captures volume-related effects on price movements at $b(t)$ and $a(t)$, it doesn't consider volume changes across all levels of the LOB. For example, in a case where volume at $b(t)$ is high but total volume at other levels of the bid side is low, then OBI would be high, but a large sell order could cause a large negative price impact. To capture the relative depth profiles across all levels, the absolute difference between the total volume of quotes on the buy and sell sides of the LOB was extracted from the data and was calculated as:

$$AVD = \sum_{p=0}^{\max p}{}_{m} N^b(p,t) - \sum_{p=0}^{\max p}{}_{m} N^a(p,t), \qquad (6)$$

where m is the total number of levels, p is the relative price, $N^b(p,t)$ is the depth at relative price p and time t on the bid side, and $N^a(p,t)$ is the depth at relative price p and time t on the ask side. AVD was extracted for each timestamp t.

An indicator variable describing the direction of the forward 10s mid-price change (1 – mid price increased from t to t + 10s, 0 – mid-price decreased from t to t + 10s) was extracted. This feature was then regressed on the 10s OFI, OBI, AVD, and $s(t)$ for one full trading day using binary logistic regression. Coefficients for each variable and their statistical significance were used as a preliminary test for each feature's ability to explain short-term mid-price changes.

The coefficients for the intercept, OBI, AVD, and $s(t)$ were statistically significant at the 99.9% level. Whereas the OFI was statistically insignificant with a p-value of 0.49. The coefficients for the intercept and OBI were positive and all others were negative. The results imply that although OFI can ably predict contemporaneous mid-price changes, it may be less effective for predicting forward price changes. The model attained an accuracy of 80.2%. However, when $s(t)$ was removed, this score decreased to 61.4%. It's high significance and negative sign are potentially reflective of mean reversion following liquidity shortages on the ask-side of the LOB and not actual trading signals. When removed, OFI also becomes statistically significant at the 99.9% level, hinting at potential problems with the model in its current form. This issue is addressed in detail in the next section.

D. Target Variable Engineering

Zaznov et al. highlight the limiting effect that the assumption that trades can be executed at the mid-price has on real-world application of contemporary models. The problem is exacerbated for markets with frequent liquidity shortages, where predictions of mid-price changes are often instead predicting large bid-ask spreads that reflect low market liquidity. To validate this hypothesis, the execution prices for one trading day of LOB data were examined. Fig. 4 shows summary statistics for the execution price of trades for the same trading day as Fig. 1. The mid-price assumption only holds if limit orders are submitted at equal distances from the desired mid-price and are executed by the end of the trading day. Reviewing both tables reveals that, for this asset, when the best ask price rises, quotes submitted at this dramatically higher level will never be fulfilled by the market. The mid-price is therefore an unviable target variable for models whose purpose is to trade profitably in markets with low liquidity.

Statistic	Execution Price
Mean	104.36
Standard Deviation	3.61
Minimum	90.00
Maximum	110.00

Fig. 4. Summary stats for execution price for trading day used in Fig. 1.

A trading algorithm that aims to trade profitably requires a model that predicts changes in the *true* price of the asset. Based on the premise that an asset's true price is what someone last paid for it, it was hypothesised that the most recent execution price fulfilled this requirement. Thus, the tape and LOB data were combined, and a new target variable was extracted that indicated whether the most recent execution price at time t + 10s had increased (= 1) or decreased (= 0) relative to the most recent execution price at time t. When regressed on OFI, OBI, AVD, and $s(t)$ for the same full trading day used previously, each of the explanatory variables were statistically significant at the 99.9% level, achieving an accuracy score of 63.5% with $s(t)$ and 62.7% without it. Although this is a logical improvement from mid-price, the most recent execution price does not consider market liquidity at time t or t + 10s. Consequently, a wide bid-ask spread at either timestamp may mean that the trader cannot execute trades at a price near the most recent execution price, resulting in losses despite the model indicating a trade signal.

