Deep Learning Coursework

Ilias Ramim, Ulysse Ristorcelli

November 1, 2025

1 Introduction

Our goal is to predict the direction of midprice evolution, with respect to some features. To do so, we will:

- Visualize and explore the Data at hand.
- Compute additional meaningful features to feed the network.
- Train and finetune a Feedforward Neural Network.

2 Data Exploration

To make our prediction, we have: four order book levels, including price and volume (bid/ask), the five previous midprice evolution directions, and the actual midprice evolution direction (the target). You can find some analysis of the available data below.

2.1 Order Book Imbalance (OBI)

2.1.1 Level 1 OBI

The first meaningful feature we can compute is the order book imbalance. Indeed, a large bid volume compared to a small ask volume typically indicates strong upward pressure on the price. Let us take a look to the midprice direction ditribution w.r.t. to the level 1 OBI:

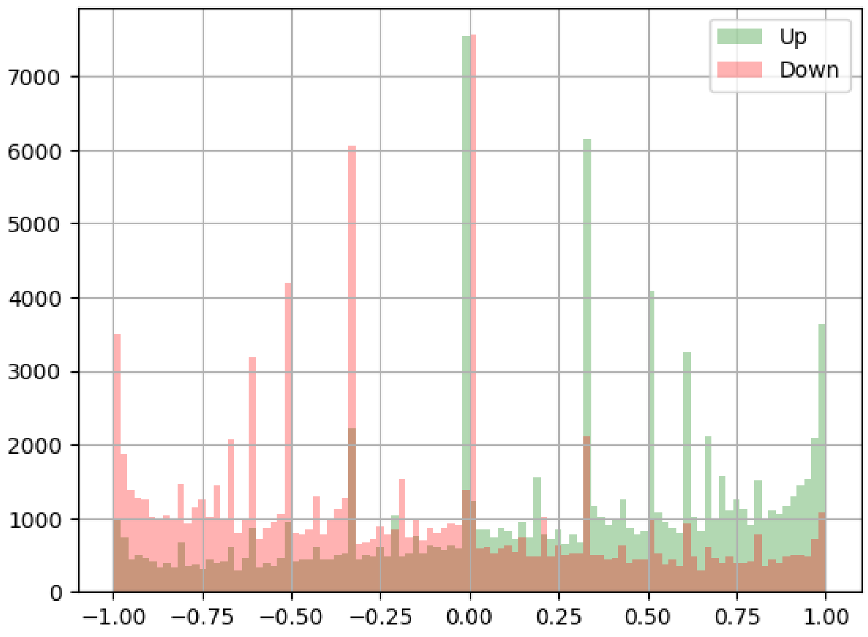


Figure 1: Order Book Imbalance of the first level.

We can extract some information:

- On average, the midprice direction follows the OBI pressure.
- The spike around 0 with switched colors is explainable (if the OBI is small, bid and ask volume are almost equivalent, there is distinct pressure).
- We observe distinct spikes at very specific values that are symmetric.

The aforementioned spikes seem to come from the distribution of the volumes: There is a much larger proportion of multiples of 100 than others numbers; this results in some values being overrepresented in the first order OBI histogram:

