Imperial College London

11 June 2024

Forecasting Returns for High Frequency Cryptocurrency WebSocket Data

MSci Thesis Presentation

Isaac Lee

Imperial College London

Contents

Introduction

Summary

- Predict high frequency returns
- Introduce novel dataset
- Explore results for different models, trading pairs and prediction horizons

Centralized Markets

- Centralized vs. Decentralized
- Advantage: better regulation
- Disadvantage: must trust a third party

The Orderbook

- Used to organize buying and selling activity
- Different order types, limit vs. market
- Snapshot of supply and demand
- Supply and demand drives price, Mankiw [2014]

Imperial College London

Simplified Orderbook Visualization

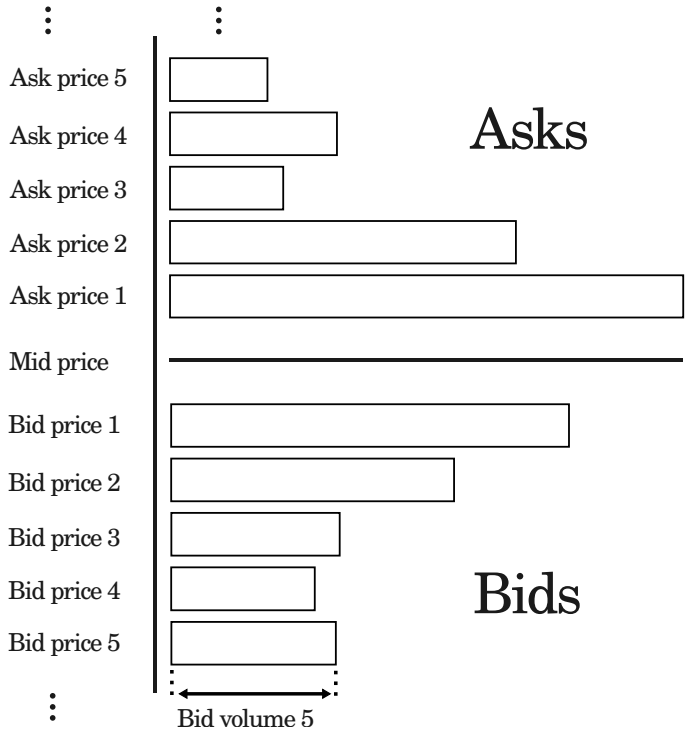

Figure 1: A simplified visualization of a limit orderbook.

Related Works

- Early methods were parametric, e.g. linear regressions, Cao et al. [2009], or Hawkes processes, Bacry and Muzy [2013]
- Explosion in popularity of deep learning
- DeepLOB, Zhang et al. [2019], DeepOF, Kolm et al. [2023], DeepVOL, Lucchese et al. [2024]
- Relatively few papers for crypto data
- Akyildirim et al. [2023] use traditional ML, Shin et al. [2021] use LSTM models for Bitcoin futures prediction
- Both for longer time horizons of minutes, hours, days

Contribution

- Application to novel cryptocurrency data.
- Prediction horizons ranging from ≈ 500 milliseconds to ≈ 20 seconds for Binance perpetual futures limit orderbook WebSocket data
- Previous similar literature, Zhang et al. [2019], Kolm et al. [2023], Lucchese et al. [2024], was for NASDAQ data
- Also the use of XGBoost, Chen and Guestrin [2016], for high frequency orderbook modelling.

Orderbook Data

Data Source

- Binance, worlds largest centralized cryptocurrency exchange
- WebSocket, Fette and Melnikov [2011], API endpoint: `wss://fstream.binance.com/ws/{tradingpair}@depth{L}@100ms`
- Logged to SQL database on AWS instance
- Data ranges from 2024-02-05 13:07:53.664 to 2024-02-25 11:18:50.866
- Between 11 to 13 million rows of data per trading pair
- Trading pairs: BTCUSDT, ETHUSDT, MATICUSDT, SOLUSDT

Imperial College London

Example Data

Row	timestamp	ask_price_1	...	ask_price_10	bid_price_1	...	bid_price_10	ask_qty_1	...	ask_qty_10	bid_qty_1	...	bid_qty_10
0	1707138494390	0.7877	...	0.7886	0.7876	...	0.7867	9368.0	...	58006.0	15095.0	...	44990.0
1	1707138494515	0.7877	...	0.7886	0.7876	...	0.7867	9367.0	...	58006.0	15095.0	...	84785.0
2	1707138494760	0.7877	...	0.7886	0.7876	...	0.7867	9367.0	...	58006.0	15095.0	...	44990.0
3	1707138494881	0.7877	...	0.7886	0.7876	...	0.7867	9367.0	...	58006.0	15095.0	...	44990.0
4	1707138495248	0.7877	...	0.7886	0.7876	...	0.7867	9356.0	...	58006.0	15099.0	...	44990.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
11294483	1708857980137	0.9726	...	0.9735	0.9725	...	0.9716	16199.0	...	64495.0	20555.0	...	37554.0
11294484	1708857980246	0.9726	...	0.9735	0.9725	...	0.9716	16199.0	...	61163.0	19527.0	...	37554.0
11294485	1708857980360	0.9726	...	0.9735	0.9725	...	0.9716	20199.0	...	61163.0	19527.0	...	37554.0
11294486	1708857980494	0.9726	...	0.9735	0.9725	...	0.9716	21610.0	...	61163.0	19577.0	...	37554.0
11294487	1708857980602	0.9726	...	0.9735	0.9725	...	0.9716	22992.0	...	61163.0	18401.0	...	37554.0

Table 1: Example data for MATICUSDT.

Imperial College London

Data Irregularities

	BTCUSDT Δt ms	ETHUSDT Δt ms	MATICUSDT Δt ms	SOLUSDT Δt ms
count	12891080	13612890	11294487	13334051
median	116	114	122	115
mean	133	126	152	129
std	50	41	82	47
min	3	3	3	3
25%	107	105	108	107
50%	116	114	122	115
75%	142	133	162	135
max	61604	61862	61697	61815

Table 2: Summary statistics for the time difference between observations in ms, Δt ms, for each trading pair.

Imperial College London

Orderbook Shape

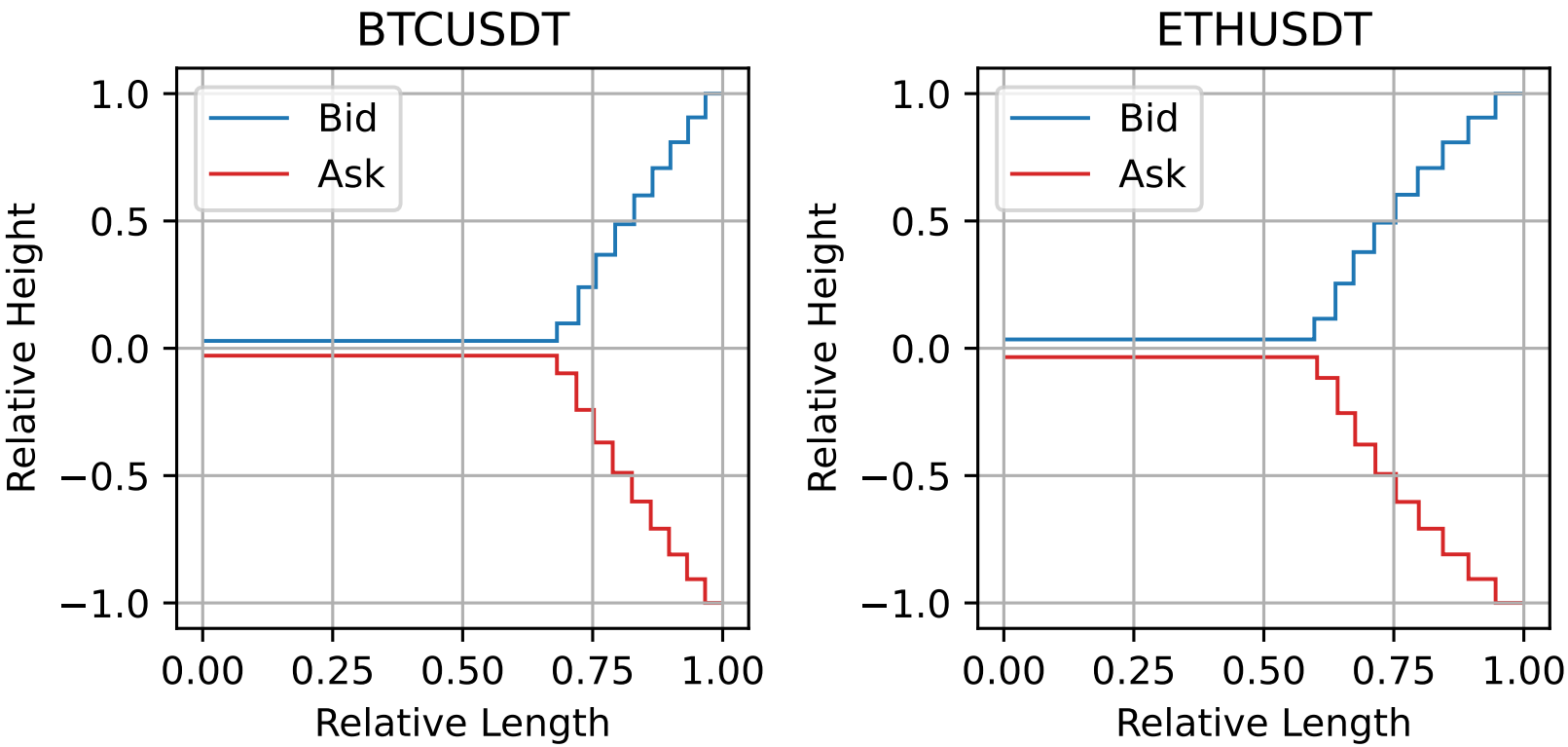

Figure 2: Average orderbook shape for BTCUSDT and ETHUSDT.

Imperial College London

Orderbook Shape

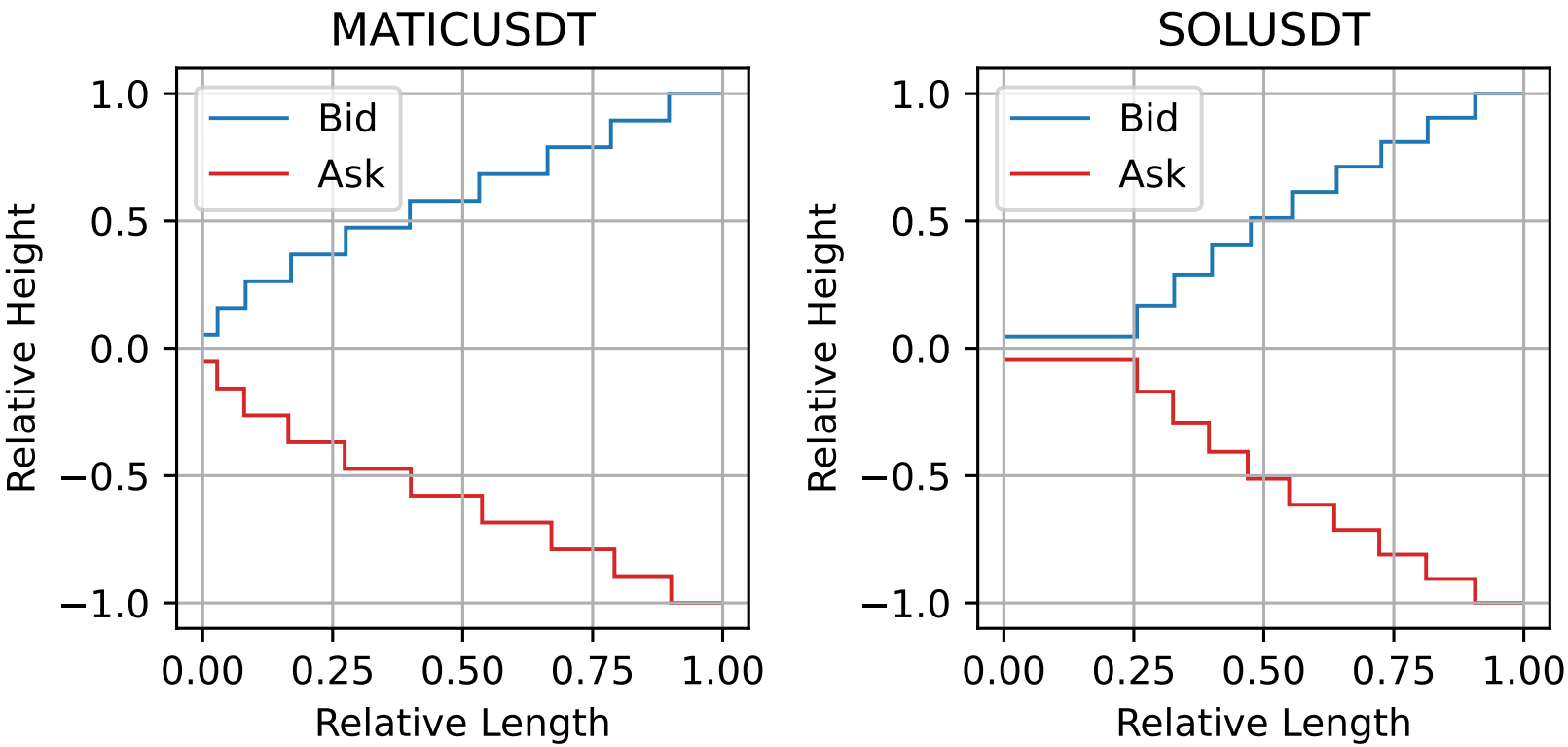

Figure 3: Average orderbook shape for MATICUSDT and SOLUSDT.

Modelling Objective

The Modelling Objective

- Aim: predict direction of future smoothed mid-price
- Three class classification problem

Smoothed Returns

Introduced in in Tsantekidis et al. [2017]. Define the quantites:

$$m_-(t) := \frac{1}{k}\sum_{i=0}^{k} p_0(t-i) \tag{1}$$

$$m_+(t) := \frac{1}{k}\sum_{i=1}^{k} p_0(t+i) \tag{2}$$

$$\ell(t) := \frac{m_+(t) - m_-(t)}{m_-(t)} \tag{3}$$

- $m_-(t)/m_+(t)$: average mid-price for the previous (inclusive)/next k observations

Discretizing Smoothed Returns

Convert to a classification problem by discretizing returns:

$$y(t) = \begin{cases} +1 & \text{if } \ell(t) \in (\epsilon, \infty) \\ 0 & \text{if } \ell(t) \in [-\epsilon, \epsilon] \\ -1 & \text{if } \ell(t) \in (-\infty, -\epsilon) \end{cases} \tag{4}$$

Choice of ϵ discussed later.

