威旭 x 清大期末*Kaggle*專題競賽
高頻交易策略設計與研究

110071025 計財大四 張智傑

19 May, 2025

特徵設計與篩選

主要是以「價格趨勢、壓力、波動性與成交動能」為核心邏輯，期望能從整段市場片段中萃取出反映市場結構與行為變化的統計資訊，並建立具備預測力的特徵集合

採用以下階段篩選策略：

- Feature Importance（Gain）+ SHAP 初步排序
- 共線性分析：熱力圖排除 |r| > 0.85 中的弱變數

- **價格與波動性衡量**

mid_price_mean：報價中間價平均，代表市場價格中心水平
spread_mean, spread_std：反映短期流動性與報價波動性
log_return_std, rolling_std_mid_price：價格穩定程度與近期波動結構

- **掛單深度與市場壓力**

bid_vol_total, ask_vol_total：觀察雙邊深度累積
depth_imbalance1：第一檔掛單不對稱程度，評估主導方向
order_pressure：觀察壓力波動性與是否反轉
liquidity_score：刻劃市場供需張力與潛在反轉訊號

- **成交動能與交易主導性**

buy_ratio：買方成交比率，辨識是否主動買入
vol_buy_minus_sell：買賣成交量差，量化多空力量強弱
fill_density, avg_fill_speed：代表市場活躍度與交易頻率
tick_imbalance：主動買入/賣出節奏比率來了解市場熱度與多空主導力量

- **價格偏移與結構扭曲**

vwap_bid, vwap_ask：買/賣方成交價格重心。
mid_vs_vwap_avg：報價與實際成交價格間的差距，揭示市場偏誤程度。

特徵設計與篩選

```python
# === 特徵萃取函數 ===
def extract_features(order_df, trade_df):
    mid_price = (order_df['BidPrice1'] + order_df['AskPrice1']) / 2
    spread = order_df['AskPrice1'] - order_df['BidPrice1']

    bid_vol_total = order_df[['BidLots1', 'BidLots2', 'BidLots3']].sum(axis=1)
    ask_vol_total = order_df[['AskLots1', 'AskLots2', 'AskLots3']].sum(axis=1)

    depth_imbalance1 = ((order_df['BidLots1'] - order_df['AskLots1']) /
                        (order_df['BidLots1'] + order_df['AskLots1'] + 1e-6)).mean()

    vwap_bid = ((order_df['BidPrice1'] * order_df['BidLots1'] +
                 order_df['BidPrice2'] * order_df['BidLots2'] +
                 order_df['BidPrice3'] * order_df['BidLots3']) /
                (order_df['BidLots1'] + order_df['BidLots2'] + order_df['BidLots3'] + 1e-6)).mean()

    vwap_ask = ((order_df['AskPrice1'] * order_df['AskLots1'] +
                 order_df['AskPrice2'] * order_df['AskLots2'] +
                 order_df['AskPrice3'] * order_df['AskLots3']) /
                (order_df['AskLots1'] + order_df['AskLots2'] + order_df['AskLots3'] + 1e-6)).mean()

    log_return = np.diff(np.log(mid_price + 1e-6))
    rolling_std = pd.Series(mid_price).rolling(window=5, min_periods=1).std()

    liquidity_score = ((order_df['BidLots1'] + order_df['AskLots1']) / (spread + 1e-6)).mean()
    order_pressure = ((order_df['BidLots1'] - order_df['AskLots1']) /
                      (order_df['BidLots1'] + order_df['AskLots1'] + 1e-6)).std()
```