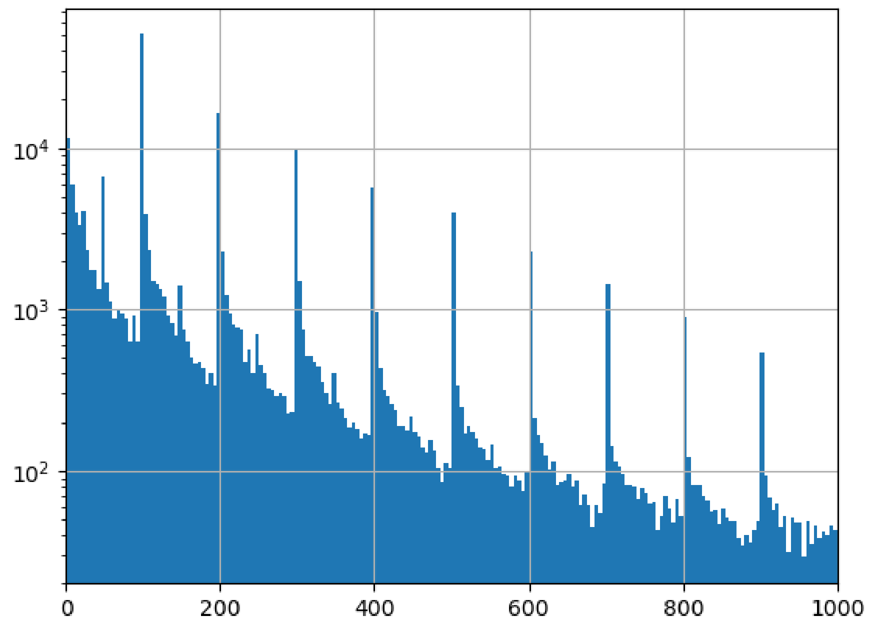


Figure 2: Bid volume for the 1st order book level.

2.1.2 Three first levels OBI

When we aggregate more levels to compute the OBI, we find a bimodal gaussian distribution:

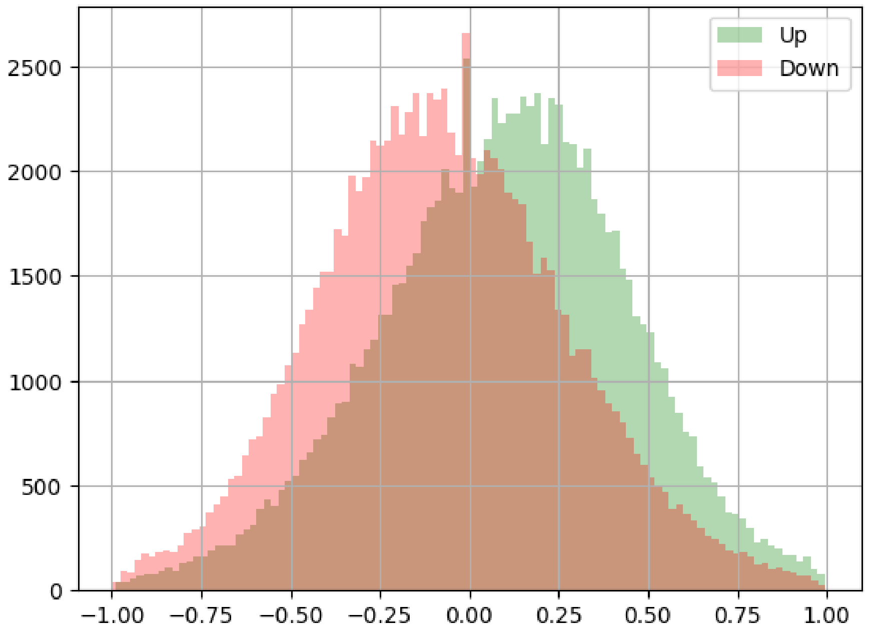


Figure 3: Order Book Imbalance of the first three levels.

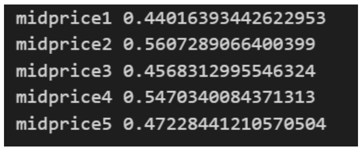
This finding makes sense, as we expect that the pressure on either way induces a midprice change towards the pressured side, more or less some noise. As we aggregate several levels, the distribution tends to smooth, and the spikes aforementioned tend to vanish. We end up with such a probability distribution of the midprice evolution direction conditioned by the OBI. We can conclude that The OBI, computed wether on the first level or on successive levels, may be a good feature to feed the FNN with.

2.2 Previous midprice evolutions

Another meaningful information is the previous midprice direction changes. To see the relation between these features and the actual target, we computed the ratio of how much midprices went up considering their j-th previous midprice evolution lag_j went up:

$$\frac{\sum_{i=0}^{N_j} \mathbf{1}_{\{x_i=1, lag_j=1\}}}{N_j}$$

N_j being the number of samples for which the previous midprice j (denoted lag_j) equals 1 (midprice went up); x_i being the i-th target for which the lag_j equals 1. We observe a mean reversal pattern:



```
midprice1 0.44016393442622953
midprice2 0.5607289066400399
midprice3 0.4568312995546324
midprice4 0.5470340084371313
midprice5 0.47228441210570504
```

Figure 4: actual midprice evolution ratio considered a previous evolutions

The value oscillates around 0.5: this may show that on average, after an evolution, the next midprice change tends to counterbalance the previous change in the other way.