A new target variable was proposed to tackle this problem and is calculated as:

$$Signal = \begin{cases} 0, & \text{if } b^1(\tau_n) \geq a^1(\tau_{n+t}) + MS, \\ 2, & \text{if } a^1(\tau_n) \leq b^1(\tau_{n+t}) - MS, \\ 1, & \text{otherwise,} \end{cases} \qquad (7)$$

where $b^1(\tau_n)$ and $a^1(\tau_n)$ represent the best bid and ask price at timestamp τ_n, and $b^1(\tau_{n+t})$ and $b^1(\tau_{n+t})$ represent the best bid and ask price after some arbitrary regular interval t. A value of 0 signifies to the trading algorithm that the current bid price is greater than or equal to the future ask price plus some arbitrary Margin of Safety (MS). Thus, the algorithm should sell the asset and buy it back after t seconds to make a profit. The same logic is applied to a buy signal, represented by a value of 2. In all other cases, the equation returns a value of 1, indicating that trading at timestamp τ_n and reversing the trade at τ_{n+t} is not profitable for a given MS. MS can be chosen to best

accommodate a trader's risk tolerance. This new target variable considers market liquidity at both τ_n and τ_{n+t}, and when used as a target variable for price prediction, indicates trade signals based on the *true* price of the asset. Thus, price prediction tasks described in the rest of this report use (7) with MS = 2 and t = 10s for each model's target variable.

E. Modelling

Two approaches to modelling the target variable were considered. Each approach was dictated by the model input chosen: handcrafted features described in Section C or unsupervised feature extraction using artificial neural networks (ANN).

1) Modelling with Handcrafted Features

Decision tree models for classification provide a highly interpretable non-linear mapping between a vector of inputs and a class label. Using a decision tree model for this task assumes that the input variables, 10s OFI, OBI, AVD, and the

bid-ask spread, at time *t* are necessary to predict whether a trade executed at time *t* and reversed at time *t* + 10s will generate a profit.

To reduce selection bias, 9-fold cross-validation was used on a randomly chosen window of 9 consecutive trading days. To simulate batch training in active trading, the model was trained on eight trading days and then used to predict trading signals and resulting profit for the ninth day.

2) Modelling using Unsupervised Feature Extraction

Instead of running supervised learning algorithms on handcrafted features, neural networks can be used to learn features automatically from the LOB data. Convolutional Neural Networks (CNNs) are very powerful tools for extracting information using filters. With a careful design of filters and strides, it can learn relevant features and capture useful local spatial dependencies from the input data. In the example of using the LOB, each level in the LOB contributes differently to the change in the price.

Long Short-Term Memory Networks (LSTMs) were designed originally to solve the gradient vanishing problem that occurs in Recurrent Neural Networks (RNNs). The concept of gates was introduced in LSTM to control the flow of information, thus enabling the network to "choose" what information to remember and what to forget. Like RNNs, LSTMs learn time dependencies from the data but also maintain a much longer memory.

Data normalisation is required before feeding the data into the neural network. It is an important step because the model performance is highly dependent on the normalisation process. A robust normalisation scheme allows the model to adapt to frequent price changes in a volatile market. Instead of normalising data statically, a dynamic normalisation scheme is used for this task. To normalise the current day's data, the mean and standard deviation of the data from the three previous days are used. Specifically, normalisation is done by removing the mean and scaling to unit variance. The normalised value for a sample *x* is calculated as:

$$z = (x - u) / s \tag{8}$$

where *u* and *s* are the mean and standard deviation of the training samples respectively. In this task, the most recent 10 states of the LOB are used as input to the network and a single state contains 6 levels of LOB data. Specifically, a single input can be denoted by

$$\mathbf{X} = [x_1, x_2, \dots, x_t, \dots, x_{10}] \in R^{\mathbb{10}\times\mathbb{24}}, \tag{9}$$

where $x_t = \left[P_a^{\{i\}}, V_a^{\{i\}}, P_b^{\{i\}}, V_b^{\{i\}}\right]_{i=1}^{n=6}$ with $P^{\{i\}}$ and $V^{\{i\}}$ denotes the price and volume of the *i*-th level of the limit order book. This results in a single input of shape (10, 24). After processing the limit order data, the 10 second forward trade signal described in (7) with MS = 2 is used as target variables for each timestamp.

