Imperial College London

MATH70116 Deep Learning

Imperial College London

Department of Mathematics

# Assessed Coursework

*Authors:*
Jérôme Allier (CID 06012483)
Charles Garrisi (CID 06009336)
Arthur Nahmias (CID 06020137)

*Professors:*
Dr. Lukas Gonon

December 5, 2024

# Contents

# Introduction

In this coursework, we explore the application of deep learning techniques to predict high-frequency mid-price changes of two major US stocks, using limit order book (LOB) data which provides a granular view of the supply and demand for a particular asset. Using deep learning methods offers promising advantages in such settings as the ability to model complex and high-dimensional relationships. By leveraging these techniques, we aim to model the intricate patterns embedded in the LOB data to forecast price movements with a higher degree of accuracy.

# 1 Data

The provided dataset contains a $200{,}000 \times 22$ array with detailed information extracted just prior to midprice changes. The midprice is defined as the average of the best bid and ask prices. The dataset includes a label column indicating the direction of the midprice change, encoded as 0 for a downward movement and 1 for an upward movement, as well as several features representing the LOB's first four levels on both the sell and buy sides, including price (in US dollars multiplied by 10,000), volume (in number of shares), and the five previous midprice change directions, encoded similarly to the labels. Note that there is no time series structure in the data: each row corresponds to a randomly drawn sample from a larger dataset covering the period from August 1, 2022, to November 1, 2024.

# 2 Data Cleaning

## 2.1 Missing and infinite values

We address any potential infinite or missing values calculations by replacing infinite values with NaN and subsequently filling any NaN values with zero. Fortunately, the dataset does not contain any NaN values initially so this is more a sanity check.

## 2.2 Data Analysis

Next, we can plot the distribution of several volume, price, and previous change features using boxplots to better understand their behavior. From these analyses, we observe that the price distribution is relatively well-scaled, with values generally of the same order of magnitude. The previous change features are equally distributed, which is consistent with what we observe in practice.

Conversely, when examining the volume features, we observe a very large variance as visible on Figure 1. While most values are of the order of $10^3$ or less, some values range between $10^4$ and $10^5$. These extreme values, particularly those exceeding $10^4$, represent approximately $5 - 10\%$ of the observations and are roughly 10 standard deviations above the mean, depending on the feature level.

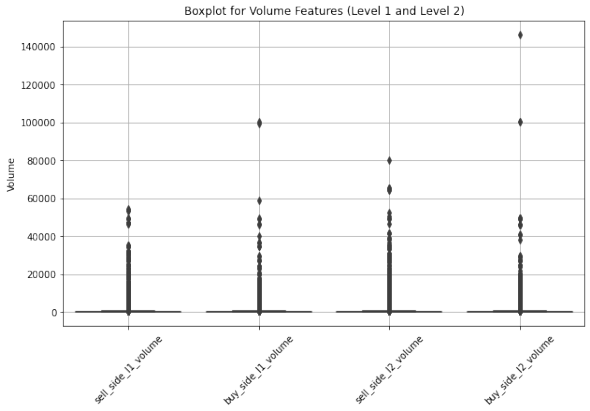


Figure 1: Boxplot for level 1 and 2 volume features

Given that our goal is to predict the mid-price change direction using data without a time series structure, it might be beneficial to exclude these extreme volume values from the training set to avoid their potential distortion of the model. Removing those outliers reduces the training set by approximately 0.5%, which is rather reasonable.

### 2.3 Scaling

Since we may apply PCA or other scale-sensitive algorithms later to reduce the dimensionality of the feature space, we standardize the data. PCA assumes that variance reflects the amount of information a feature provides, meaning that features with higher variance will have a greater influence on the principal components. Without standardization, features with larger variances will disproportionately affect the resulting components, potentially skewing the analysis. By standardizing, we ensure that all features contribute equally to the dimensionality reduction process.

## 3 Methodology

### 3.1 Baseline Model: Intuitive LOB

We begin with a baseline model rooted in a fundamental economic principle: when buy volume exceeds sell volume, prices are likely to rise, and vice versa. Given the multiple volume features available, we can try different combinations to identify the most effective predictors of price movement. There are different ways to approach this: we can analyze the first-order book level for a more granular intuition of market behavior, or we can aggregate the volume data across all levels to capture a broader market perspective. Assuming the levels follow each other sequentially, as the project statement suggests, we aggregate the volumes accordingly. This first naive model sets a target accuracy for subsequent models.

Using Level 1 volume alone leads to an accuracy of 0.68, which already represents a significant arbitrage. This analysis gives us a first intuition that the information is not evenly distributed across the dataset, suggesting it would be beneficial to focus on most important features.