3 Feature Engineering

200 000 examples may seem a few number of samples for a small FNN (<10 layers) to learn patterns from. Thus, we added the following features we thought may be meaningful to find patterns:

3.1 Order Book Imbalances (OBIs)

We added as features the four "aggregated" order imbalance, meaning for$i \in \{1, 2, 3, 4\}$:

$$OBI_i = \frac{\sum_{j=1}^{i} V_i^{bid} - V_i^{ask}}{\sum_{j=1}^{i} V_i^{bid} + V_i^{ask}}$$

where V_i^{bid} is the bid volume for the i-th level, V_i^{ask} is the ask volume for the i-th level.

3.2 Spread

We added as a feature the bid-ask spread for level 1:

$$Spread = P_1^{ask} - P_1^{bid}$$

3.3 Microprice

As an extension of the simple midprice, we also added as a feature the Microprice defined as follows:

$$\text{Microprice} = \frac{P_{\text{ask},1}\, V_{\text{bid},1} + P_{\text{bid},1}\, V_{\text{ask},1}}{V_{\text{bid},1} + V_{\text{ask},1}}$$

The idea behind this formula is that the microprice moves the midprice toward the side with the highest trading pressure by weighting each quote with the volume of the opposite side, reflecting the expected next move of the midprice given the current imbalance in the order-book.

3.4 Momentum

We wanted to have a feature that embodied the current trend using the last 5 changes in price. We called it momentum and computed three different kind of momementum.

The first formula gives a directional tendency by keeping signs.

$$Momentum = \frac{1}{5} \sum_{i=18}^{22} (2x_i - 1)$$

The second formal yields large values for strong directional trends ($\pm 5 \rightarrow 25$), small for choppy movement, allowing us to emphasize persistence. Although we lose the sign (direction) the first formula keeps it, so in the end we don't lose information.

$$MomentumSq = \left(\sum_{i=18}^{22} (2x_i - 1) \right)^2$$

The last formula creates an interaction feature that captures whether the current static pressure from the order book (OBI) either confirms or contradicts the recent dynamic trend (the average momentum of the last five price moves).

$$MomemtumOBI = \text{Imbalance} \times \frac{1}{5} \sum_{i=18}^{22} (2x_i - 1)$$

3.5 Volume Depth

The total volume depth acts as a measure of the resistance or support of the market. For example, a deep book with high volume on the sell side implies significant resistance that could prevent the price from rising. This is why we thought it could be a good feature to add.

$$V_{\text{depth, ask}} = \sum_{i=1}^{4} V_{\text{ask},i}$$

$$V_{\text{depth, bid}} = \sum_{i=1}^{4} V_{\text{bid},i}$$

3.6 Volume Gap

The idea behind this feature is that inter-level price gaps reveal the book's 'thickness'. Indeed, large gaps suggest a thin, illiquid market where consuming one level of orders will cause a significant price jump, signaling the potential for higher volatility.

$$\text{Gap}_{\text{ask},i} = P_{\text{ask},i+1} - P_{\text{ask},i} \quad \text{pour } i \in \{1, 2, 3\}$$

$$\text{Gap}_{\text{bid},i} = P_{\text{bid},i} - P_{\text{bid},i+1} \quad \text{pour } i \in \{1, 2, 3\}$$

4 Model training

To predict the target, we are going to train a Feedforward Neural Network (FNN), with the following steps:

- Scale the data
- Split the data into train/test sets
- Optimize hyper-parameters using K-fold Cross Validation
- Train the final model on the train set
- Evaluate the model on the test set

You may see the code details in the Appendix, we only explain our theoretical approach in the following paragraphs.

4.1 Scaling the data

Using scikit-learn, we compute:

$$X_{scaled} = \frac{X - \overline{X}}{\sigma_X}$$

4.2 Splitting the data

We used **90% of the data for the train split, 10% for the test split**. The train split is used to find the best hyper-parameters, the test split is used to evaluate the final model. Such a split prevent any **data leakage** from happening.

4.3 Optimizing hyper-parameters

To find the best hyper-parameters of the model (number of layers, number of neurons per layer, number of epochs, best optimizer, etc.) we used **Optuna**, a python package that allows us to repeat N number of trainings. In each training iteration:

- A new set of parameters is chosen among the boundaries we set
- A model is trained using K-fold cross validation, allowing the final model to be more robust
- The model is evaluated on the last fold, from which we derive the accuracy
- We compute the average accuracy over all the folds, and store that accuracy to compare to the other iterations of Optuna.