Example Labelled Data

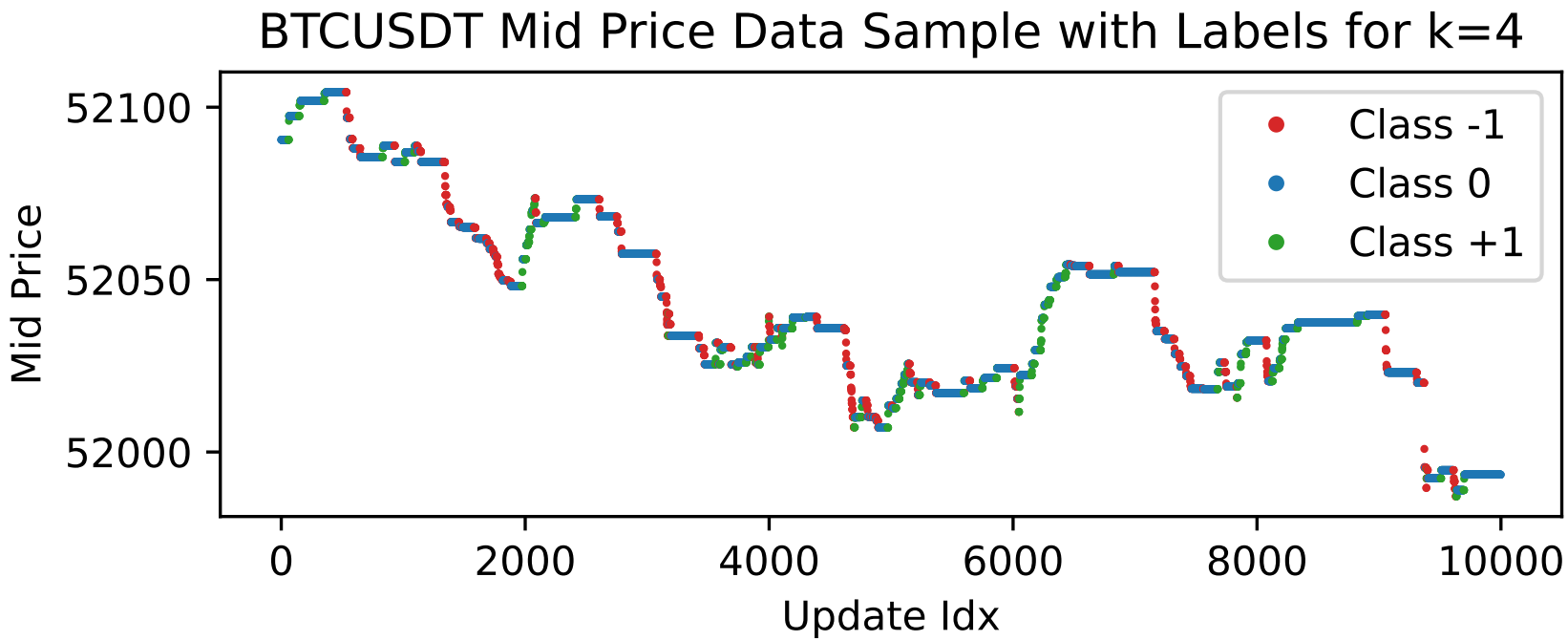

Figure 4: An example BTCUSDT mid-price sample, coloured according to class label.

Model Theory

Logistic Regression

- Introduced in Cox [1958].
- Key idea: map the output of a linear model onto the interval [0, 1] using the logistic function:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{5}$$

- Treat these values as probabilities and select the class with the highest probability.

Decision Trees

- Optimal data partitioning using a tree datastructure
- Tree is built iteratively by selecting the best feature and threshold to split on at each node
- Greedy approach
- For classification, class label is the mode of the training samples in the leaf node

Imperial College London

Example Decision Tree

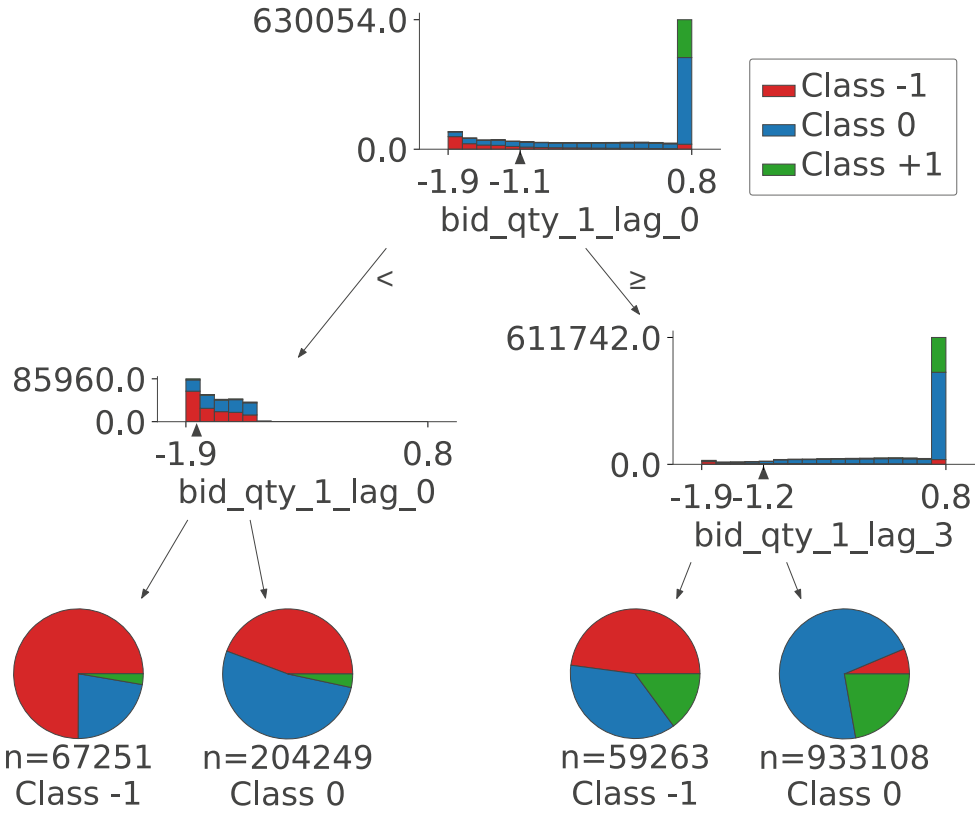

Figure 5: A visualization of a single decision tree.

Boosting and Boosted Decision Trees

- Simple decision trees have high variance, leading to overfitting
- *Boosting*, Hastie et al. [2001], is an ensemble method used to solve this problem
- Basic idea: iteratively train many *weak learners* and then combine their predictions to produce a single, more robust prediction.
- Boosting lowers the model bias, whilst keeping variance low.

Neural Networks

- Most famous example, the MLP, Rumelhart. et al. [1986].
- Affine and non-linear projections:

$$f_i(x; W_i, b_i) := \sigma_i(W_i x + b_i)$$
$$f(x; W_1, W_2, ..., W_N, b_1, b_2, ..., b_N) := f_N \circ f_{N-1} \circ ... \circ f_2 \circ f_1(x)$$

- Hornik et al. [1989] shows, they are *universal function approximators*

Neural Networks

- Loss function:

$$\mathcal{L}(\theta; x, y) := \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta), y_i) \tag{6}$$

- Gradient descent via *backpropagation*, Rumelhart. et al. [1986]:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta^{(t)}) \tag{7}$$

- In practice non-convex local minima, so other methods e.g. *stochastic gradient descent*, Robbins [1951], with *momentum*, used.

Convolutional Neural Networks

- Introduced in LeCun et al. [1998]
- Specialized class of NNs designed for processing structured grid data, such as images.
- Key advantages: local receptive fields, shared weights and spatial subsampling, Kauderer-Abrams [2017].
- Convolutional layer applies a convolution operation across previous layer:

$$
\begin{aligned}
h_{ij} &= \sigma\left((W * x)_{ij} + b\right) \\
(W * x)_{ij} &:= \sum_{m=-k}^{k} \sum_{n=-k}^{k} W_{mn} x_{i-m,j-n}
\end{aligned}
\tag{8}
$$

Convolution Visualized

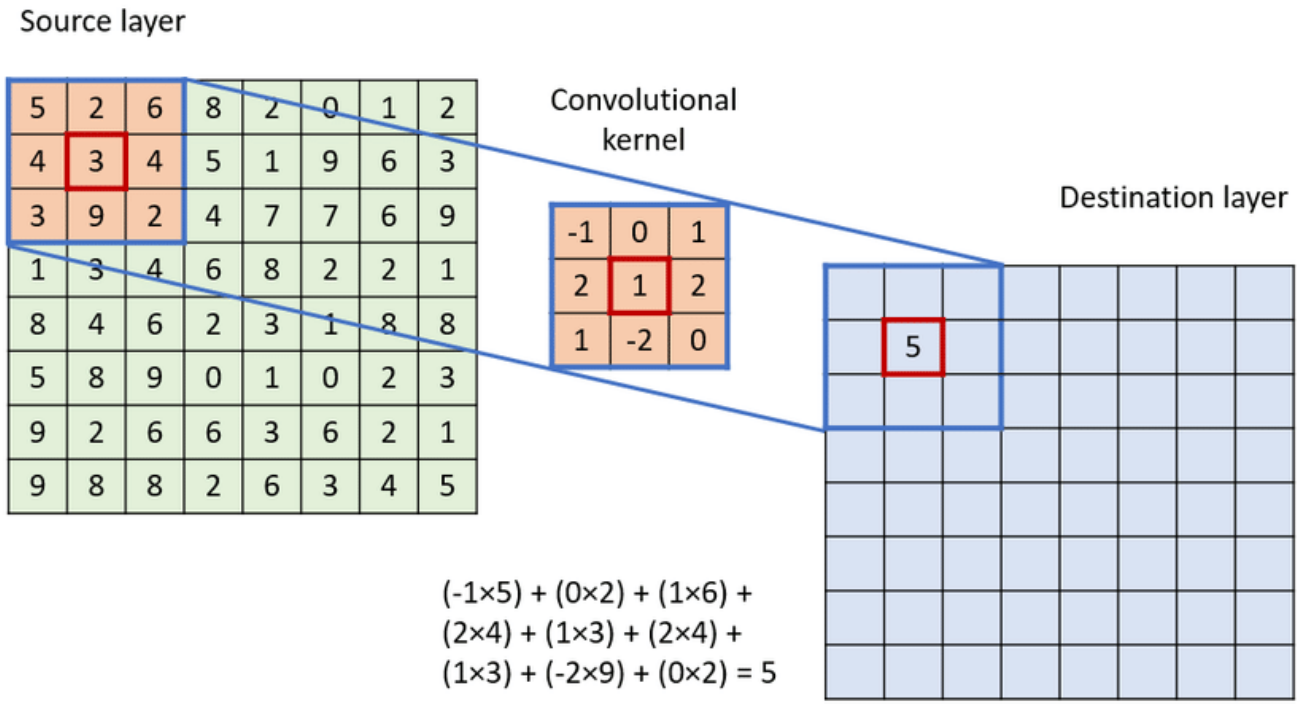

Figure 6: Example convolutional fitler applied to a 2D array. Image source: Podareanu et al. [2019].

Recurrent Neural Networks

- RNNs are NNs that take in sequences of input iteratively.
- At each iteration: take in input and previous hidden state, update hidden state then output
- Famous example, the Jordan RNN, Jordan [1997]:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{9}$$

Recurrent Neural Networks

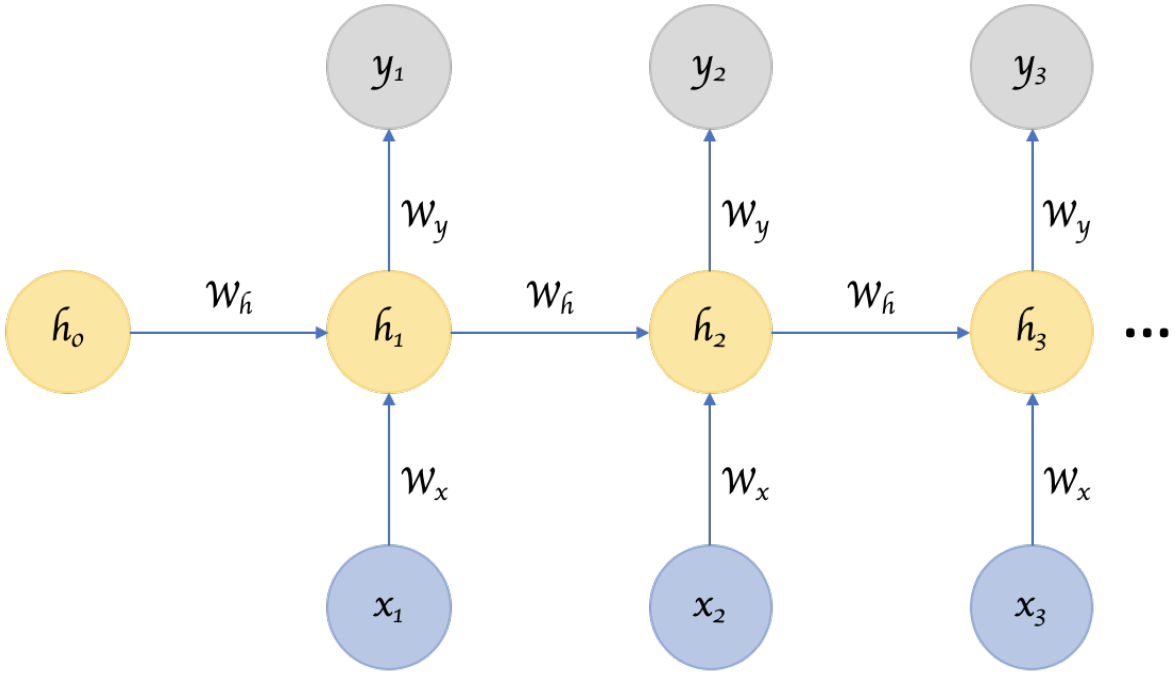

Figure 7: A visualization of an RNN. Image source: Venkatachalam [2019]

Long Short-Term Memory

- Vanilla RNNs suffer from *vanishing gradients*, Pascanu et al. [2013]
- The LSTM, Hochreiter and Schmidhuber [1997] was devised to address this
- Main ideas: input gate, output gate, forget gate and cell
- Cell learns information and other gates regulate flow of information to the cell
- A modification of an RNN with more hidden state for greater control of what information is learnt

Long Short-Term Memory

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
\tag{10}
$$

where $\odot$ denotes the element-wise product. Hochreiter and Schmidhuber [1997].