特徵設計與篩選

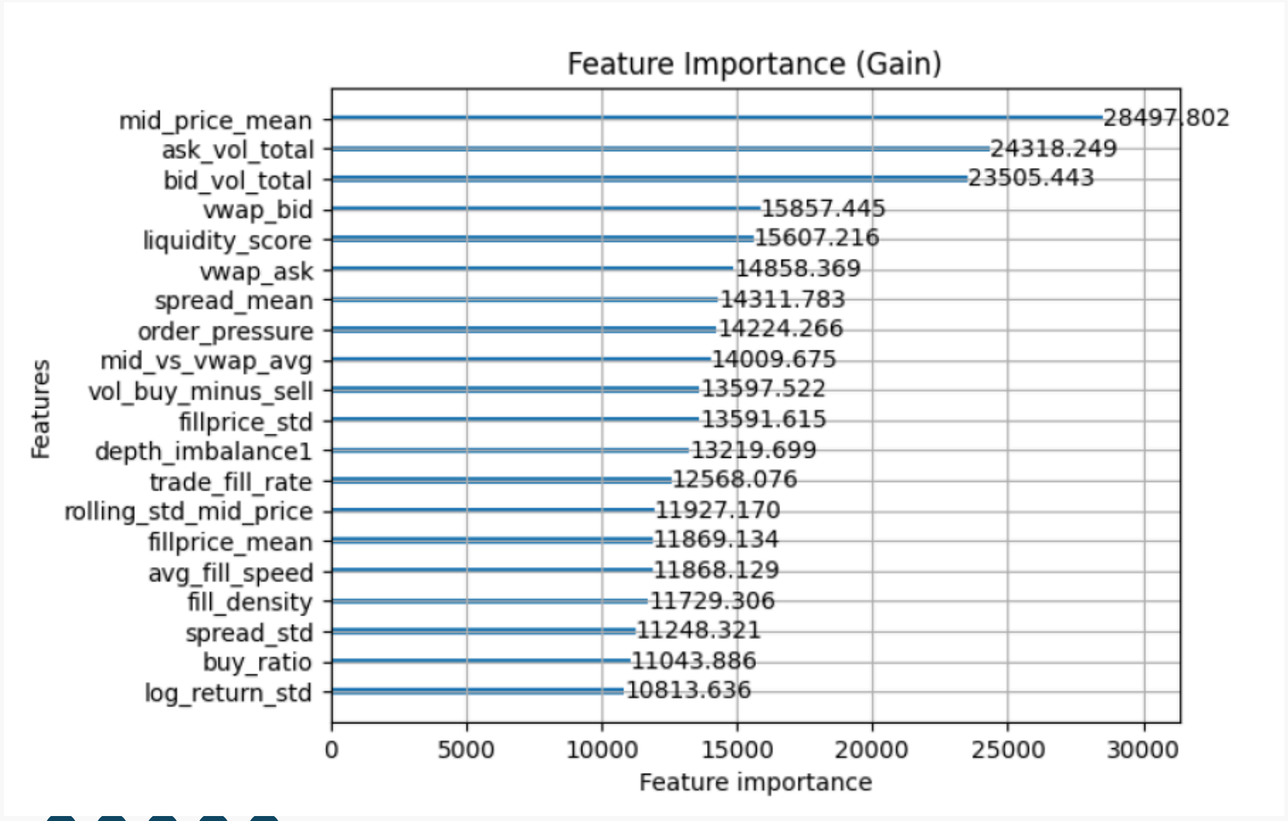
```python
# 成交特徵與進階統計
if len(trade_df) > 0:
    inout = trade_df['InOut'].to_numpy()
    fill_price = trade_df['FillPrice']
    fill_lots = trade_df['FillLots']
    buy = trade_df[trade_df['InOut'] == 1]
    sell = trade_df[trade_df['InOut'] == -1]

    buy_ratio = len(buy) / (len(trade_df) + 1e-6)
    vol_buy_minus_sell = buy['FillLots'].sum() - sell['FillLots'].sum()

    feature_dict.update({
        'fillprice_mean': fill_price.mean(),
        'fillprice_std': fill_price.std(),
        'buy_ratio': buy_ratio,
        'vol_buy_minus_sell': vol_buy_minus_sell,
        'inout_imbalance_std': np.std(inout),
        'fill_density': len(trade_df) / (len(order_df) + 1e-6),
        'avg_fill_speed': fill_lots.sum() / (len(order_df) + 1e-6),
        'tick_imbalance': len(buy) / (len(trade_df) + 1e-6),
        'trade_fill_rate': fill_lots.sum() / (bid_vol_total.sum() + ask_vol_total.sum() + 1e-6),
    })
```