4.4 Training and Evaluating the Final Model

Finally, we trained the model with the best parameters on the whole train split, and test its accuracy on the test set.

5 Results

We made two hyper-parameters optimization, and two final models: one without any other features than the original data, one with the features mentioned in the **Feature Engineering** part.

5.1 Accuracies

Model 1 (Without additional features)	Model 2 (With additional features)
0.7131	0.7238

Figure 5: accuracies of the model depending on the selected features

5.2 Features Importance (Raw Data)

We have seen that adding features improved the accuracy by nearly 1%. Let us see which features are more meaningful for the model. To do so, we iteratively selected one feature and used scikit-learn to shuffle the data of that specific feature, and observe how the accuracy decreased.

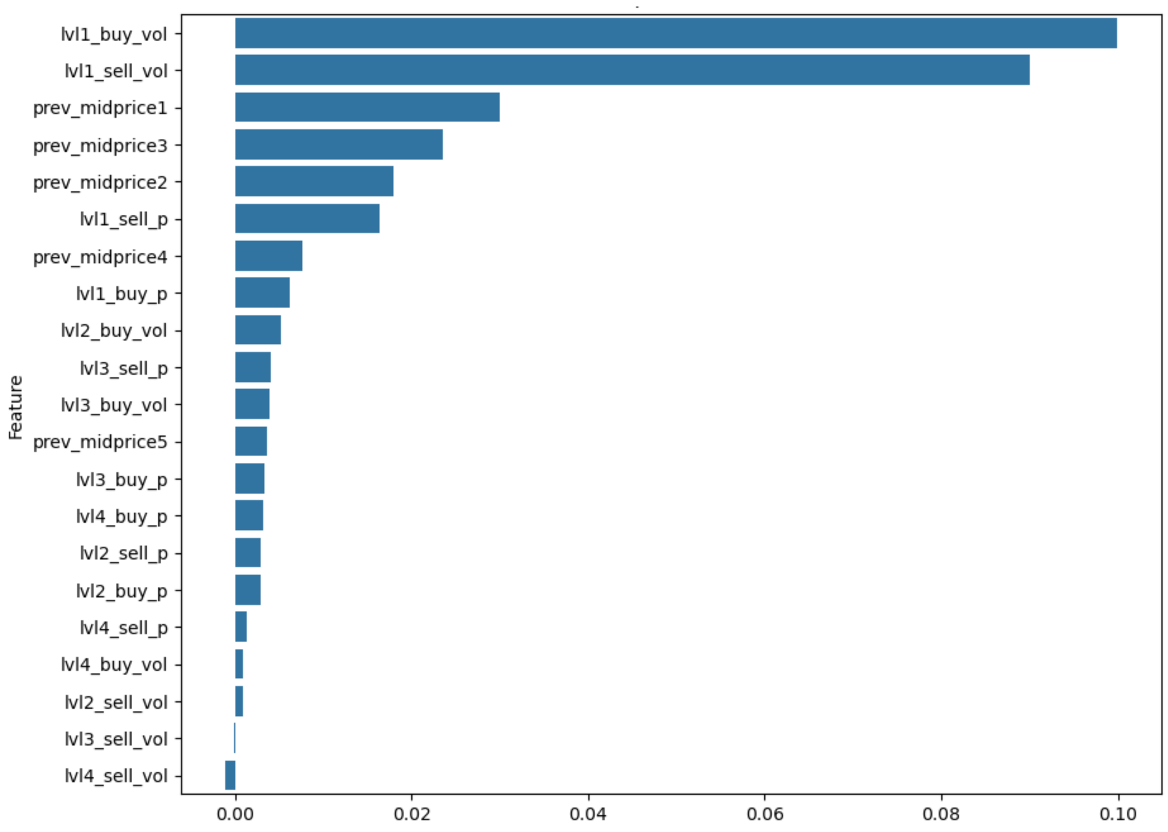


Figure 6: Feature Importance (accuracy decrease when the feature data is shuffled)

The bid/Ask volumes of the 1st level are the most important features (which suggests adding the OBI may have been a good choice), followed by the previous midprice change directions which makes sense, and the level 1 bis/ask prices.