Orderbook Features

The Raw Limit Orderbook Representation

Zhang et al. [2019] use the raw orderbook data for their model input i.e. $\boldsymbol{x}_t$ takes the form:

$$\boldsymbol{x}_t = [p^A_{t,\ell}, q^A_{t,\ell}, p^B_{t,\ell}, q^B_{t,\ell}]_{\ell=1}^{L} \in \mathbb{R}^{4L} \tag{11}$$

The Orderflow Representation

Orderflow is a stationary transformation of the the raw orderbook data introduced in Cont et al. [2013]. Define the contribution of the n^{th} event at level ℓ on the (A)sk/(B)id side as:

$$e^A_{t,\ell} := I_{\{p^A_{n,\ell} \leq p^A_{n-1,\ell}\}} q^A_n - I_{\{p^A_{n,\ell} \geq p^A_{n-1,\ell}\}} q^A_{n-1,\ell} \tag{12}$$

$$e^B_{t,\ell} := I_{\{p^B_{n,\ell} \geq p^B_{n-1,\ell}\}} q^B_{n,\ell} - I_{\{p^B_{n,\ell} \leq p^B_{n-1,\ell}\}} q^B_{n-1,\ell} \tag{13}$$

Then we define the Orderflow feature vector as:

$$\boldsymbol{x}_t := [e^A_{t,\ell}, e^B_{t,\ell}]^L_{\ell=1} \in \mathbb{R}^{2L} \tag{14}$$

lrLOB & lrOF

- For LOB/OF features concatenate with lagged features:

$$\boldsymbol{x}_t^{AR} = [\boldsymbol{x}_t; \boldsymbol{x}_{t-1}; \boldsymbol{x}_{t-2}; \cdots ; \boldsymbol{x}_{T-1}] \in \mathbb{R}^{dT} \tag{15}$$

- lrLOB: logistic regression model with raw limit orderbook AR features
- lrOF: logistic regression model with orderflow AR features
- $T := 10$ (memory constraints)

xgbLOB & xgbOF

- XGBoost first introduced in Chen and Guestrin [2016]
- Efficient implementation of Boosted Decision Trees
- xgbLOB: XGBoost model with raw limit orderbook AR features
- xgbOF: XGBoost model with orderflow AR features
- $T := \min(20, k)$ (memory constraints)

xgbLOB & xgbOF Hyperparameters

- **max depth** = 10. The max depth of each weak learner tree.
- **eta** = 0.1. Step size shrinkage used in update to prevent overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.
- **data subsample ratio** = 0.8. The proportion of the training data that each weak learner can use.
- **column subsample ratio** = 0.8. The proportion of the features available when performing splits.
- **evaluation metric** = Multiclass classification error rate. The metric used during the boosting algorithm to determine weights.

Imperial College London

DeepLOB

- First introduced in Zhang et al. [2019]
- CNN-LSTM architecture
- Input $X_t \in \mathbb{R}^{T \times 4L}$, $X_t :=$

$$\begin{bmatrix} p^A_{t-T,1} & q^A_{t-T,1} & p^B_{t-T,1} & q^B_{t-T,1} & \cdots & p^A_{t-T,L} & q^A_{t-T,L} & p^B_{t-T,L} & q^B_{t-T,L} \\ p^A_{t-(T-1),1} & q^A_{t-(T-1),1} & p^B_{t-(T-1),1} & q^B_{t-(T-1),1} & \cdots & p^A_{t-(T-1),L} & q^A_{t-(T-1),L} & p^B_{t-(T-1),L} & q^B_{t-(T-1),L} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ p^A_{t-1,1} & q^A_{t-1,1} & p^B_{t-1,1} & q^B_{t-1,1} & \cdots & p^A_{t-1,L} & q^A_{t-1,L} & p^B_{t-1,L} & q^B_{t-1,L} \\ p^A_{t,1} & q^A_{t,1} & p^B_{t,1} & q^B_{t,1} & \cdots & p^A_{t,L} & q^A_{t,L} & p^B_{t,L} & q^B_{t,L} \end{bmatrix} \tag{16}$$

DeepLOB Schematic

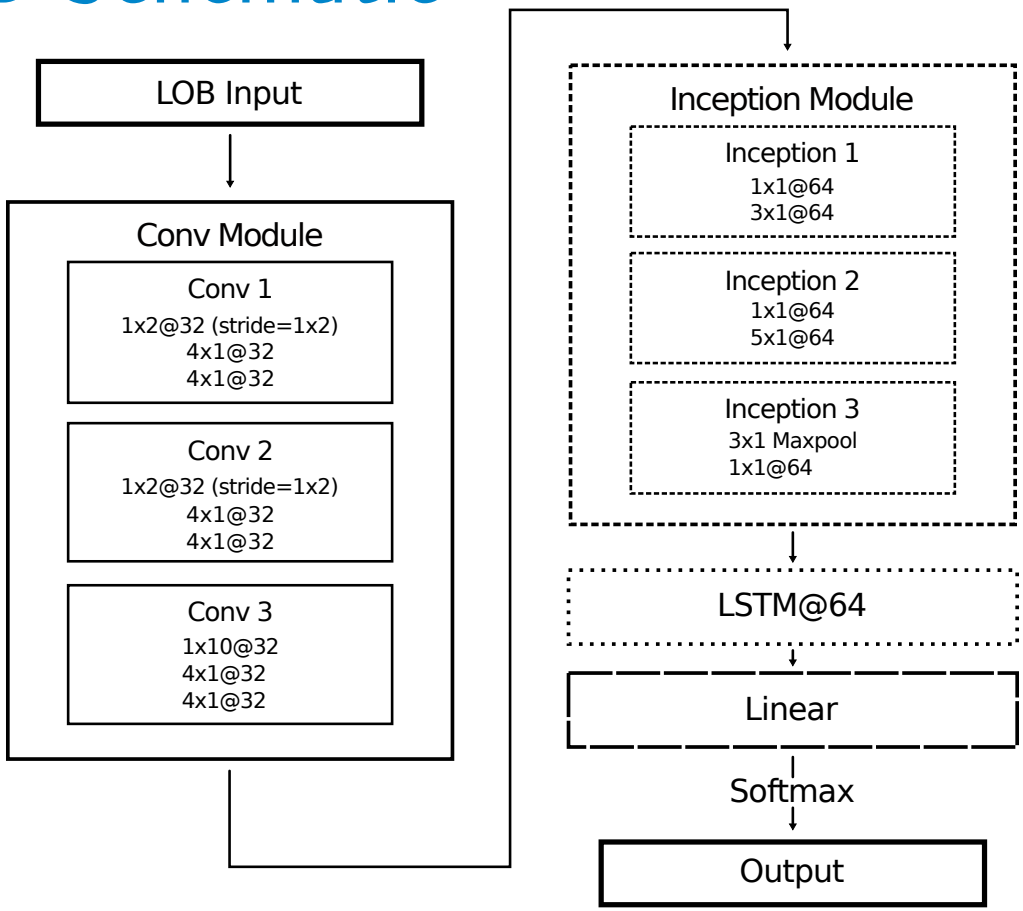

Figure 8: DeepLOB model schematic. Note: $1 \times 2@32$ means 32 1×2 convolutional filters.

DeepLOB Modules

- **Convolutional Module**
  - Conv 1 aggregates price and volume information for each level for each side
  - Conv 2 aggregates information across side for each level
  - Conv 3 aggregates information across the whole orderbook
- **Inception Module**
  - Increases dimensionality and simulates moving averages
  - Key idea: *Network-In-Network*, Lin et al. [2014]
- **LSTM Module**
  - Captures long term dependencies from inception output

DeepOF

- Introduced in Kolm et al. [2023]
- Modification of the DeepLOB architecture: uses orderflow input instead of raw orderbook
- Input:

$$X_t := \begin{bmatrix} e^A_{t-T,1} & e^B_{t-T,1} & \cdots & e^A_{t-T,L} & e^B_{t-T,L} \\ e^A_{t-(T-1),1} & e^B_{t-(T-1),1} & \cdots & e^A_{t-(T-1),L} & e^B_{t-(T-1),L} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e^A_{t-1,1} & e^B_{t-1,1} & \cdots & e^A_{t-1,L} & e^B_{t-1,L} \\ e^A_{t,1} & e^B_{t,1} & \cdots & e^A_{t,L} & e^B_{t,L} \end{bmatrix} \in \mathbb{R}^{T \times 2L}$$

Methodology

Sliding Window Setup

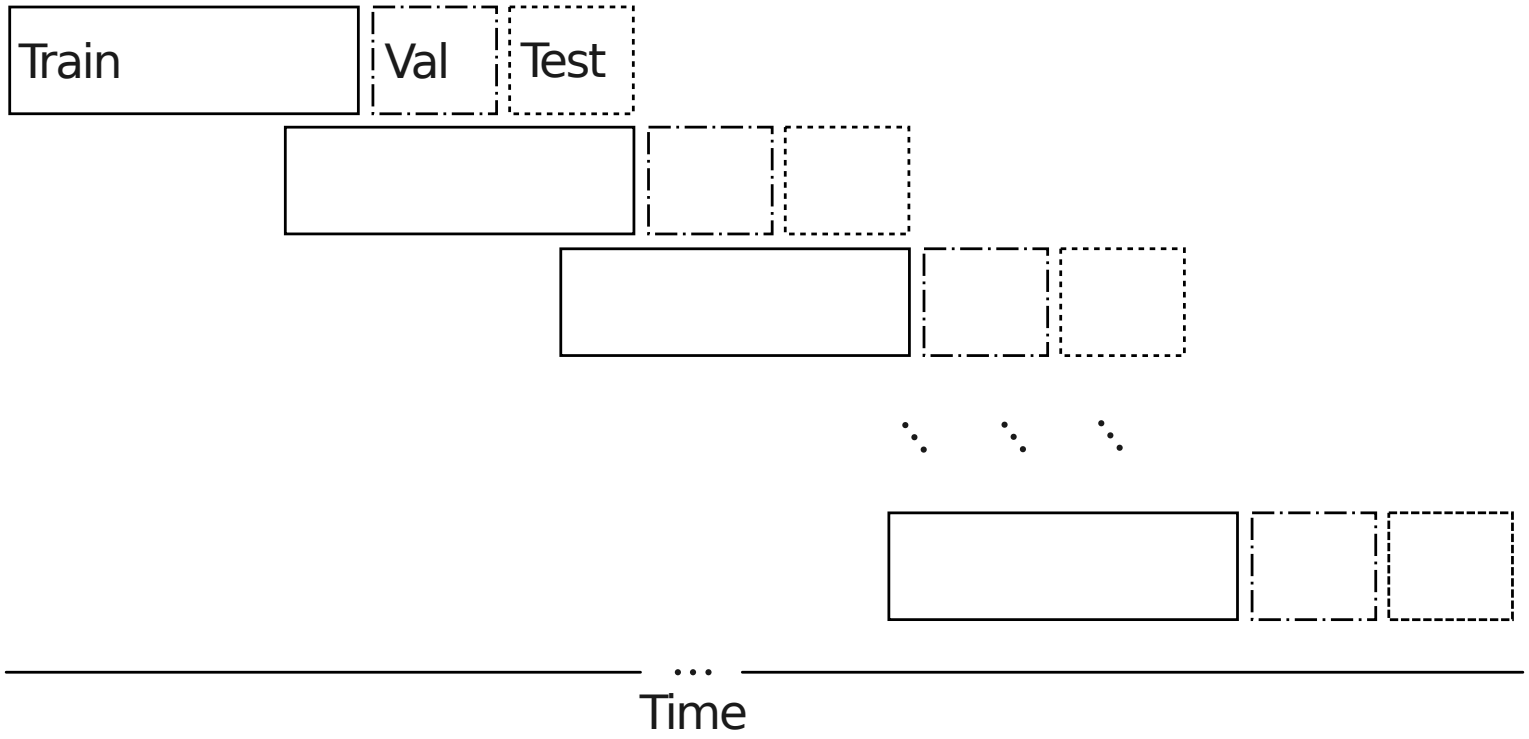

Figure 9: Train-val-test windows visualization. Note: train windows are 48 hours and val and test are 24 hours.

Data Normalization

- Calculate sample mean and standard deviation for each training window
- Scale train, val and test windows with:

$$\tilde{X}_{\text{train}} = \frac{X_{\text{train}} - \mu_{\text{train}}}{\sigma_{\text{train}}}$$

$$\tilde{X}_{\text{val}} = \frac{X_{\text{val}} - \mu_{\text{train}}}{\sigma_{\text{train}}}$$

$$\tilde{X}_{\text{test}} = \frac{X_{\text{train}} - \mu_{\text{test}}}{\sigma_{\text{train}}}$$

Choice of Label Discretization Parameter

- Following Lucchese et al. [2024], set discretization parameter, ϵ, so that classes are balanced
- For prediction horizon k, window w:

$$\epsilon := \epsilon_{k,w} := \frac{|\hat{F}^{-1}_{k,w}(\frac{1}{3})| + \hat{F}^{-1}_{k,w}(\frac{2}{3})}{2} \tag{17}$$

Model Training

- DeepLOB and DeepOF trained with PyTorch Paszke et al. [2017]
- Hardware: Nvidia RTX 3060 12GB, 64GB memory, 14 core 20 thread Intel i5 13600K
- Optimizer: ADAM, Kingma and Ba [2017]
- Loss: Cross Entropy Loss
- Trained until validation loss plateaued

Results

Classification Metrics

- Precision for class c is:

$$\text{Precision}_c := \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \tag{18}$$

"Out of all the times that we guessed class c, how many times were we correct?"

- Recall for class c is:

$$\text{Recall}_c := \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \tag{19}$$

"Out of all the times the true class was c, how many times did we predict it correctly?"

- F1-score for class c is harmonic mean of precision and recall:

$$\text{F1-score}_c := 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{20}$$

Imperial College London

Macro-Averages Averaged Across Windows, $k = 4$

accuracy	precision	recall	f1-score	support	model
0.87(0.03)	**0.91**(0.03)	**0.74**(0.01)	**0.80**(0.01)	647295(30003)	**xgbOF**
0.79(0.04)	0.75(0.05)	0.68(0.02)	0.70(0.01)	647295(30003)	xgbLOB
0.84(0.04)	0.87(0.02)	0.70(0.01)	0.76(0.01)	647289(30003)	lrOF
0.69(0.07)	0.66(0.08)	0.60(0.07)	0.59(0.06)	647289(30003)	lrLOB
0.86(0.03)	0.88(0.03)	**0.74**(0.01)	0.79(0.0)	647199(30003)	deepOF
0.75(0.2)	0.73(0.22)	0.70(0.04)	0.69(0.17)	647199(30003)	deepLOB

Table 3: Mean macro averaged BTCUSDT test set classification results across windows for $\boldsymbol{k = 4}$. (Standard deviations are given in parenthesis).

Imperial College London

Macro-Averages Averaged Across Windows, $k = 10$

accuracy	precision	recall	f1-score	support	model
0.80(0.03)	**0.84**(0.04)	0.75(0.01)	**0.78**(0.02)	647289(30003)	**xgbOF**
0.72(0.02)	0.71(0.02)	0.71(0.01)	0.71(0.01)	647289(30003)	xgbLOB
0.76(0.04)	0.81(0.03)	0.72(0.01)	0.74(0.02)	647289(30003)	lrOF
0.58(0.07)	0.62(0.05)	0.62(0.04)	0.57(0.08)	647289(30003)	lrLOB
0.79(0.03)	0.81(0.02)	**0.76**(0.01)	0.77(0.01)	647199(30003)	deepOF
0.74(0.06)	0.76(0.04)	0.71(0.05)	0.72(0.06)	647199(30003)	deepLOB

Table 4: Mean macro averaged BTCUSDT test set classification results across windows for $\boldsymbol{k = 10}$.