### 3.2 A first neural network model

To challenge our baseline model, we start with a simple sequential architecture consisting of two hidden layers with ReLU activation, $L^2$ regularization, and Dropout for regularization purposes. We optimize the model using the Adam optimizer with a learning rate of 0.001 and binary cross-entropy as the loss function, given that the task is binary classification. This first model enables us to investigate two interesting approaches that might leverage our previous observations: SHAP values and PCA.

#### SHAP values

To interpret the model's predictions, we conduct SHAP (SHapley Additive exPlanations) analysis [1], which provides a clear understanding of feature importance and their impact on the model's output. To mitigate computational complexity and reduce overfitting risks, we resample the original data to create a subset of 5,000 instances. The model is trained on this subset for 10 epochs with a batch size of 32. Finally, SHAP values are computed for a sample of 50 instances from the resampled dataset.

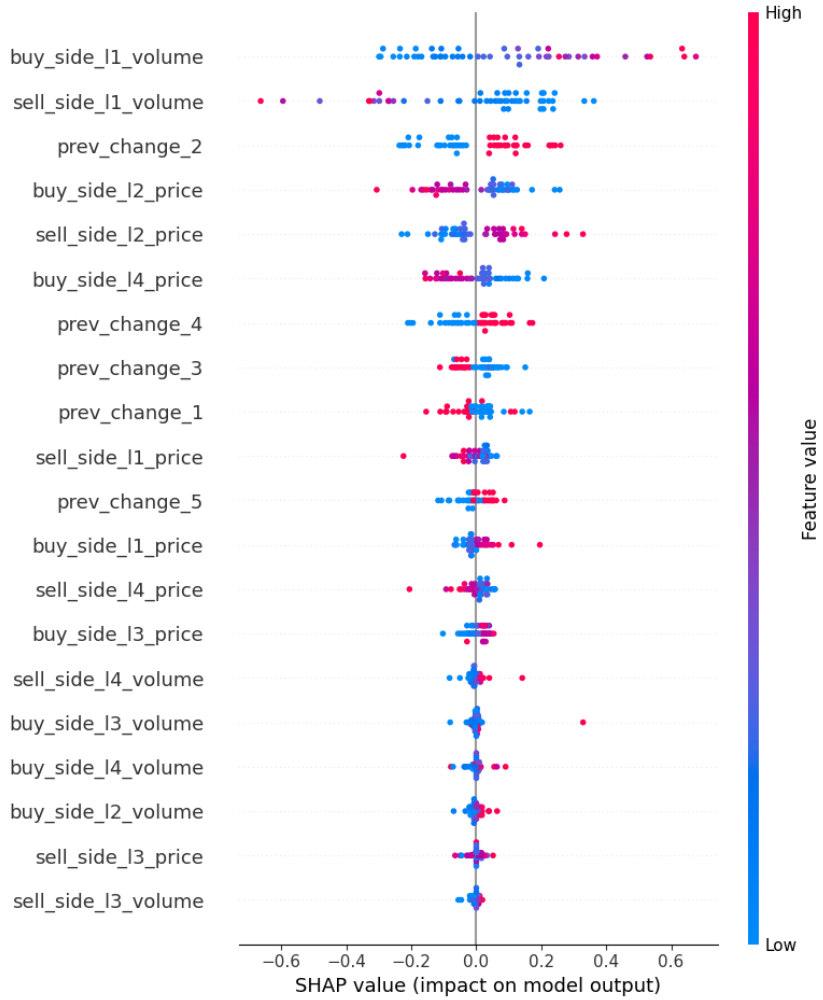


Figure 2: SHAP values for feature importance

From Figure 2, we can see that `buy_side_l1_volume` and `sell_side_l1_volume` are the most informative features. This explains why our first baseline model performed reasonably well. Additionally, note how `buy_side_l1_volume` exhibits positive contributions when feature values are high (pink), whereas low values (blue) tend to reduce the output. Conversely, `sell_side_l1_volume` shows the opposite pattern, which aligns with our goal of predicting midprice direction. Furthermore, `prev_change_2` also exhibits a clear pattern: low values (blue) lead to negative SHAP contributions, reducing the predicted output, while high values (pink) have a positive effect. This pattern highlights a monotonic relationship between `prev_change_2` and midprice change direction.

This analysis further suggests an asymmetry in the information distribution across the feature space, pointing to potential inefficiencies in the model. This motivates our next step: applying dimensionality reduction via PCA to capture the underlying structure of the data more effectively while mitigating the risk of overfitting.

### PCA

As we already scaled the data, we can perform a PCA and retain the first few components that explain most of the variance. Plotting the cumulative explained variance against the number of principal components in Figure 4 helps determine the optimal number of components to retain. The analysis sets the number of components to $N = 14$, capturing approximately 99.9% of the explained variance. Reducing the number of components further and excluding some variance information introduces potential instability in the model, as seen in our experiments where reduced variance weakened accuracy. For those reasons we retain the first 14 principal components.