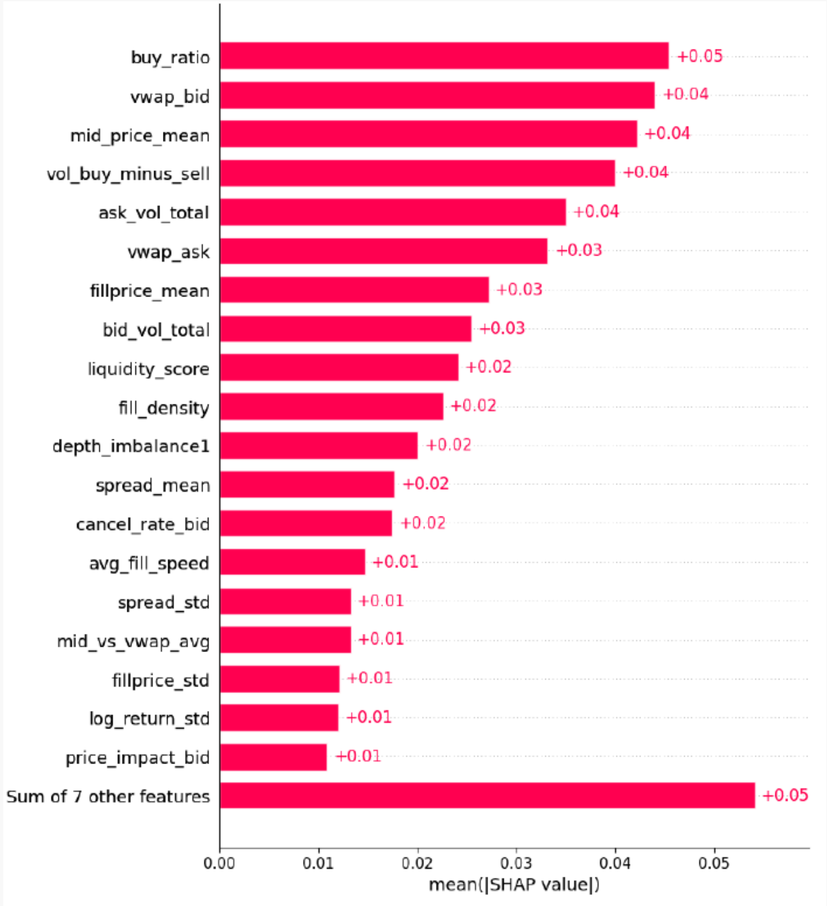
```python
else:
    feature_dict.update({
        'fillprice_mean': 0.0,
        'fillprice_std': 0.0,
        'buy_ratio': 0.0,
        'vol_buy_minus_sell': 0.0,
        'inout_imbalance_std': 0.0,
        'fill_density': 0.0,
        'avg_fill_speed': 0.0,
        'tick_imbalance': 0.0,
        'trade_fill_rate': 0.0,
    })

# 交互特徵
feature_dict.update({
    'buyratio_x_liquidity': buy_ratio * liquidity_score,
    'depthspread_x': depth_imbalance1 * spread_mean,
    'midprice_x_buyflow': mid_price_mean * vol_buy_minus_sell,
    'volatility_pressure': log_return_std * order_pressure,
    'vwap_bid_div_ask': vwap_bid / (vwap_ask + 1e-6),
})

return feature_dict
```

特徵設計與篩選— 特徵重要性分析

- 使用LightGBM內建 feature_importance 中的Gain-based 指標做排序，反映實際預測貢獻
- 計算SHAP (SHapley Additive exPlannations) 值，了解模型中每個特徵對於輸出的影響程度