Imperial College London

Macro-Averages Averaged Across Windows, $k = 50$

accuracy	precision	recall	f1-score	support	model
0.64(0.04)	**0.65**(0.03)	0.62(0.01)	0.63(0.02)	647279(30003)	xgbOF
0.56(0.04)	0.57(0.03)	0.58(0.04)	0.53(0.06)	647279(30003)	xgbLOB
0.57(0.05)	0.62(0.03)	0.55(0.0)	0.55(0.02)	647289(30003)	lrOF
0.51(0.06)	0.52(0.03)	0.54(0.03)	0.46(0.06)	647289(30003)	lrLOB
0.66(0.03)	**0.65**(0.02)	**0.65**(0.02)	**0.65**(0.02)	647199(30003)	**deepOF**
0.57(0.07)	0.58(0.04)	0.57(0.07)	0.55(0.09)	647199(30003)	deepLOB

Table 5: Mean macro averaged BTCUSDT test set classification results across windows for $\boldsymbol{k = 50}$.

Imperial College London

Macro-Averages Averaged Across Windows, $k = 200$

accuracy	precision	recall	f1-score	support	model
0.52(0.04)	0.53(0.03)	0.50(0.01)	0.51(0.01)	647279(30003)	xgbOF
0.46(0.04)	0.45(0.02)	0.48(0.02)	0.40(0.04)	647279(30003)	xgbLOB
0.48(0.05)	0.50(0.02)	0.46(0.01)	0.45(0.01)	647289(30003)	lrOF
0.44(0.05)	0.45(0.04)	0.47(0.03)	0.39(0.04)	647289(30003)	lrLOB
0.56(0.03)	**0.56**(0.02)	**0.55**(0.02)	**0.55**(0.02)	647199(30003)	**deepOF**
0.49(0.05)	0.48(0.04)	0.49(0.06)	0.47(0.05)	647199(30003)	deepLOB

Table 6: Mean macro averaged BTCUSDT test set classification results across windows for $\boldsymbol{k = 200}$.

Imperial College London

BTCUSDT Window-Wise Macro Averages, $k = 4$

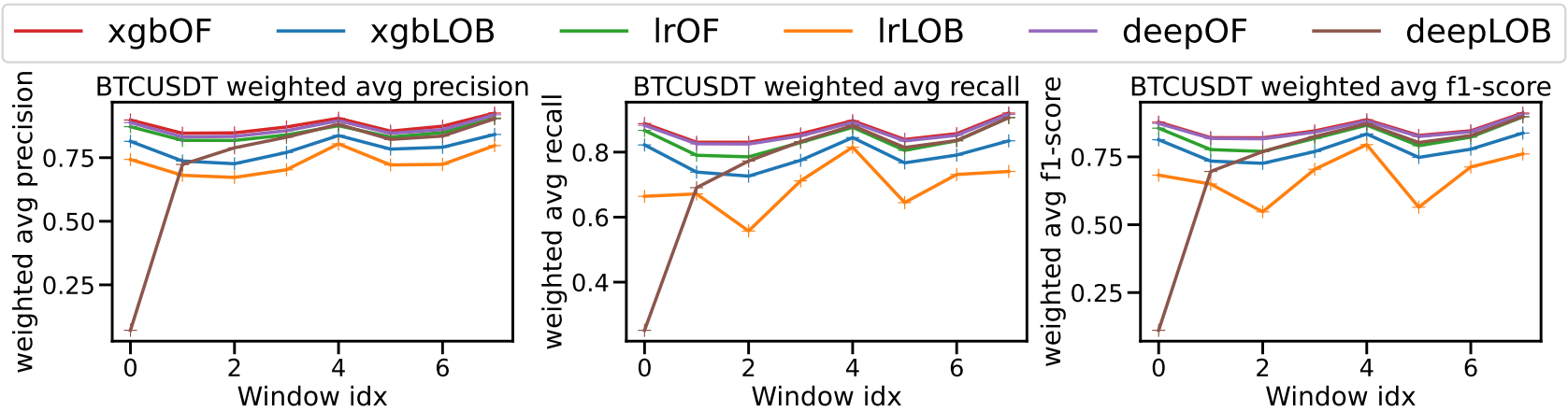

Figure 10: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 4$.

Imperial College London

ETHUSDT Window-Wise Macro Averages, $k = 4$

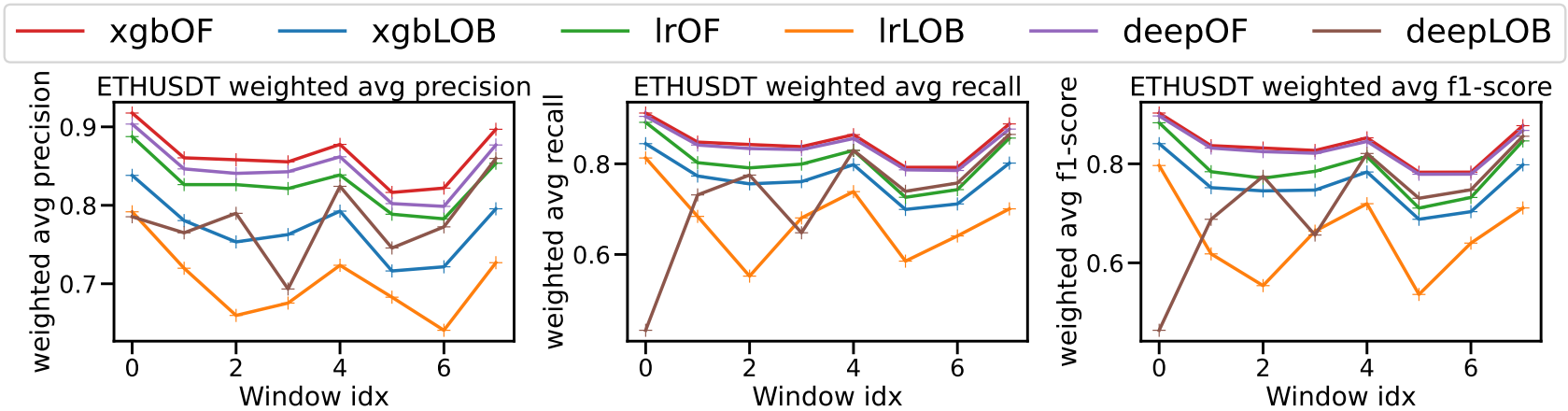

Figure 11: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 4$.

Imperial College London

MATICUSDT Window-Wise Macro Averages, $k = 4$

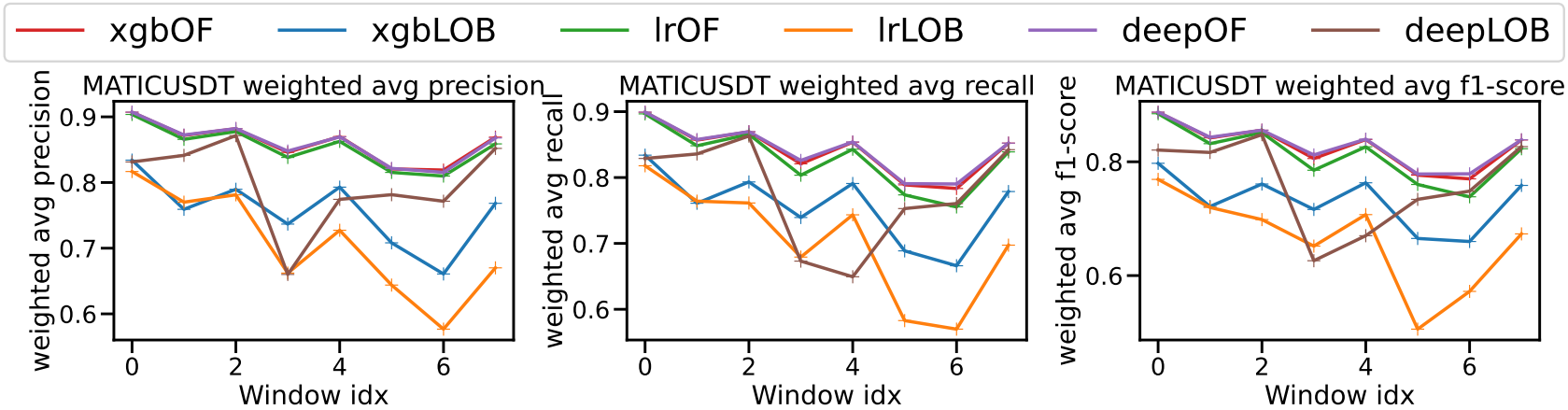

Figure 12: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 4$.

Imperial College London

SOLUSDT Window-Wise Macro Averages, $k = 4$

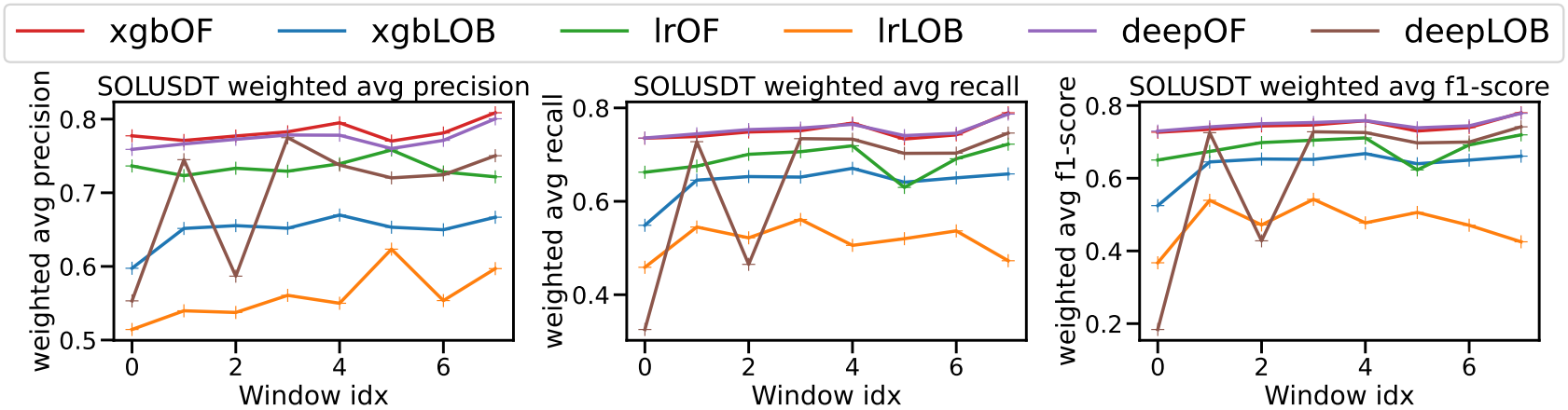

Figure 13: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 4$.

Imperial College London

BTCUSDT Window-Wise Macro Averages, $k = 10$

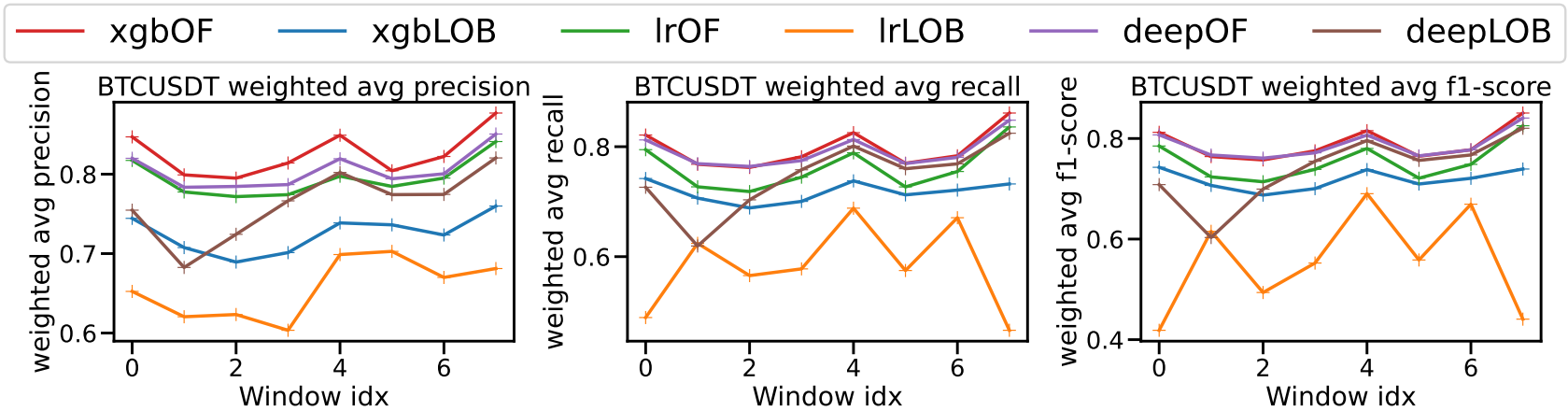

Figure 14: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 10$.

Imperial College London

ETHUSDT Window-Wise Macro Averages, $k = 10$

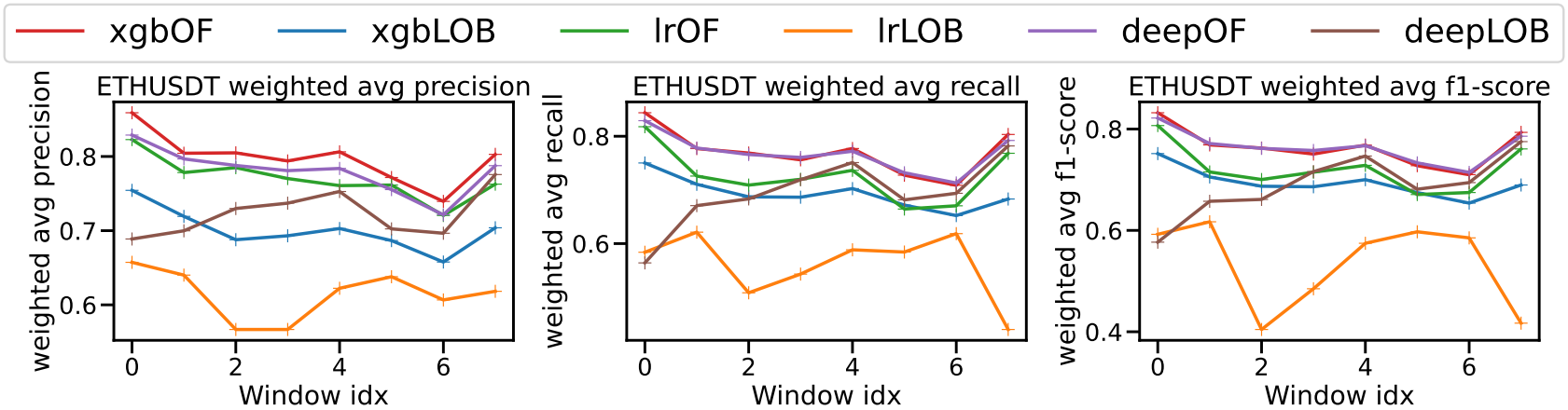

Figure 15: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 10$.