Instead of feeding the data directly to an LSTM network, the network was designed to combine both CNN layers and LSTM layers. Feature maps learned by the CNN layers not only capture the spatial dependencies, but also preserve the time information in the data. In total, the network contains three convolutional layers and one LSTM layer. The purpose of the final LSTM layer is to capture the time dependencies in the resulting feature maps. CNN layers use filters with a defined stride to slide through the input, which means parameters are shared between input features. In this task, the size of the filters from the first two convolutional layers is set to (1 x 2) with a stride of (1 x 2). The intuition behind this is that the parameters are shared by each price-volume pair at the first layer. At the second layer, the receptive field is expanded as parameters are now shared by each level of the limit order book. After two layers of convolution, the resulting feature maps will have a shape of (10, 6). In the last convolutional layer, a filter of size (1 x 6) is applied to integrate all the information from the feature maps, resulting in final feature maps of shape (10, 1). Note that each convolutional layer uses 32 filters, the output is then reshaped and fed into the LSTM layer with 64 units. The output layer uses a softmax activation function and hence the output reflects the probability of the three trading operations (sell, neutral, buy).

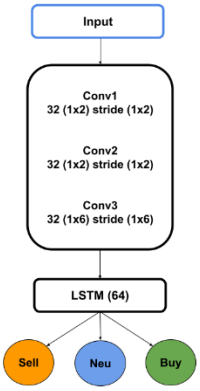

Fig. 5. Network architecture diagram.

F. Trading Algorithm and Profit Assumptions

The models described above map inputs to trade signals, which instruct the trader to buy or sell at time *t* and reverse the trade at time *t* + 10s to generate a net profit. During testing, false positives often led to outsized per-trade losses when the spread at time *t* or *t* + 10 was large. To mollify the negative effect of these instances on profitability, two simple trading rules were implemented:

1. If *s(t)* > *x*, where *x* is a variable to be determined, do not execute the trade.
2. If reversing the trade at time *t* + 10s is not profitable, hold the position as inventory.

For performance evaluation and comparability between model performance, it was necessary to include the value of inventory in profit calculation for each trading day. Therefore, total profit was calculated as the summation of three numbers: 1) profit from executed (and reversed after 10s) trades, 2) net profit from 'netting' the inventory of buy orders and sell orders at their average price, and 3) profit from selling (buying back) the remaining inventory at the day average bid (ask) price.

For example, the model generates 35,879 from trading on correct predictions (making the indicated trade and reversing it after 10s). It makes a total of 3,768 incorrect sell predictions, yielding a sell inventory of 3,768 with average price 105. It also makes a total of 7,531 incorrect buy predictions, yielding a buy inventory of 7,531 buy orders with average price £1.08. The day average bid price was 106. Total Profit (TP) is calculated as follows:

$$TP = 35{,}879 + (-3 * 3{,}768) + \big((7{,}531 - 3{,}768) * -2\big) = 17{,}049$$

For simplicity, this makes the realistic assumption that any remaining inventory can be sold (bought back) at the average bid (ask) price for the trading day. More sophisticated methods, including using the moving average bid (ask) price to inform inventory offloading decisions, are likely to yield superior profit performance.

This aim of this report is to create models that produce profitable outcomes in active trading, not maximise profit generation. Therefore, volume per trade is always equal to 1. Although this limits total profit generated, the additional complexity provided by mechanisms to optimise volume traded is outside the scope of this report.

IV. Data Description / Preparation

Nine consecutive trading days were randomly sampled from the dataset. For the decision tree model using handcrafted features, 10s OFI, OBI, AVD, and *s(t)* were extracted using the equations described in Section III C. The target variable – 10s forward price signal described in (7) with MS = 2 – was extracted for each timestamp in the dataset. Total instances of each class are shown in Fig. 6.

For the ANN, five consecutive days were randomly sampled from the dataset to train the model and feature vectors were extracted in the form described in Section III. The dataset consisted of 1,604,195 instances and exhibited similar class imbalances to those in Fig. 6. The model was trained using a window size of 20 and a stride of 1 and used the same target variable described above. Another five consecutive days were randomly sampled for testing.