藉由上述兩者，可以大致過濾出重要度低的特徵





特徵設計與篩選 — 共線性分析

- 計算特徵間的相關係數，當兩者特徵相關係數達到絕對值超過0.85，代表高度共線，會使得模型解釋偏誤

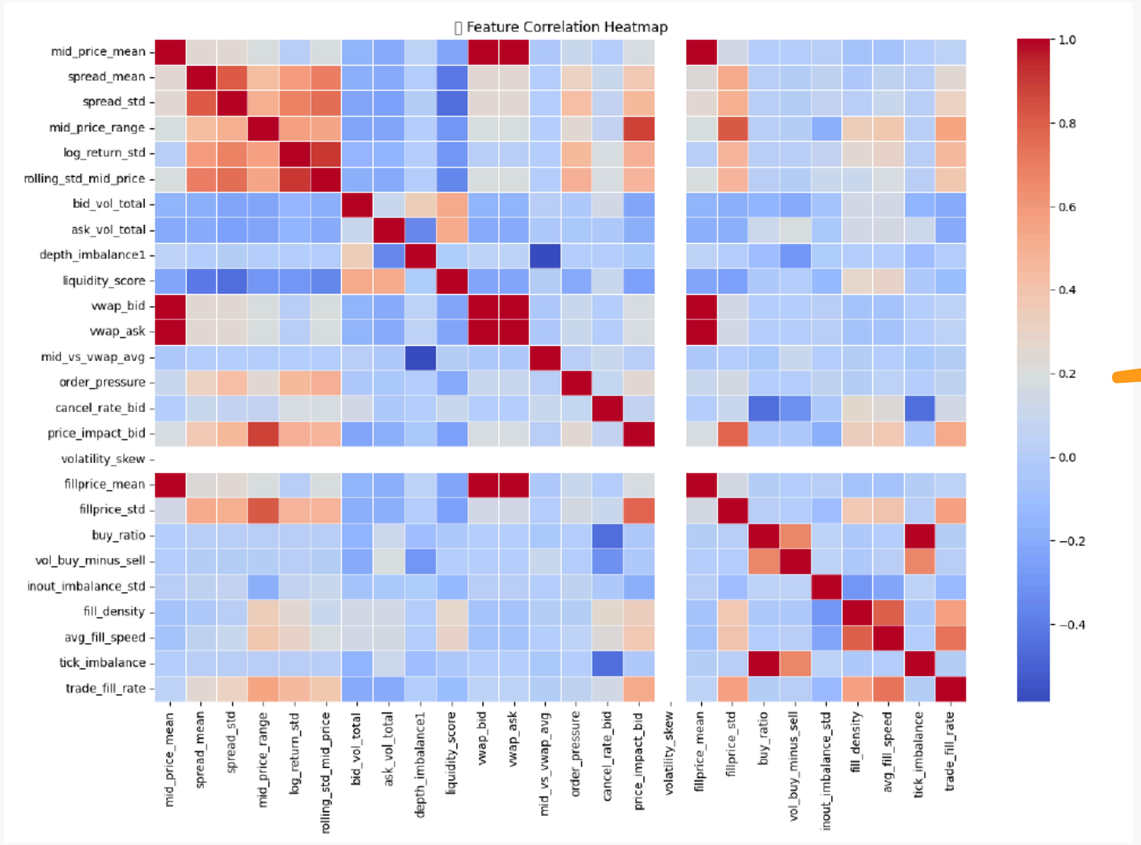
策略：保留具有解釋力與穩定 SHAP 貢獻的特徵，並對其中共線性高的組合轉為交互特徵，達到保訊降噪的目的