Imperial College London

MATICUSDT Window-Wise Macro Averages, $k = 10$

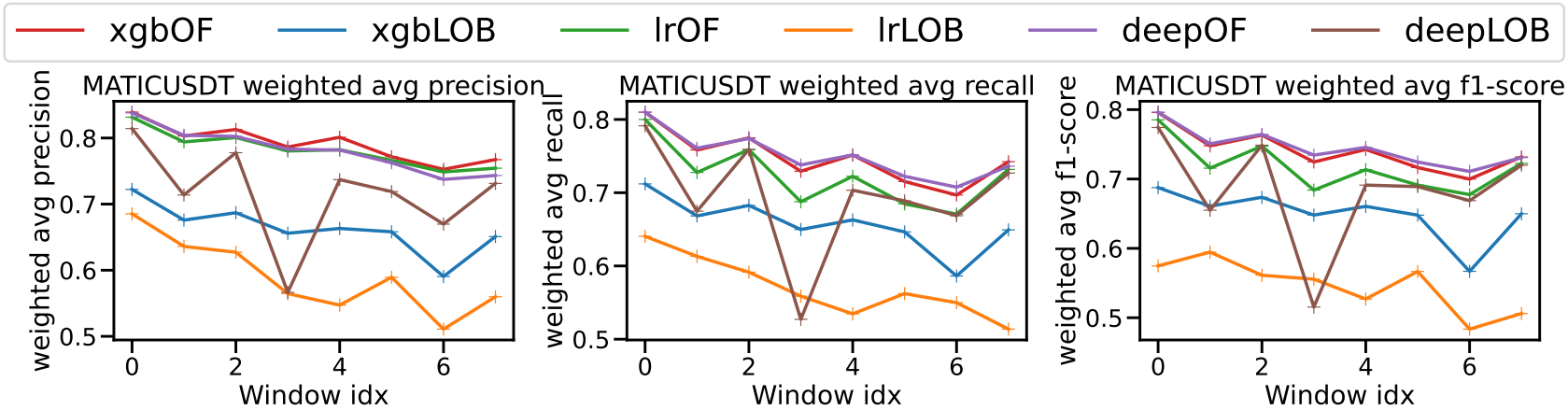

Figure 16: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 10$.

Imperial College London

SOLUSDT Window-Wise Macro Averages, $k = 10$

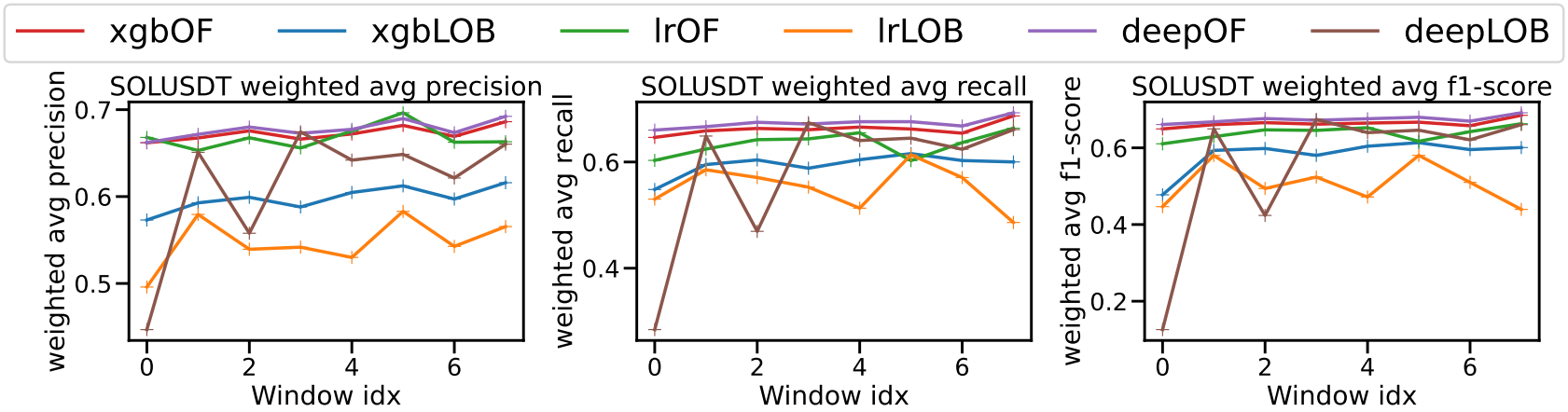

Figure 17: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 10$.

Imperial College London

BTCUSDT Window-Wise Macro Averages, $k = 50$

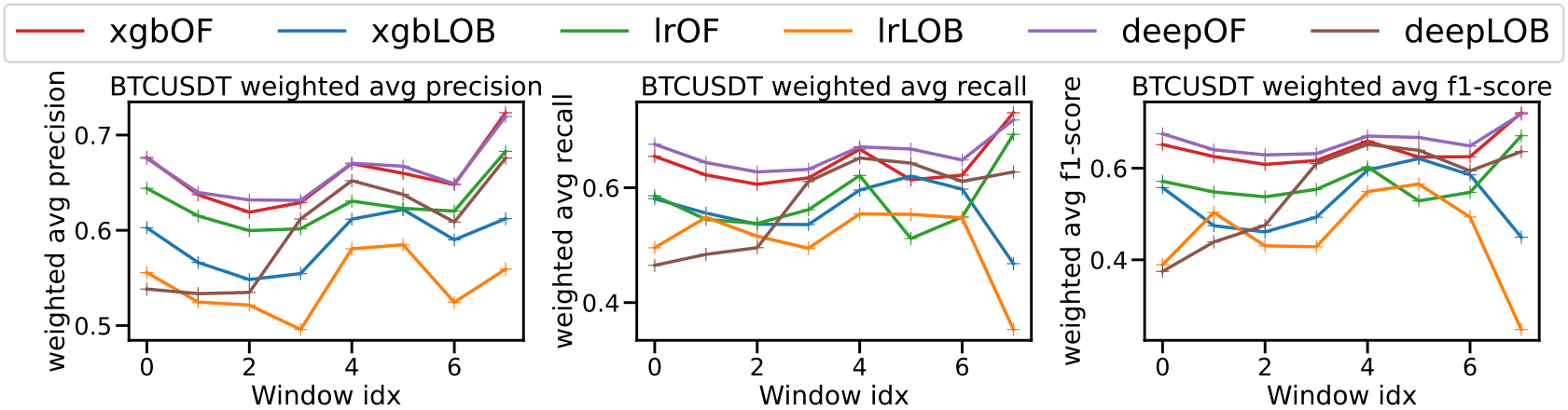

Figure 18: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 50$.

Imperial College London

ETHUSDT Window-Wise Macro Averages, $k = 50$

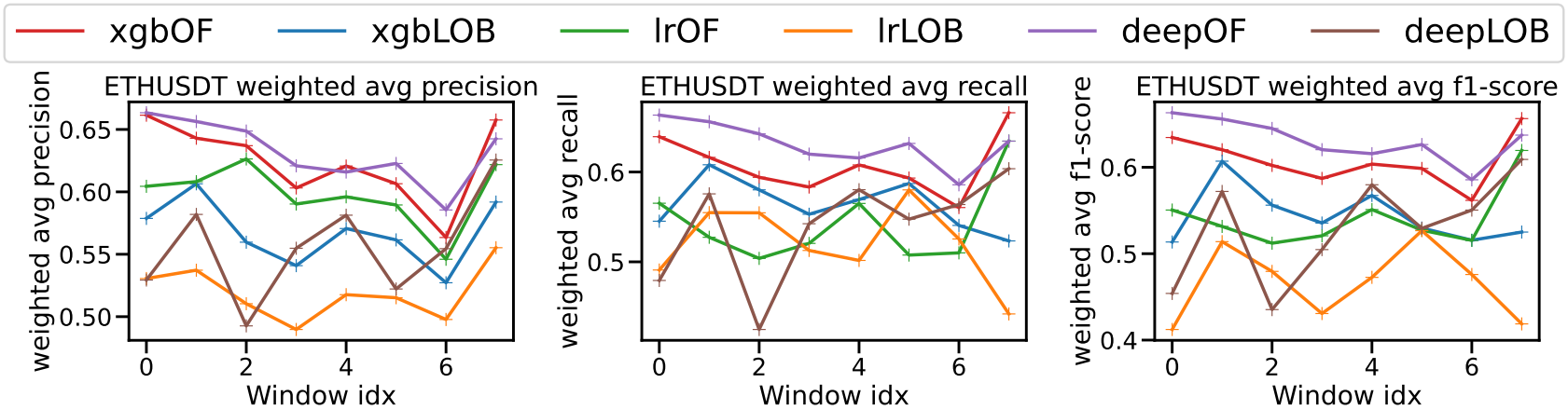

Figure 19: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 50$.

Imperial College London

MATICUSDT Window-Wise Macro Averages, $k = 50$

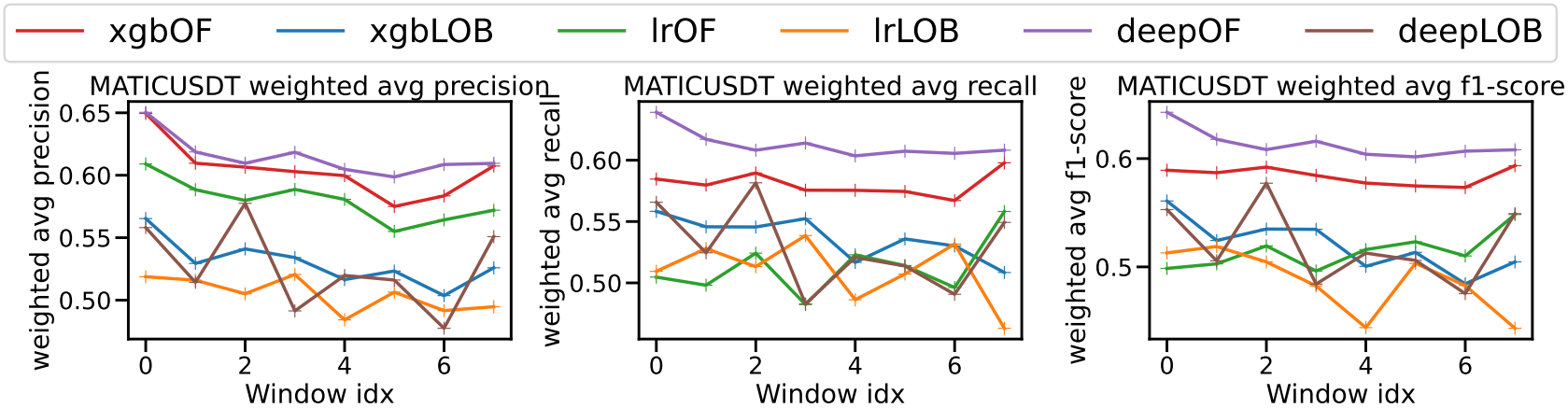

Figure 20: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 50$.

Imperial College London

SOLUSDT Window-Wise Macro Averages, $k = 50$

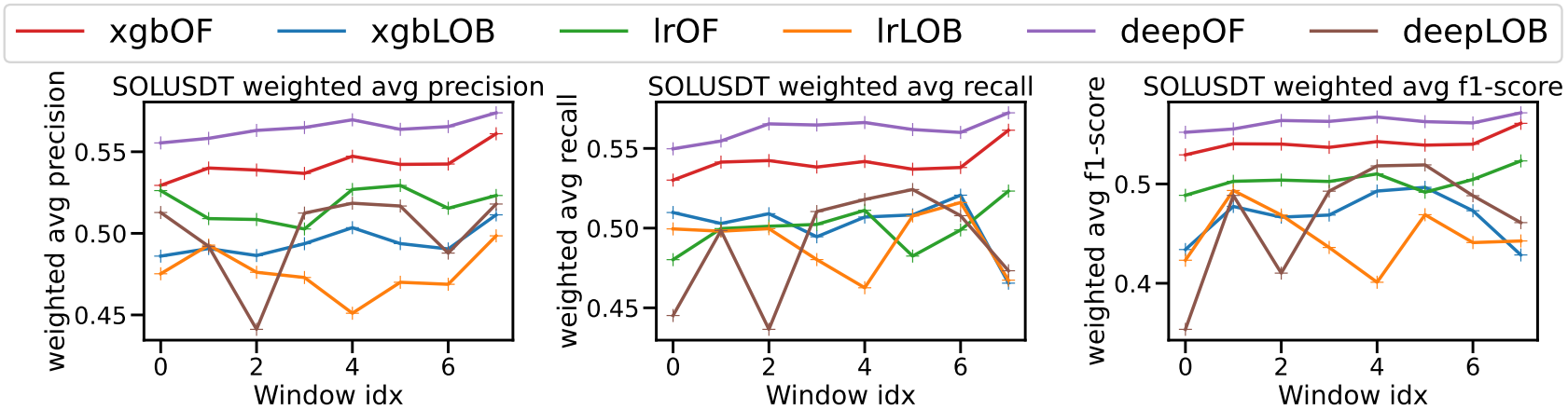

Figure 21: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 50$.

Imperial College London

BTCUSDT Window-Wise Macro Averages, $k = 200$

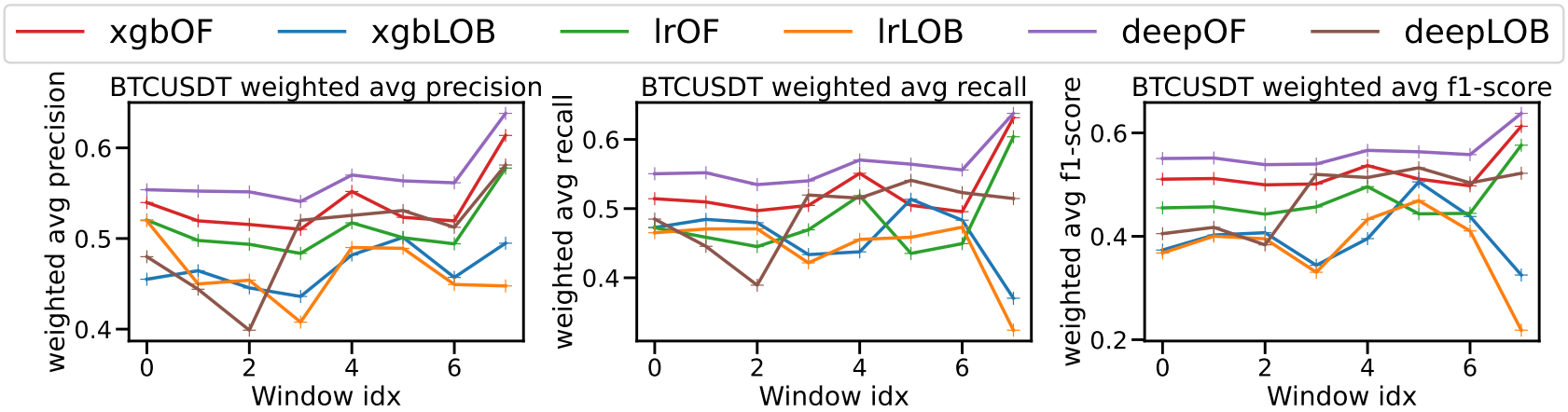

Figure 22: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 200$.