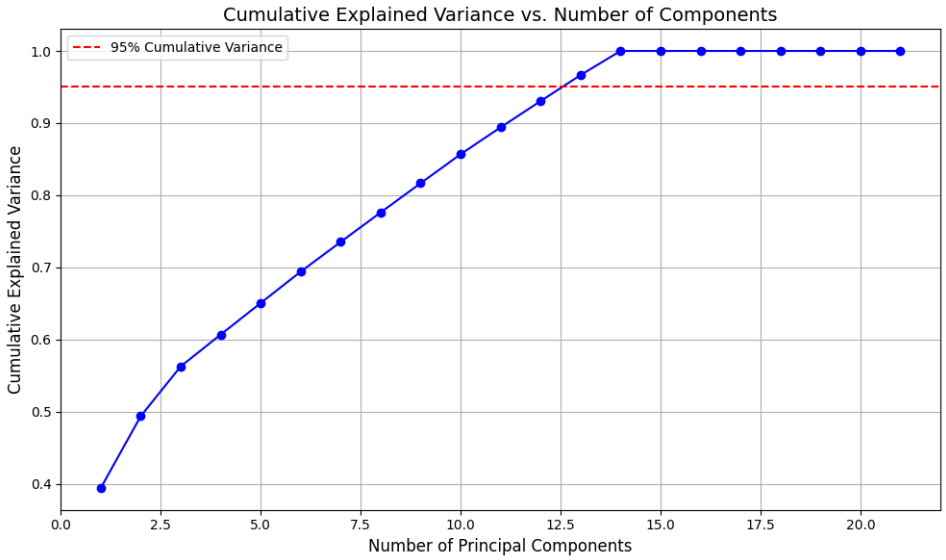


Figure 3: Cumulative variance vs. number of principal components

## 3.3 Building the final neural network

Now that we saw how critical it is to focus on the features space, let see how we can optimize each hyperparameter to achieve a better accuracy on the test set.

### Number of hidden layers

The first step is to determine the number of layers. The methodology applied was straightforward: we tested a classic neural network (NN) model with 2, 3, and 4 hidden layers. The following settings were used: dropout of 0.2, batch size of 100, and 30 epochs. For the number of neurons, we proceeded as follows:

- **2 hidden layers:** 64 neurons (1st layer), 32 neurons (2nd layer).
- **3 hidden layers:** 128 neurons (1st layer), 64 neurons (2nd layer), 32 neurons (3rd layer).
- **4 hidden layers:** 256 neurons (1st layer), 128 neurons (2nd layer), 64 neurons (3rd layer), 32 neurons (4th layer).

The code was not included in the notebook, as we chose to focus the analysis on tuning a single model. However, we found that the best-performing model had 3 hidden layers.

### Dropout, Batch size, Number of neurons and Activation function

To find optimal values for dropout, batch size, number of neurons, and hidden activation functions, we performed a manual grid search analysis over the following ranges, presented in Table 1

| Hyperparameter | Possible Values |
|---|---|
| **Dropout** | [0.1, 0.2, 0.3] |
| **Activation function** | ReLU, LeakyReLU |
| **Batch size** | [20, 50, 100, 200] |
| **Number of neurons** | [60, 80, 120, 160, 180, 200] |

Table 1: Hyperparameters and their possible values.

Best hyperparameters are summarized in Table 2. However, a batch size of 20 is too slow for training, so we increase it to 100. Additionally, 160 neurons is relatively high, so we reduce it to 128.

| Hyperparameter | Value |
|---|---|
| **Dropout** | 0.2 |
| **Activation function** | ReLU |
| **Batch size** | 20 |
| **Number of neurons** | 160 |

Table 2: Best hyperparameters

### Number of epochs

The methodology we use to choose the number of epochs is to train a model for 300 epochs, monitoring the accuracy on the validation set and select the best epoch with a patience of 10. We use the parameters we found in the hyperparameters section. The best epoch to take is 50. Figure 4 indeed shows that the accuracy on validation set curve meets the accuracy on train set curve around 50 epochs.