Class Label	Number of Instances
0 (Sell Prediction)	262,833
1 (Neutral Prediction)	2,374,833
2 (Buy Prediction)	231,696
Dataset Size	**2,869,362**

Fig. 6. Decision tree Model dataset class summary.

V. Results and Discussion

The performance of each model is evaluated in turn. Section C compares the benefits and drawbacks of applying each approach to active trading.

A. Decision Tree Results

Fig. 7 displays the average performance metrics for the decision tree model's performance across each of the nine days used as a test set.

Accuracy	73.18%
Macro-average Precision	42.61%
Macro-average Recall	42.85%
Macro-average F1	42.66%
Sell Prediction (=0) Precision	11.36%
Sell Prediction (=0) Recall	12.68%
Buy Prediction (=2) Precision	31.35%
Buy Prediction (=2) Recall	31.89%

Fig. 7. Average decision tree model performance metrics from 9-fold cross validation.

Macro-averages were chosen to reflect equal class importance for active trading. Precision is useful for evaluating *whether* the model is profitable, and recall is useful for determining the *proportion of* profit opportunities it takes advantage of. For profitability, class-specific precision for classes 0 and 2 are more important than that for class 1, as false positives lead to unprofitable trades. Class-specific recall for classes 0 and 2 indicate the proportion of sell and buy opportunities respectively the model predicts correctly. A further review of the normalised confusion matrix for all 9 test days in Fig. 8 visualises the problem of class imbalance for making predictions using the decision tree model.

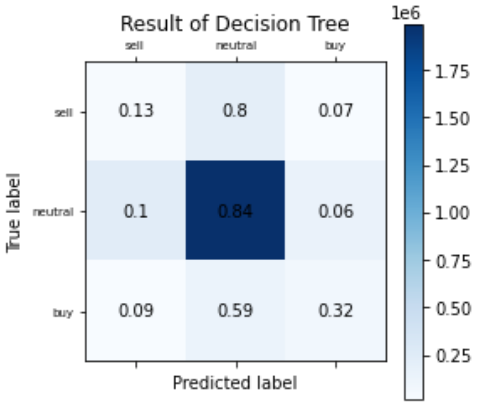

Fig. 8. Decision tree model normalised confusion matrix for performance on all 9 test sets.

The model does well at predicting the majority class (=1), with a recall score of 83.89%. However, it achieves precision and recall scores for the buy class (=2) of <35% and precision and recall scores of <13% for the sell class (=0). Decision tree models are notoriously poor at predicting minority classes due to the prioritising of the majority class during tree-building.

Using the model predictions, profits generated by the trading algorithm described in Section III F can be seen in Fig. 9. Despite poor classification performance, the algorithm was consistently profitable throughout the period, making a positive total profit for every trading day except day 4 for x = 5. As expected, higher values for the minimum spread yielded increasing variability in daily profit, with x = 5 achieving the maximum or minimum total profit for 78% of trading days. Considering a trader's risk tolerance, values for x of 1, 2, or 3 are all viable options. Values of 4 and 5 do not yield superior profits, but still experience higher standard deviation of profits.

Day	**Profit with Trading Algorithm Min *s(t)* Condition (*x*)**				
	x = 1	x = 2	x = 3	x = 4	x = 5
1	9,816	14,676	15,616	16,771	17,916
2	5,758	6,748	6,821	6,646	6,950
3	5,383	9,071	10,641	10,760	12,127
4	4,185	2,849	1,577	380	-1,299
5	6,186	7,790	8,040	6,760	6,188
6	4,652	4,212	2,180	524	56
7	6,584	8,601	8,059	8,135	7,817
8	4,529	6,497	7,094	6,639	7,277
9	7,541	11,539	13,826	13,860	16,018
Total	**54,633**	**71,984**	**73,853**	**70,475**	**73,050**
Average	**6,070**	**7,998**	**8,206**	**7,831**	**8,117**
Std	**1,666**	**3,389**	**4,420**	**5,141**	**6,092**

Fig. 9. Decision tree model profit performance for each of the 9 test days and each value of x (minimum spread condition parameter used in the trading algorithm).