Imperial College London

ETHUSDT Window-Wise Macro Averages, $k = 200$

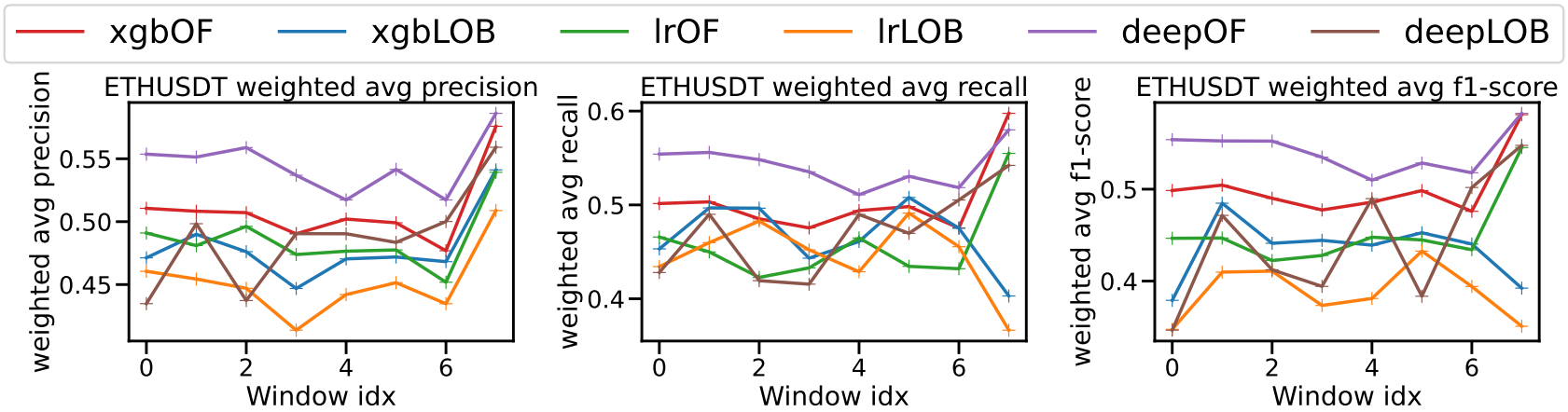

Figure 23: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 200$.

Imperial College London

MATICUSDT Window-Wise Macro Averages, $k = 200$

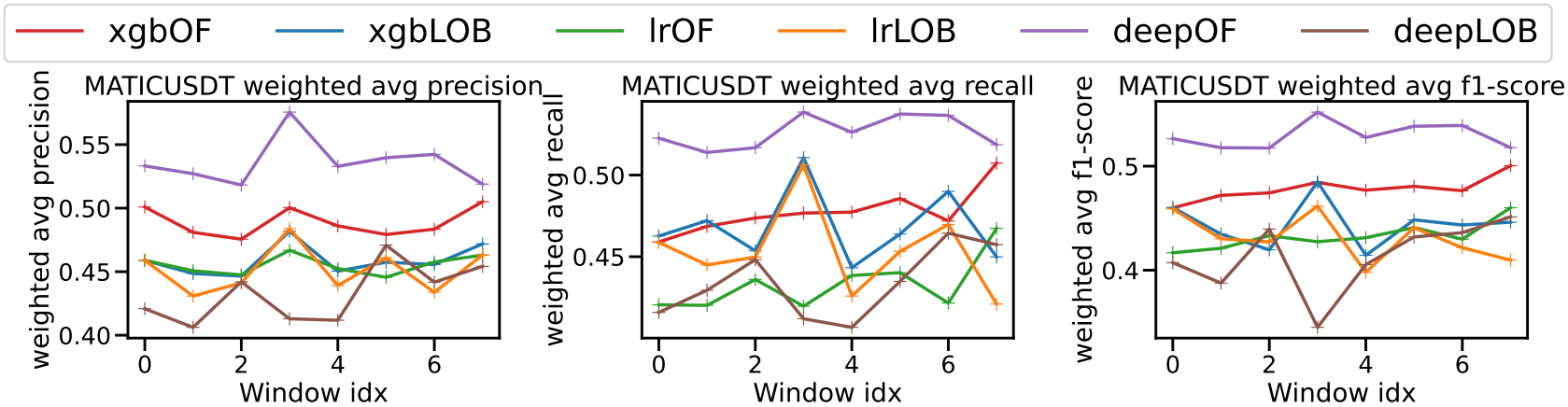

Figure 24: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 200$.

Imperial College London

SOLUSDT Window-Wise Macro Averages, $k = 200$

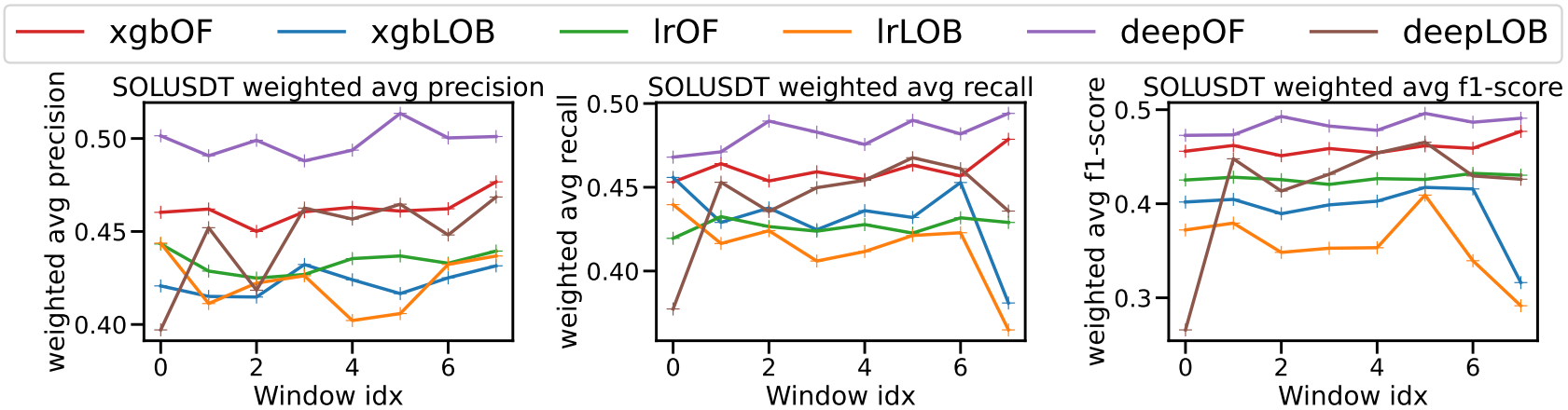

Figure 25: Comparing the macro averaged precision, recall and F1 score on the unseen test set for each window, for each trading pair, for $k = 200$.

Imperial College London

xgbOF Confusion Matrices, BTCUSDT

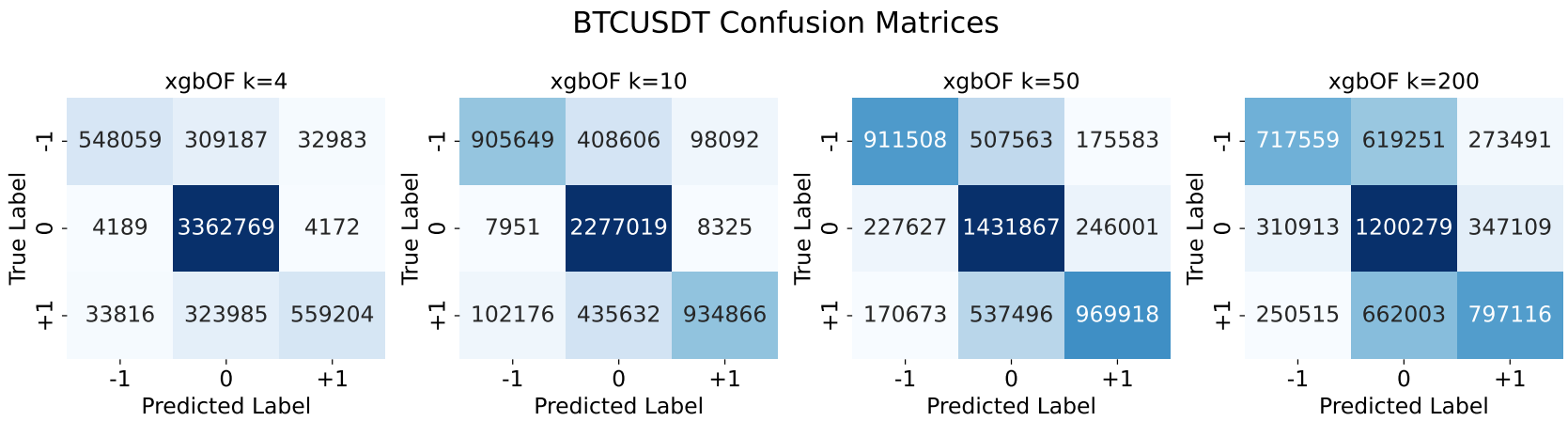

Figure 26: BTCUSDT xgbOF confusion matrices. Note that counts are summed across windows.

Imperial College London

xgbLOB Confusion Matrices, BTCUSDT

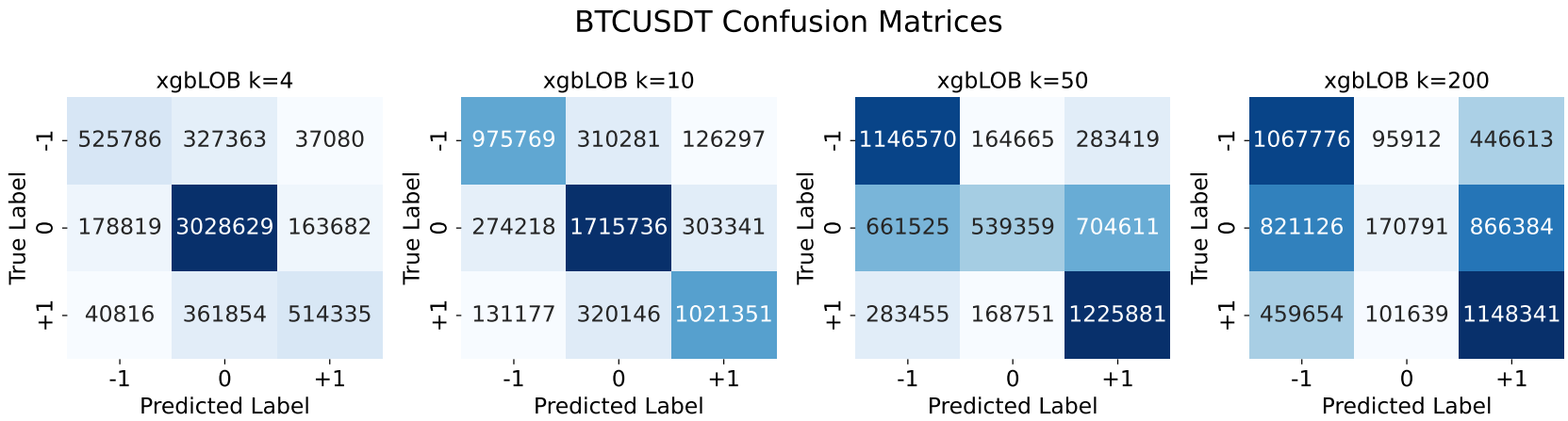

Figure 27: BTCUSDT xgbLOB confusion matrices. Note that counts are summed across windows.

Imperial College London

IrOF Confusion Matrices, BTCUSDT

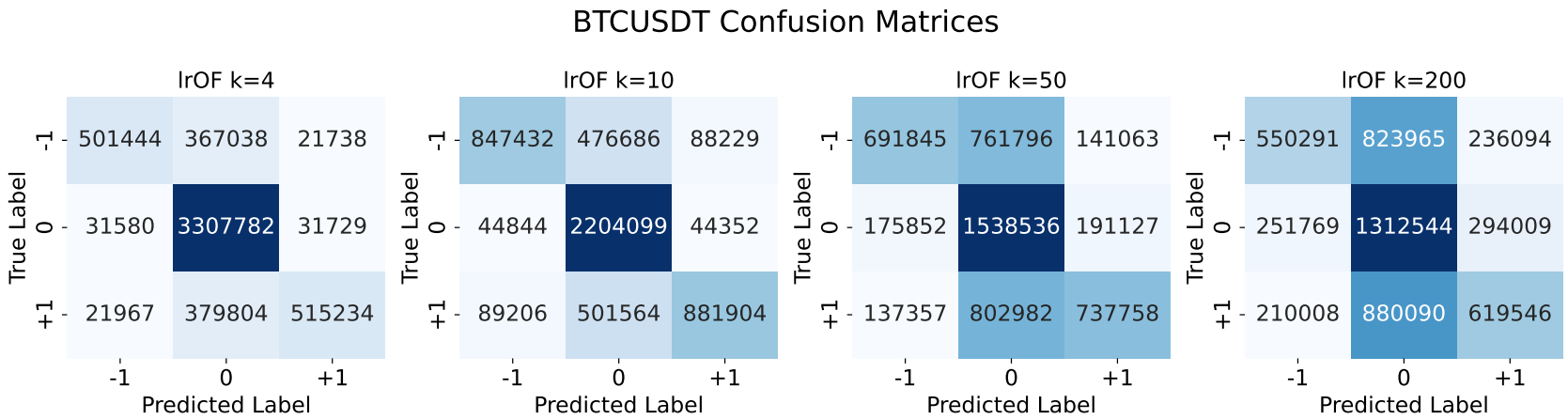

Figure 28: BTCUSDT IrOF confusion matrices. Note that counts are summed across windows.