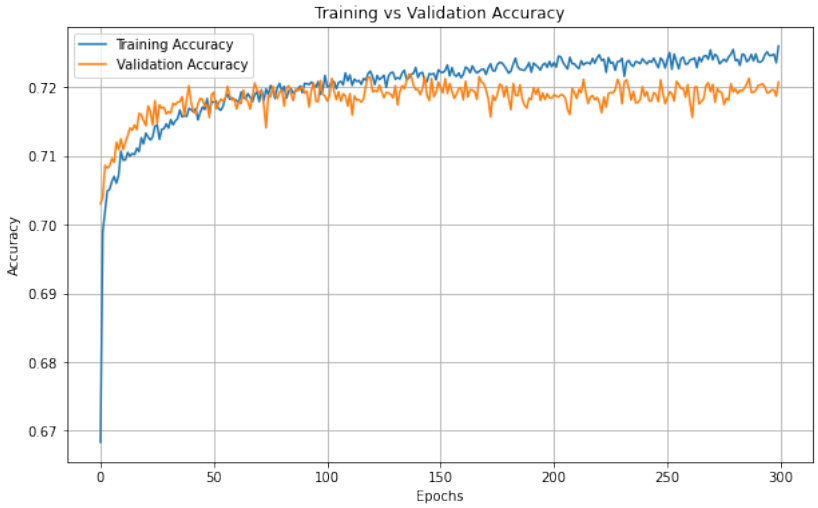


Figure 4: accuracy on test set vs accuracy on train set

### Bagging

The bagging model aims to capture more information in the dataset by building several models that overfit on different random parts of the dataset and then averaging the predictions of all of them. This approach follows the law of large numbers, which states that the average predictions of weak models will converge to the correct prediction.

The bagging model reuses the same truncated PCA dataset for training because this training dataset showed the best results. For the hyperparamters, we reduce the dropout at 0.1 because the aim of bagging is to overfit the models. The objective is to make a prediction based on an average prediction from several models instead of relying on a single model.

**Remark**: At the beginning, the aim was to build a bagging model with overfitted models in order to align with the law of large number. This model should work if we train a huge number of models that contain each a great number of neurons and epochs. We tried to train such a model but it failed because of the speed of execution.

## 3.4 Final Model

Eventually we come up with 4 slightly different models, based on 2 different architectures. We then select the one that gives us the best accuracy.

- **Unique Model Architecture:** we use a classic neural network with parameters determined using the grid search algorithm. The parameters are summarized in Table 3

| Parameters | Value |
|---|---|
| Dropout | 0.2 |
| Hidden layer activation function | ReLU |
| Batch size | 100 |
| Number of neurons in the first layer | 128 |
| Number of neurons in the second layer | 64 |
| Number of neurons in the third layer | 32 |
| Number of epochs | 50 |

Table 3: Parameters for our 1-model Neural Network Architecture

For some models, the number of epochs was extended arbitrarily with a patience of 15, as training performance can fluctuate due to dropout. This architecture is then used based on three different datasets, making three different models:

1. Raw dataset with standard rescaling.
2. Raw dataset with standard rescaling and PCA applied.
3. Dataset where we removed rows where any volume feature value exceeded $10^4$ and with standard rescaling and PCA applied.

- **N-Models (Bagging) Architecture:** we train 40 models using the bagging approach described earlier, using the parameters summarized in Table 4.

| Parameters | Value |
|---|---|
| Dropout | 0.1 |
| Hidden layer activation function | ReLU |
| Batch size | 100 |
| Number of neurons in the first layer | 128 |
| Number of neurons in the second layer | 64 |
| Number of neurons in the third layer | 32 |
| Number of epochs | 60 |
| Patience | 15 |

Table 4: Parameters for the Neural Network Architecture in Bagging Model

# 4 Results

## 4.1 Results for the Unique Model Architecture

For the unique model architecture, we obtain the following results:

- **Unchanged dataset (scaled only):**
  Test set accuracy: **71.9%**.
- **Dataset with PCA applied (14 components retaining 99.9999% of the variance):**
  Test set accuracy: **71.9%**.
- **Dataset with PCA applied and rows removed where volume feature values exceeded $10^4$:**
  Test set accuracy: **72.1%**.

In order to confirm the choice of removing extreme values in the training data set, we compute the prediction on a dataset containing only the extreme values with the model trained with extreme values removed and the model trained without removing extreme values. The accuracy on extreme values prediction for the model trained with extreme values is 69% and the accuracy on extreme values prediction for the model trained without extreme values is 70%. These results show that keeping the extreme values in training dataset doesn't improve its results on extreme values prediction.

### 4.2 Results for the Bagging Model

For the bagging model, we use the dataset with PCA applied and rows removed where volume feature values exceed $10^4$:

- **Bagging model:** Composed of models with the 1-model classic architecture for each model. Test set accuracy: **72.2%**.

### 4.3 Conclusion

The model with the best accuracy is the **bagging model**. Not only its accuracy is superior to the accuracy obtained using the baseline intuitive LOB model – which is reassuring because deep learning models are able to make better prediction –, but also it will be less prone to overfit on dataset B given it uses the approach of bagging.

## Conclusion

Accurately predicting short-term price movements in high-frequency trading is a highly challenging problem and remains difficult to achieve. Nonetheless, our models demonstrated promising results, achieving an accuracy of 72%. This highlights the potential of deep learning techniques in financial prediction tasks, despite the inherent complexity and volatility of the financial markets.

## Bibliography

**[1]** SHAP Documentation