5.3 Features Importance (Features Engineered)

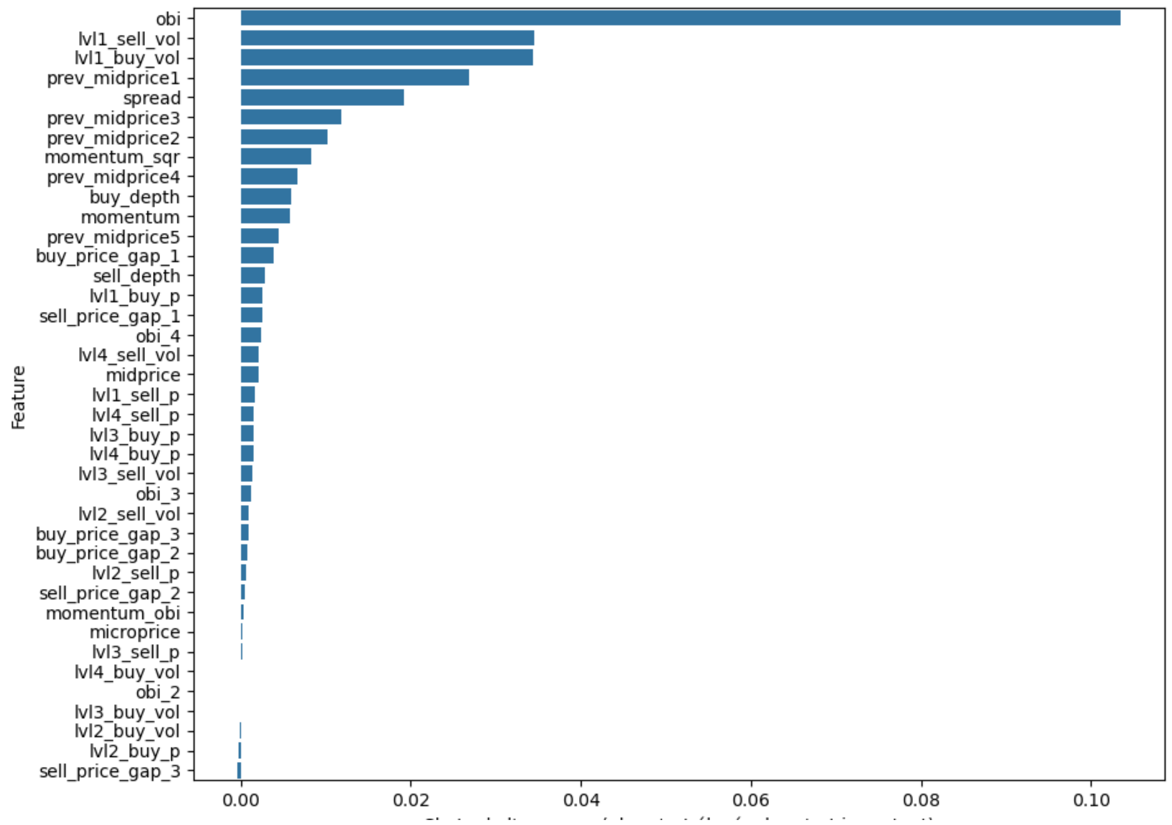


Figure 7: Feature Importance (accuracy decreases when the feature data is shuffled)

As expected when we saw that bid/ask volumes were the most important features, the first order OBI is the most important one when added. The spread is also an important feature, corroborating the fact that level 1 bid/ask prices are important.

5.4 McNemar's Test

We performed McNemar's test to statistically compare the performance of the two models (with and without the features). This test is appropriate because both models were evaluated on the same paired test set, allowing it to specifically assess whether their prediction disagreements are significant. The resulting Chi-squared statistic was 16.855, with an associated p_{value} of approximately 4.03×10^{-5}. Since this value is well below the conventional 0.05 threshold, we reject the null hypothesis. This provides strong statistical evidence that the improvement in accuracy is unlikely to be due to random variation, confirming that the addition of engineered features yields a statistically significant performance gain.

6 Conclusion

FNNs are performing quite well predicting the next midprice evolution direction (with around 72% accuracy) with respect to a small number of features. Feature engineering appears to guide the model in the right direction, though its overall impact on accuracy remains modest. This suggests that the raw data already encapsulate most of the relevant information, and that the engineered features provide only marginal additional value. However, the statistically significant result from McNemar's test confirms that this modest gain is not due to random variations.

The best model found through our investigation is a FNN with 3 layers, 97 neurons by layers, an ADAM optimizer, 12 epochs and ReLU activation functions for the hidden layers (and a Sigmoid for the output layer).