Imperial College London

IrLOB Confusion Matrices, BTCUSDT

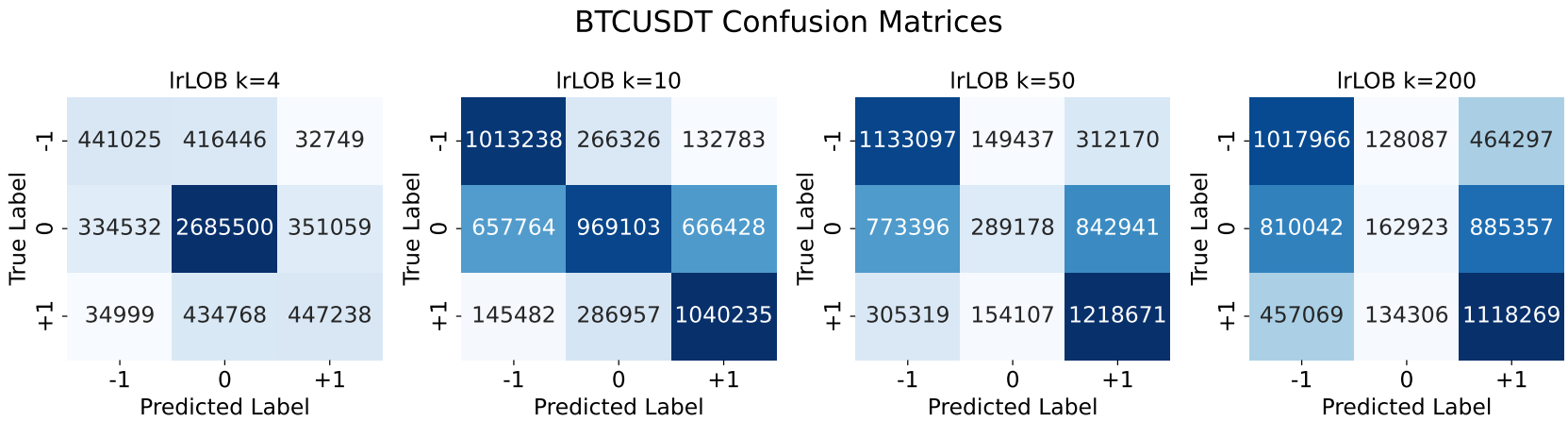

Figure 29: BTCUSDT IrLOB confusion matrices. Note that counts are summed across windows.

Imperial College London

deepOF Confusion Matrices, BTCUSDT

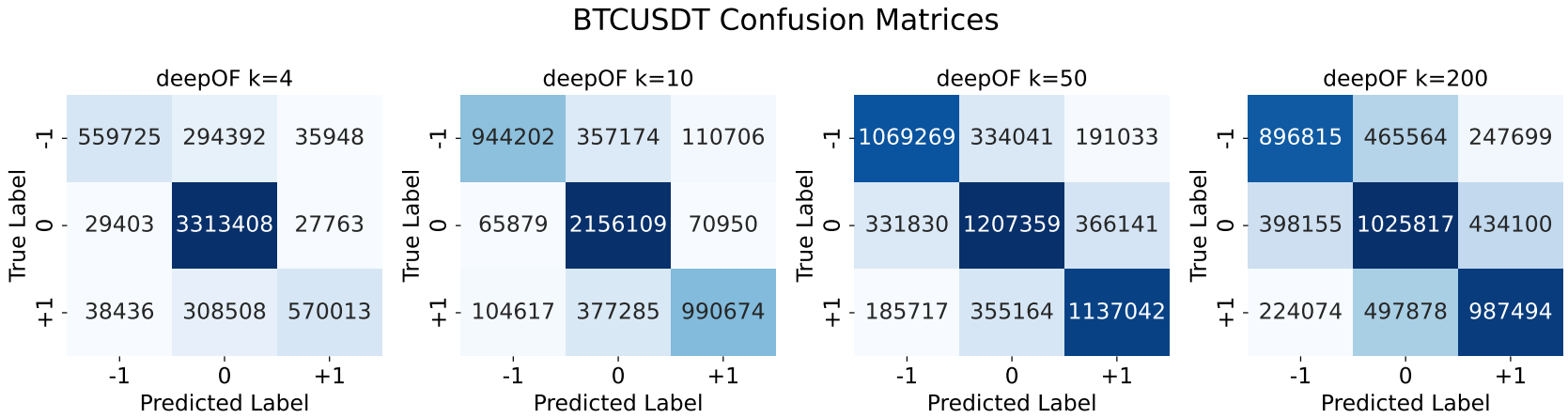

Figure 30: BTCUSDT deepOF confusion matrices. Note that counts are summed across windows.

Imperial College London

deepLOB Confusion Matrices, BTCUSDT

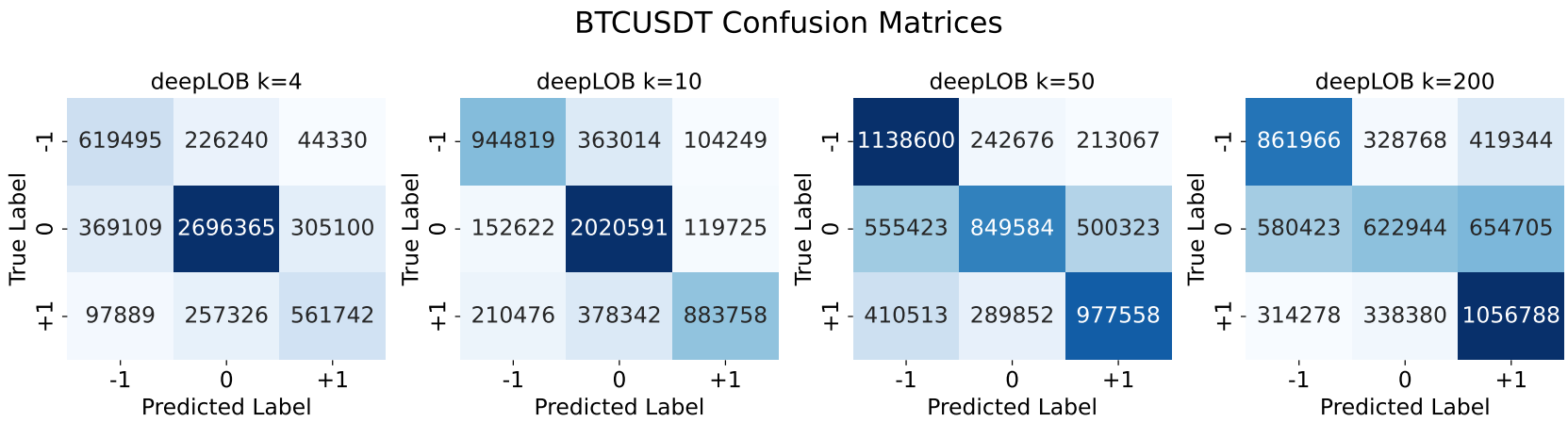

Figure 31: BTCUSDT deepLOB confusion matrices. Note that counts are summed across windows.

Imperial College London

Top 10 LOB Feature Importances

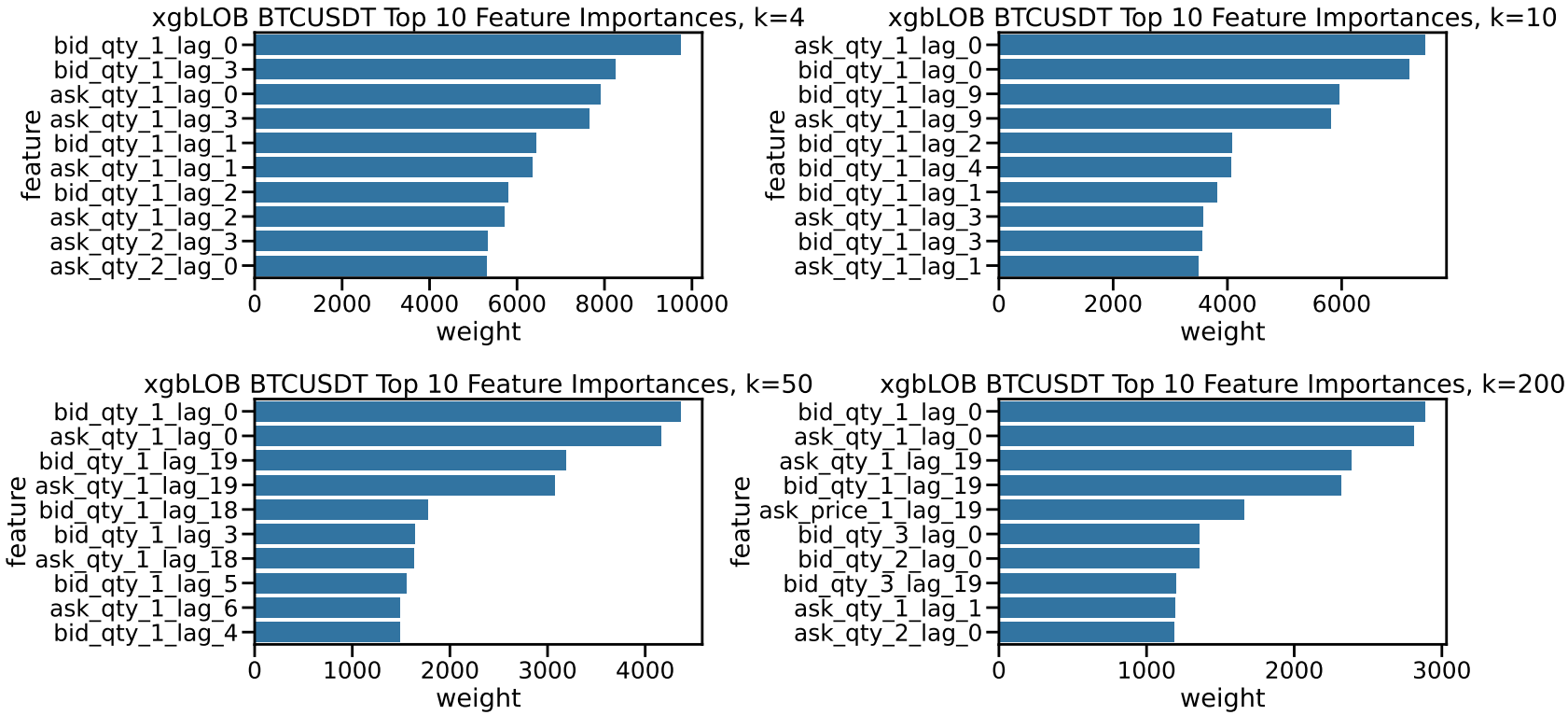

Figure 32: Top 10 XGBoost LOB Feature Importances for BTCUSDT, measured by weight = number of times a feature was split on.

Imperial College London

Top 10 OF Feature Importances

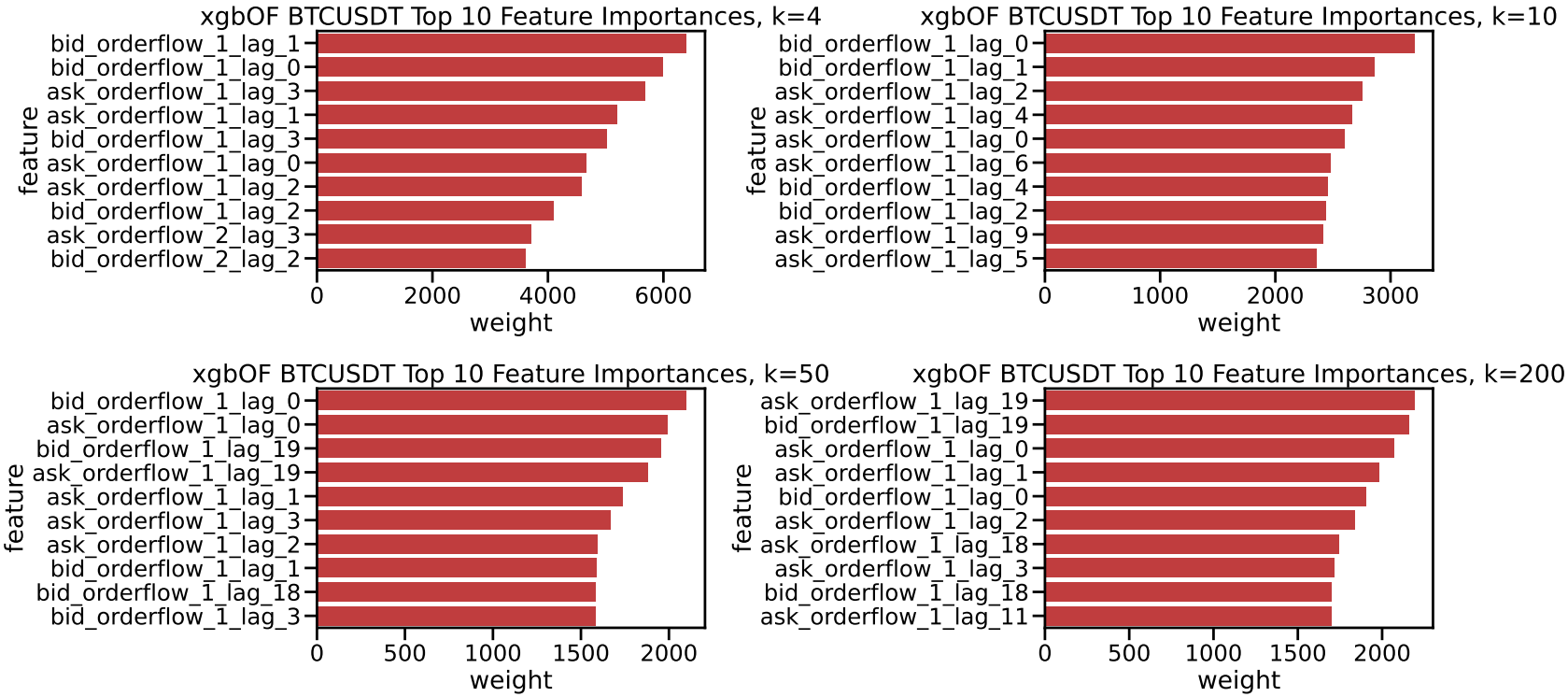

Figure 33: Top 10 XGBoost OF Feature Importances for BTCUSDT, measured by weight = number of times a feature was split on.

