**# 1.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Window Size** | **Step** | **MSE** |
| 0 ( baseline ) | 10 | 15 | 343.7744 |
| 1 | 10 | 5 | 20.2622 |
| 2 | 20 | 10 | 159.3617 |
| 3 | 30 | 15 | 483.2328 |

在本次實驗中，Scenario 1 (Window Size=10, Step=5) 得到最低的 MSE (20.2622)，顯示較短 Window Size 與較小 step 較能捕捉短期價格變化。反之 step 較大 (如 Scenario 0 與 3) 會導致訓練樣本減少、泛化能力變差，MSE 明顯上升。而較大的 window size 雖然能捕捉長期趨勢，但可能導致資料稀疏與學習困難。整體而言，小視窗配合密集採樣在此題目中較具預測效能。

**# 2.**

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Feature Combination** | **MSE** |
| 0 (baseline) | ['Open', 'High', 'Low', 'Close'] | 19.1997 |
| 1 | ['Open', 'High', 'Low', 'Close', 'Volume'] | 1096.2781 |
| **2** | **['Open', 'High', 'Low', 'Close', 'OpenInt']** | **13.9858** |
| 3 | ['Open', 'Low'] | 22.5414 |
| 4 | ['Open', 'High'] | 24.0502 |
| 5 | ['Open', 'High', 'Low'] | 22.6784 |
| 6 | ['Open', 'Close'] | 27.9799 |
| 7 | ['High', 'Low'] | 25.6523 |

上述實驗都在 Window Size=10, Step=5 的條件下進行。

(1) Baseline 模型 (使用 ['Open','High','Low','Close']) 的測試 MSE 為 19.1997。當將 Volume 納入特徵組合後，MSE 驟升至 1096.2781，模型表現嚴重惡化。此結果推測與兩點有關：其一為 Volume 數值尺度遠高於價格欄位，若未正規化易造成模型梯度不穩或收斂困難；其二為 Volume 在預測「隔日 High 價格」上未能提供實質訊息，反而引入噪聲干擾學習。

(2)最佳組合為 Scenario 2，將 OpenInt 加入原始四個價格特徵後，MSE 降至 13.9858。OpenInt 作為衍生性商品市場的未平倉合約量，可能揭示市場資金流動與預期波動性，間接輔助模型捕捉價格轉折訊號。

**# 3.**

|  |  |
| --- | --- |
| **有無正規化** | **Test MSE** |
| X | 20.8992 |
| V | 15.5296 |

\*本題選擇 **# 2**. 的 baseline model (Scenario 0) 進行正規化實驗，輸入之特徵為 ['Open', 'High', 'Low', 'Close']，並且在 Window Size=10, Step=5 的條件下進行。

本次實驗使用 MinMaxScaler 對特徵與標籤進行正規化，結果顯示，未正規化時 Test MSE 為 20.8992，正規化後下降至 15.5296，準確度明顯提升。正規化可以將不同量級的特徵縮放到相近範圍，避免特徵之間因尺度不同而影響模型學習效率。【Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). *How does batch normalization help optimization?* (No, it is not about internal covariate shift). In *Advances in Neural Information Processing Systems* (NeurIPS), 31.】研究指出，正規化 (尤其是 Batch Normalization 與資料層級的 normalization) 不僅提升訓練穩定性，更是促進模型泛化能力關鍵的一步。作者也透過實驗證明，即便在不調整學習率的情況下，光是加入正規化就能顯著減少 loss variance、加速收斂、提升測試準確度。

**# 4.**

視窗小於步幅的設計能減少樣本重疊從而提升訓練效率，但可能會導致捕捉不到時序關聯。然而，【Santoso, A., et al. (2021). *The Effect of Sliding Window Size and Step on Time Series Classification*. Proceedings of ICCSA.】研究指出步幅過大恐導致樣本不足、時序關聯遺失。

從 **#1.** 的實驗組合亦可見，當 window=10、step=5 時 MSE 僅 20.26，優於其他組合。因此，我認為視窗大於步幅的設計，能讓每筆資料重疊更多歷史資訊，提升樣本密度與時序連貫性，對於金融數據中微幅變化的捕捉尤其關鍵。此作法雖會增加訓練量，但有助於強化模型對趨勢轉折的辨識能力。因此視窗與步幅的設計仍應視資料性質與預測任務調整。

**# 5.**

在時間序列中，常見的資料增強技術包括 Time Warping、Jittering、Window Slicing 與 Mixup for Time Series。其中，Time Warping 可以透過非線性地調整時間軸上的資料點位置，模擬不同速率下的變化行為，進而提升模型對時序變化的泛化能力。

【Iwana, B. K., & Uchida, S. (2021, January). Time series data augmentation for neural networks by time warping with a discriminative teacher. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 3558-3565). IEEE.】提出了一種名為「Guided Warping」的資料增強方法，該方法結合了動態時間扭曲 (Dynamic Time Warping, DTW) 和形狀DTW (shapeDTW)，利用參考樣本的時間步長對齊特性，將樣本特徵扭曲以匹配參考樣本的時間步長，顯著提升模型準確率。

**# 6.**

在時間序列任務中，不同模型對 window size 大小有不同的處理策略。以下針對卷積神經網路 (CNN)、遞迴神經網路 (如 RNN、LSTM) 與Transformer 類模型進行說明。

1. 卷積類模型 (CNN)

* 視窗大小即定義卷積核一次能感知的時間範圍，透過固定感受野 (fixed receptive field，即 filter 大小) 於時間軸上滑動卷積
* 【Zhao, B., Lu, H., Chen, S., Liu, J., & Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of systems engineering and electronics*, *28*(1), 162-169.】實驗比較不同視窗長度對 CNN 分類效能的影響，發現中等視窗能在捕捉短期變化與長期趨勢間取得平衡，顯著提升模型準確率。

1. 遞迴類模型 (RNN、LSTM、GRU)

* 視窗大小即為輸入序列長度，模型能依序處理時間點訊號，具有彈性設置視窗長度的優勢
* 【Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S.,& Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, *33*, 17283-17297.】提出 LSTM 架構以改善 RNN 在長視窗下梯度消失問題，提升模型記憶長距訊號的能力。

1. Transformer 類模型

* 由於具備全局注意力， 利用全序列注意力機制處理輸入序列，理論上不受視窗長度限制，對長期依賴建模具優勢
* 實務上視窗長度過大將導致注意力計算複雜度為 O(n²)，【Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S.,& Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, *33*, 17283-17297.】提出 BigBird 架構以 sliding-window attention 有效控制視窗範圍，降低記憶體與計算成本。