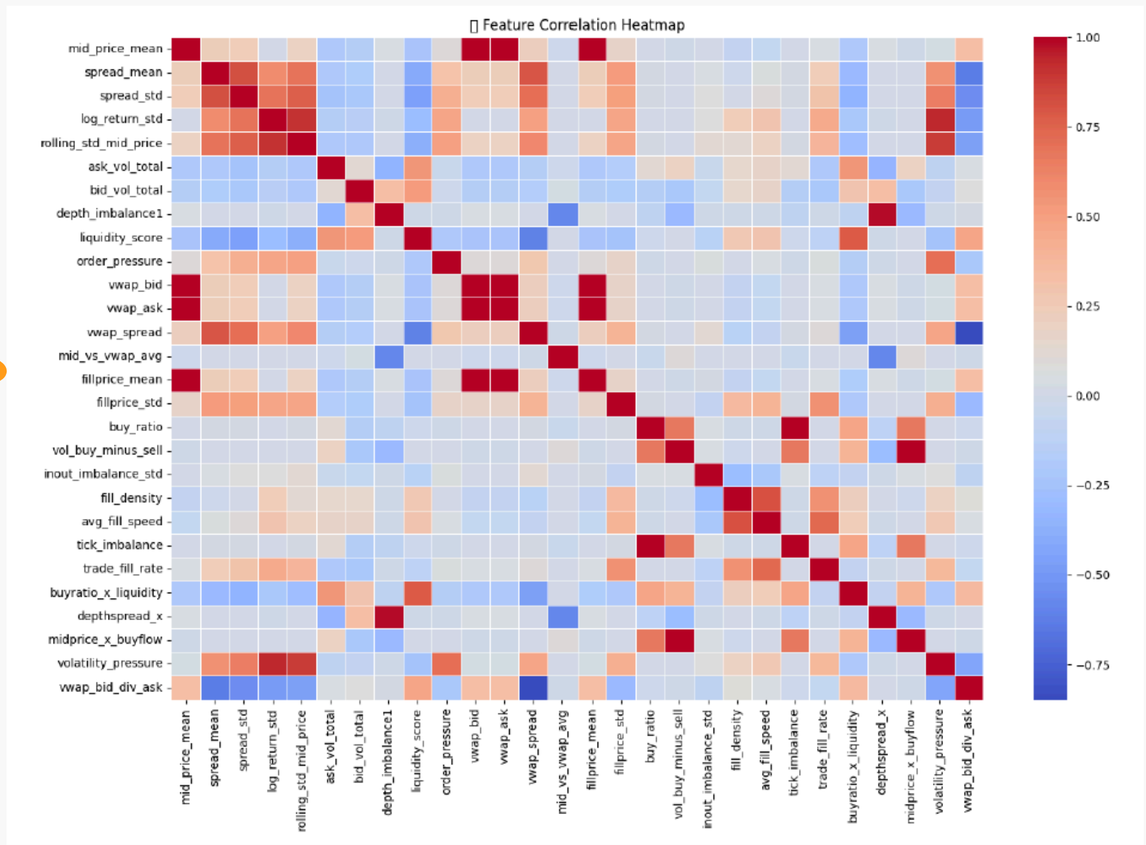
buyratio_x_liquidity：主動單強度 x 市場深度，反映急漲推動力

midprice_x_buyflow：中位價格 x 買賣量淨值，衡量趨勢導向買盤

volatility_pressure：波動程度 x 掛單壓力，偵測潛在價格突破

depthspread_x與vwap_bid_div_ask：衡量市場結構是否失衡





模型選擇與想法 — *LightGBM*

選擇使用 LightGBM 作為核心演算法，因為在處理高頻交易資料與多維數值特徵，展望其模型的穩定性、訓練效率與解釋性，其主要邏輯為透過多顆決策樹的「殘差疊加」來預測目標值，每顆樹都在上一步預測失敗的部分做修正，要去選擇最能降低Loss特徵+效果好的分裂點。

LightGBM 對於多重共線性本身並不敏感，亦能自動跳過冗餘特徵，但我仍進行了共線性分析與 SHAP 檢測，原因在於 —— 有些高度相關的特徵雖然表面冗餘，實則代表不同市場結構訊號，我會以交互方式保留；而那些在模型中實證貢獻極低的特徵，則透過 SHAP 與 ablation test 剃除，避免模型陷入雜訊與過擬合。

模型選擇與想法 — *LightGBM*

- 採用 GroupKFold 作為交叉驗證基礎，避免 group 間資訊洩漏（設定做5次)
- 加入 early stopping，限制 boosting 輪數，防止過度擬合
- 調整關鍵參數（如 num_leaves, feature_fraction, bagging_fraction）以提升泛化能力。

最終模型保有良好的擬合力，同時在不同 fold 間表現穩定，亦具備合理的預測解釋性，為後續分析提供可信的基礎

其實自己也有考慮使用其他模型，或是LightGBM與其他模型做結合，不過似乎效果不顯著

問題點 - *Overfitting*

在特徵與模型參數優化上，自己主要判斷好壞模型的依據是使用train data拆分成訓練與測試集，透過測試集預測去觀察MSE變化。起初自己過於著重在優化train data的MSE上，所以當實際丟上kaggle去測試test data時，發現MSE績效差距甚大，自己也認知到：

模型不應只是在已知資料上表現優異，
也需要在未知樣本中穩定發揮預測能力，
提升泛化力與解釋性

在Kaggle上的預測MSE也從1.25665逐漸下降到1.22974左右