Feature Ordering Recap

- LOB features:

$$[\text{ask\_price\_}\ell\text{\_lag\_}p, \text{ask\_qty\_}\ell\text{\_lag\_}p,$$
$$\text{bid\_price\_}\ell\text{\_lag\_}p, \text{bid\_qty\_}\ell\text{\_lag\_}p]_{\ell=1,\ldots,10,p=0,\ldots,T}$$

- OF features:

$$[\text{ask\_orderflow\_}\ell\text{\_lag\_}p, \text{bid\_orderflow\_}\ell\text{\_lag\_}p]_{\ell=1,\ldots,10,p=0,\ldots,T}$$

- I.e. we first enumerate levels and then lags

Imperial College London

All non-zero LOB Feature Importances, $k = 4, 10$

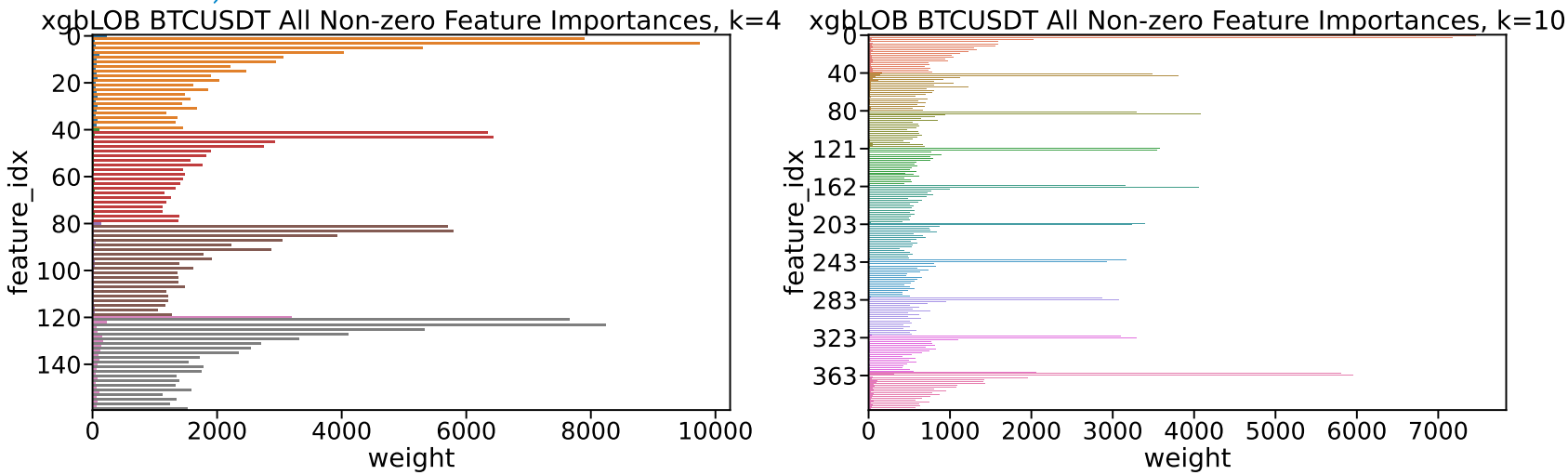

Figure 34: All non-zero LOB feature importances for BTCUSDT, measured by weight = number of times a feature was split on. Note that the colors represent unique (feature type, lag) pairs, where feature type is either quantity or price and lag $\in \{0, ..., T\}$, e.g. the first color represents price features with lag 0.

Imperial College London

All non-zero LOB Feature Importances, $k = 50, 200$

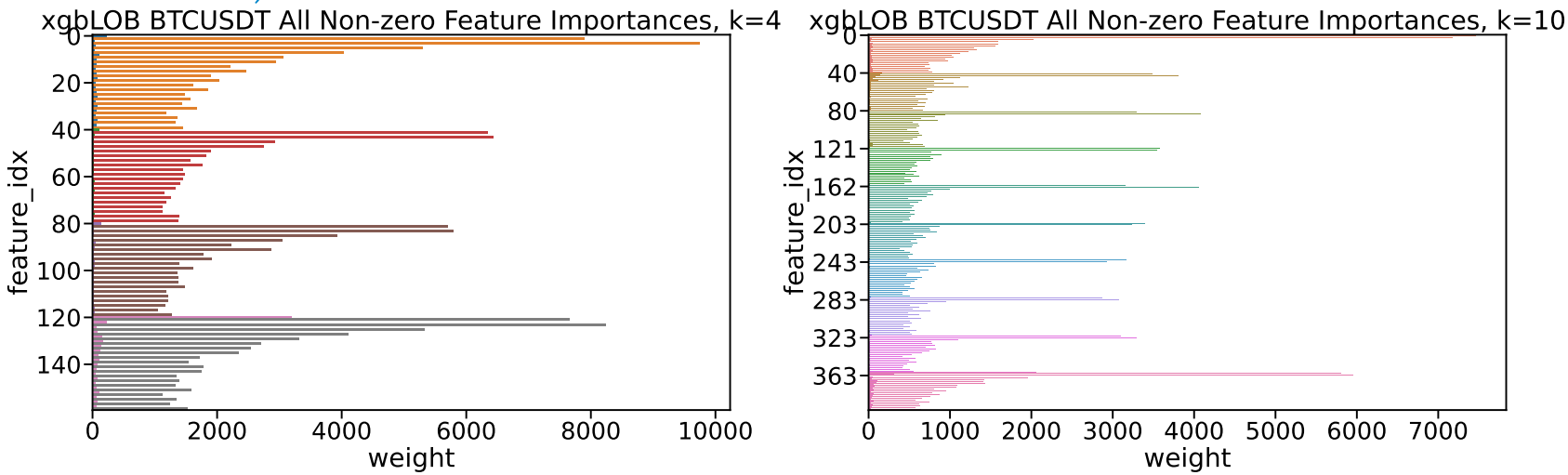

Figure 35: All non-zero LOB feature importances for BTCUSDT, measured by weight = number of times a feature was split on. Note that the colors represent unique (feature type, lag) pairs, where feature type is either quantity or price and lag $\in \{0, ..., T\}$, e.g. the first color represents price features with lag 0.

Imperial College London

All non-zero OF Feature Importances, $k = 4, 10$

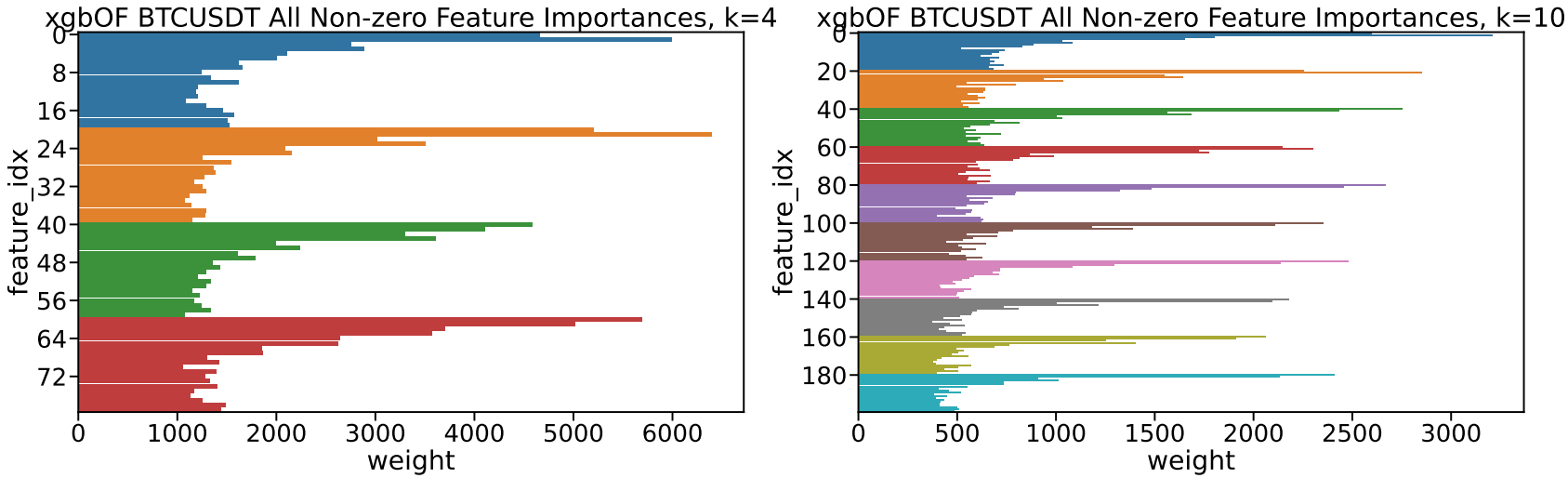

Figure 36: All non-zero OF feature importances for BTCUSDT, measured by weight = number of times a feature was split on. Note that the colors represent unique lags, where lag $\in \{0, ..., T\}$. E.g. the first color represents all features with lag 0.

Imperial College London

All non-zero OF Feature Importances, $k = 50, 200$

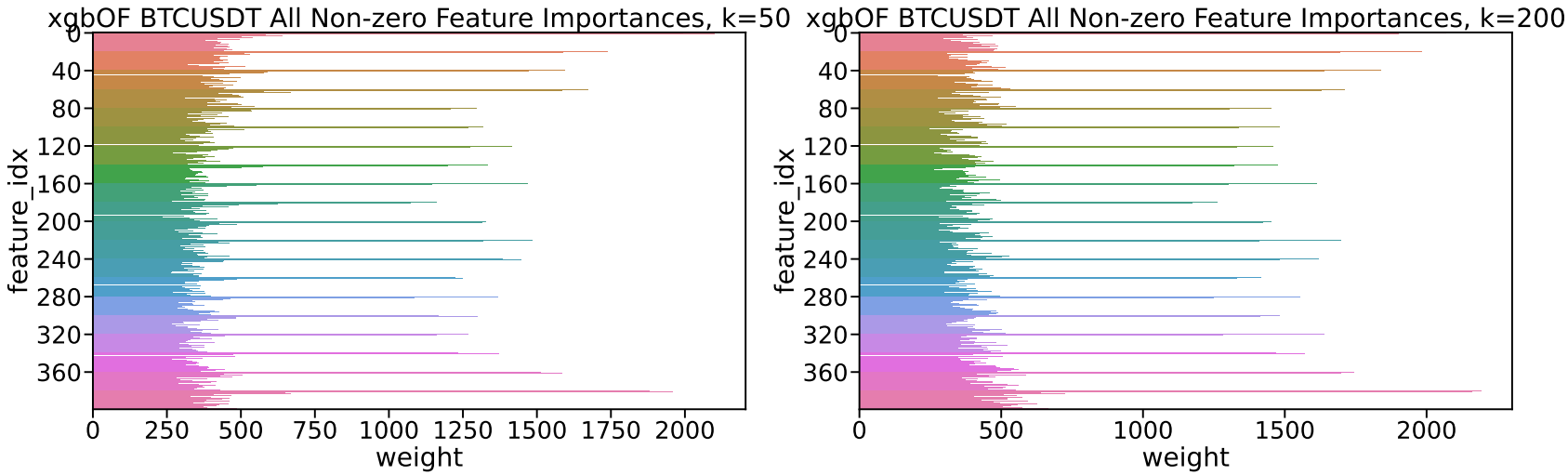

Figure 37: All non-zero OF feature importances for BTCUSDT, measured by weight = number of times a feature was split on. Note that the colors represent unique lags, where lag $\in \{0, ..., T\}$. E.g. the first color represents all features with lag 0.

Conclusion

Conclusions

- Precision of non-zero classes is perhaps most important for trading applications
- xgbOF is the winner for $k = 4$ and $k = 10$, with precision of $\approx 90\%$ for non-zero classes
- deepOF is the winner for $k = 50$ and $k = 200$ with non-zero class precision $\approx 60\%$
- Orderflow representation is far superior to raw orderbook representation

Future Work

- Compare models to those found in Zhang et al. [2019], Kolm et al. [2023], Lucchese et al. [2024] when applied to their data
- Only considered 4 trading pairs, Binance has over 200
- Live data lends itself to *online* modelling $\implies$ continuously test models

References I

Erdinc Akyildirim, Oguzhan Cepni, Shaen Corbet, and Gazi Salah Uddin. Forecasting mid-price movement of bitcoin futures using machine learning. *Annals of Operations Research*, 330(1): 553–584, 2023. doi: 10.1007/s10479-021-04205-x. URL https://doi.org/10.1007/s10479-021-04205-x.

E. Bacry and J. F Muzy. Hawkes model for price and trades high-frequency dynamics. *Quantitative Finance, 14*, 2013.

Charles Cao, Oliver Hansch, and Xiaoxin Wang. The information content of an open limit-order book. *Journal of Futures Markets*, 29(1):16–41, 2009. doi: https://doi.org/10.1002/fut.20334. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fut.20334.

References II

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794, 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

R. Cont, A. Kukanov, and S. Stoikov. The price impact of order book events. *Journal of Financial Econometrics*, 12(1):47–88, June 2013. ISSN 1479-8417. doi: 10.1093/jjfinec/nbt003. URL http://dx.doi.org/10.1093/jjfinec/nbt003.

References III

D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2): 215–232, 1958. doi: https://doi.org/10.1111/j.2517-6161.1958.tb00292.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1958.tb00292.x.

I. Fette and A. Melnikov. The websocket protocol. https://www.rfc-editor.org/rfc/rfc6455, December 2011. URL https://www.rfc-editor.org/rfc/rfc6455.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

References IV

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

References V

Michael I. Jordan. Chapter 25 - serial order: A parallel distributed processing approach. In John W. Donahoe and Vivian Packard Dorsel, editors, *Neural-Network Models of Cognition*, volume 121 of *Advances in Psychology*, pages 471–495. North-Holland, 1997. doi: https://doi.org/10.1016/S0166-4115(97)80111-2. URL https://www.sciencedirect.com/science/article/pii/S0166411597801112.

Eric Kauderer-Abrams. Quantifying translation-invariance in convolutional neural networks. *Pre-print*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations, San Diego, 2015*, 2017.

References VI

Petter N. Kolm, Jeremy Turiel, and Nicholas Westray. Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. *Mathematical Finance*, 33(4):1044–1081, 2023. doi: https://doi.org/10.1111/mafi.12413. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mafi.12413.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *Proceedings of iclr2014*, 2014.

References VII

Lorenzo Lucchese, Mikko S. Pakkanen, and Almut E.D. Veraart. The short-term predictability of returns in order book markets: A deep learning perspective. *International Journal of Forecasting*, 2024. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2024.02.001. URL https://www.sciencedirect.com/science/article/pii/S0169207024000062.

N. Gregory Mankiw. *Principles of Economics*. Cengage Learning, 7th edition, 2014. Chapter 4: The Market Forces of Supply and Demand.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Pre-print*, 2013.

References VIII

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *Pre-print*, 2017.

Damian Podareanu, Valeriu Codreanu, Sandra Aigner, Caspar Leeuwen, and Volker Weinberg. Best practice guide - deep learning. *Pre-print*, 02 2019. doi: 10.13140/RG.2.2.31564.05769.

Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951. URL https://api.semanticscholar.org/CorpusID:16945044.

References IX

David E Rumelhart., Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

MyungJae Shin, David Mohaisen, and Joongheon Kim. Bitcoin price forecasting via ensemble-based lstm deep learning networks. In *2021 International Conference on Information Networking (ICOIN)*, pages 603–608, 2021. doi: 10.1109/ICOIN50884.2021.9333853.

References X

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. *2017 IEEE 19th Conference on Business Informatics (CBI)*, 01:7–12, 2017. doi: 10.1109/CBI.2017.23.

Mahendran Venkatachalam. Recurrent neural networks – remembering what's important, gotensor, 2019. https://gotensor.com/2019/02/28/recurrent-neural-networks-remembering-whats-important/, [Accessed: 2024-06-10].

References XI

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, June 2019. ISSN 1941-0476. doi: 10.1109/tsp.2019.2907260. URL http://dx.doi.org/10.1109/TSP.2019.2907260.