B. *Neural Network Results*

Fig. 10 displays the average performance metrics for the ANN's performance across each of the five days used as a test set. It is immediately apparent that the ANN outperforms the decision tree model for trade signal prediction using the LOB data. However, it suffers from similar problems relating to class imbalance, yielding a class-specific recall for the sell class (=0) of close to 0%.

Accuracy	85.93%
Macro-average Precision	76.07%
Macro-average Recall	58.04%
Macro-average F1	53.23%
Sell Prediction (=0) Precision	80.00%
Sell Prediction (=0) Recall	0%
Buy Prediction (=2) Precision	59.21%
Buy Prediction (=2) Recall	78.51%

Fig. 10. Average ANN performance metrics

The CNN model achieves a similarly high performance when predicting the majority class (=1) with an accuracy of 95%. However, the class-specific precision and recall scores for the buy class (=2) have improved significantly by 28% and 47% respectively. This implies that it will be more profitable than the decision tree for buy market order opportunities.

As can be seen in Fig. 11, the major limitation of the CNN model is that it fails to predict sell signals. This is once again due to imbalanced classes and the model struggles to distinguish a sell signal from a neutral signal. Down-sampling the majority class was applied to attempt to resolve this problem. However, the model fails to generalise well on the test data, instead predicting a large proportion of neutral instances as sell signals. This suggests that the features that characterise sell signals are very similar to those of neutral instances.

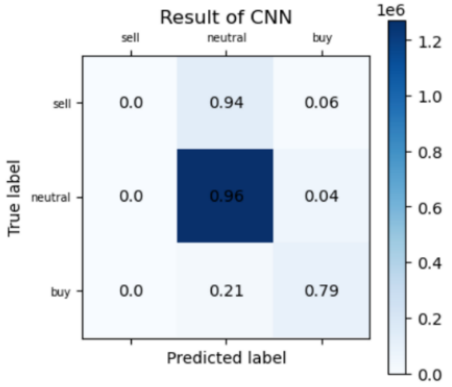

Fig. 11. ANN normalised conufsion matrix for performance on the test set.

Profits generated by applying the ANN and trading algorithm to unseen data can be seen in Fig. 12. Note that the results from the table are obtained by setting x to 3, the optimal value discovered in Fig. 9. Even though the model fails to predict sell signals, it consistently generates much higher profits than the decision tree model. On five days of test data, it yields an impressive average daily profit of 61,396.46.

Day	**Total Profit**
1	92,745.65
2	50,876.84
3	54,878.67
4	43,098.58
5	65,382.58
Total Profit	**306,982.32**
Average	**61,396.46**
Standard Deviation	**19,278.34**

Fig. 12. ANN profit performance for each of the 5 test days.

C. *Implications for Active Trading*

Although both models have problems that stem from class imbalances and consequently struggle to predict trading accurately, both generate consistently positive profit for each trading day tested. This implies that the decision to choose the target variable engineered in Section III over the frequently selected mid-price is effective for improving practicality of models for markets with low liquidity. Active traders can utilise this approach for profitable trading in real markets. Furthermore, state-of-the-art models and more sophisticated trading algorithms, are likely to achieve similar, if not better, results.

The ANN generates, on average, 800% more profit per trading day. This reflects its ability to model more complex relationships between features and the information lost in supervised feature extraction. Furthermore, the results demonstrate the trade-off between interpretability and performance. Although the decision tree model performs worse and generates inferior profits for unseen data, it makes highly interpretable decisions that can be explained by understandable, calculated features. More advanced Explainable AI (XAI) techniques may improve this trade-off, but ultimately optimal model complexity is determined by users' preferences and restraints.

VI. Further Work and Improvement

A. *Class Imbalance*

Both models suffered from class imbalance problems that could be solved using upsampling methods or class reweighting. Further work that applied these methods is likely to achieve better performance on unseen data.

B. *Computational Limitations*

Training the CNN was computationally expensive. To improve comparability and robustness of results, it would have been beneficial to use 9-fold cross validation on the same 9 days of LOB data used to test the decision tree model when testing the CNN's performance. Furthermore, performance be improved by utilising more trading days to train the model. Both these improvements were infeasible given time and computational limitations.

