Assignment 1: Building and backtesting an algorithmic trading strategy

Data download

- I used Python version: 3.12.7
- Download already cleaned and merged trades and quotes data using the following link:
- Company: Citigroup, ticker "C"
  - Time period: Jan 1, 2024 - Jan 31, 2024
  - Step_3_C_Trades_Quote_Joined.parquet
- Import Parquet file into Pandas dataframe
- If you like, you can use algotrading2.yml file (in the shared folder) to configure a Python environment

Defining variables

- Data consists of all trades (PRICE, SIZE) with timestamps merged to the prevailing quotes (BID, ASK) on the same exchange
- Calculate following variables for each minute:
  - average relative quoted spread
  - order imbalance
  - average depth imbalance as (ASKSIZ-BIDSIZ)/(ASKSIZ+BIDSIZ)
  - traded volume
  - one-minute return from closing mid-prices for each minute
  - realized volatility as sum of squared 1-minute returns over the past hour
- Winsorize all variables at 1% and 99%

Descriptive Statistics

- Calculate summary statistics for target (Ret_{t+1}) and features (relative spread, volume, volatility, order imbalance, depth imbalance):
  - mean, standard deviation, minimum, 25% 50%, 75%, maximum
- Compute correlation matrix for features
  - **think:** can you include both OIB and $DepthImbalance$ in the same regression?
- Plot histogram distributions for features

Linear regression

- Estimate following OLS regression:

$$Ret_{t+1} = \alpha + \beta_1 OIB + \beta_2 RelSpr_t + \beta_3 Vol_t + \beta_4 Volat_t + \varepsilon_{t+1}$$

- First, run in-sample estimations
  - using Statsmodels package
  - using SKLearn package
    - scale (i.e. standardize) features before estimation

Performance evaluation metrics

- Calculate and interpret following metrics:
  - correlation between predicted and actual 1-minute returns
  - R^2
  - RMSE
  - MAE

Out-of-Sample testing

- Use SKLearn package
- Split into first 80% for **training dataset** and use last 20% for **test dataset**
  - **remember:** preserving temporal order is important when predicting financial returns!
  - **fit** scaler and **linear model** on training data only
  - predict on **unseen test data**
- Recalculate R^2, RMSE, MAE, correlation
- Sort features on (absolute) regression coefficients ("feature importance")

Walk-forward testing

- Use SKLearn package to estimate 2-week rolling OLS (refitted daily):
  - Train a model on a 2-week window
  - Predict one-minute returns for next day
  - Slide the window one day forward and repeat
- Store
  - coefficients for each feature from each rolling regression
  - predicted and actual returns for each minute
- Recalculate R^2, RMSE, MAE, correlation and compare to corresponding out-of-sample metrics
- Compare "feature importance" to out-of-sample estimations

Backtesting rolling window trading strategy

- Let $s_t = 1$ if predicted return is positive ($\widehat{Ret_t} > 0$) and in the top quartile ($\geq 75\%$) of observed returns,
- $s_t = -1$ if $\widehat{Ret_t} < 0$ and in the bottom quartile ($\leq 25\%$) of observed returns
- $s_t = 0$ otherwise.
- First, ignore transaction costs
  - compute total cumulative return and plot it on the graph
  - average daily return, std of daily returns, daily Sharpe ratio
  - hit rate, max drawdown, turnover
  - t-stats of strategy daily returns
- **Question: would you trade based on your strategy? why or why not?**

Backtesting rolling window trading strategy

- Now add estimated transaction costs
  - Recompute all metrics above
  - Compare the strategy performance with vs without transaction costs
- **Question: would you trade based on your strategy? why or why not?**