C. *Probabilistic Modelling*

Probabilistic modelling is a statistical approach that places a probability distribution over model parameters. Unlike frequentist approaches, probabilistic models use Bayesian inference to infer the parameters of the model given the observed data *and* prior knowledge. The result is the incorporation of uncertainty into model predictions.

Using a Bayesian decision tree, Probabilistic Artificial Neural Network (PANN), or other probabilistic model to make predictions would enable a trading algorithm to consider prediction uncertainty and make better-informed trading decisions accordingly. This is likely to lead to more profitable trading outcomes than those described in this report.

D. Reinforcement Learning

Reinforcement learning is a relatively new agent-based approach to LOB price prediction. An agent learns to predict the future price by taking actions that modify the limit order book and observing the resulting rewards. The rewards are based on the difference between the predicted price and the actual price, and the agent uses these rewards to update its policy for taking actions. Future work could use the experimental setup described in this report in combination with reinforcement learning models, like the S3C model, to investigate whether they improve profitability.

VII. Conclusions

Existing techniques for price prediction using LOB data make the unrealistic assumption that trades can be executed at the mid-price. This assumption is invalidated for markets with low liquidity. High bid-ask spread volatility in these markets means that models trained to predict mid-price changes cannot be applied profitably to active trading scenarios. This report describes methods used to engineer a target variable that considers the quoted best bid and ask prices, thus overcoming these challenges.

Two approaches to modelling this variable were proposed. 1) A decision tree model using highly interpretable handcrafted features, including the OFI, OBI, AVD, and $s(t)$, and 2) an ANN with LSTM and CNN layers, which uses 10 previous states of the LOB to make its prediction. Despite problems with class imbalance that caused suboptimal classification performance, both achieved profitable outcomes when combined with a simple trading algorithm that considered market liquidity and made realistic inventory assumptions.

There is increasing importance placed on model interpretability by regulators and finance professionals and one method for achieving this is to use less complex models with interpretable features. Although the handcrafted features described in this report provide some information for predicting trade signals, the ANN generated on average 800% more profit per trading day than the decision tree model. This implies that price prediction using LOB data is an inherently complex problem, and to achieve the highest profitability, statistical techniques that can model this complexity are required.

The results suggest that adjustments to experimental set up, when combined with state-of-the-art modelling and trading algorithms conditioned on market liquidity, can overcome the practicality issues associated with current methods. Future work can build on the methods and findings described in this report and apply them to active trading scenarios.

References

[1] Cartea, Ivaro, Donnelly, R. F., & Jaimungal, S. (2015). Enhancing Trading Strategies with Order Book Signals. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.2668277

[2] Cont, R., Kukanov, A., & Stoikov, S. (2013). The Price Impact of Order Book Events. Journal of Financial Econometrics, 12(1), 47–88. https://doi.org/10.1093/jjfinec/nbt003

[3] Deloitte. (2022, May 17). Unleashing the power of machine learning models in banking through explainable artificial intelligence (XAI). Deloitte Insights. https://www2.deloitte.com/uk/en/insights/industry/financial-services/explainable-ai-in-banking.html

[4] European Central Bank. (2019, February 13). Algorithmic trading: trends and existing regulation. European Central Bank - Banking Supervision. https://www.bankingsupervision.europa.eu/press/publications/newsletter/2019/html/ssm.nl190213_5.en.html

[5] Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. Quantitative Finance, 13(11), 1709–1742. https://doi.org/10.1080/14697688.2013.803148

[6] Xu, K., Gould, M., & Howison, S. (2019). Multi-Level Order-Flow Imbalance in a Limit Order Book. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3479741

[7] Zaznov, I., Kunkel, J., Dufour, A., & Badii, A. (2022). Predicting Stock Price Changes Based on the Limit Order Book: A Survey. Mathematics, 10(8), 1234. https://doi.org/10.3390/math10081234

Github repository: https://github.com/zepingchen/dsmp-hsbc-13