Highlights

A novel feature engineering approach for high frequency financial data

Pablo Mantilla,Sebastián Dormido-Canto

- It is proposed a feature engineering for high frequency financial data based on time series segmentation.
- This methodology allows to extract and analyze variables by intraday trends, as well as feeding artificial intelligence models in order to forecast response variables in future trends.
- This feature engineering is applied to estimate high frequency volatility, duration and direction linked to future intraday trends.
- Experimentation was conducted using high frequency financial data from the Brazil Stock Exchange.

A novel feature engineering approach for high frequency financial data

Pablo Mantilla, Sebastián Dormido-Canto

Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain

ARTICLE INFO

Keywords:
Artificial intelligence in finance
High frequency financial data
Time series segmentation
Intraday volatility
Directional forecasting

ABSTRACT

It is proposed a feature engineering for high frequency financial data based on constructing dynamic data subsets, defined by time intervals in which high frequency trends occur. These intervals are obtained through time series segmentation. This methodology allows to extract and analyze variables by intraday trends, as well as feeding artificial intelligence models in order to forecast response variables in future trends. Furthermore, in order to show how to use this feature engineering, this methodology is applied to estimate high frequency volatility, duration and direction linked to future intraday trends, developing multiclass classification models based on the machine learning method XGBoost. Experimentation was conducted using high frequency financial data from the Brazil Stock Exchange, corresponding to 206 trading days related to 20 listed assets from this financial market.

1. Introduction

In the new technology era, artificial intelligence has become a central issue for data science, with applications in numerous branches of science. One of these fields is finance, where machine learning techniques have emerged as powerful tools to extract knowledge from high frequency data and have become an increasingly important research area to tackle high frequency financial data analysis and forecasting. On the other hand, the widespread use of electronic systems on the stock markets has made it possible to store these data with the high precision with which it is generated. This situation has caused a boom in the application of learning algorithms to this type of data, as it is showed in the comprehensive review presented in (Ntakaris, Magris, Kanniainen, Gabbouj & Iosifidis, 2018).

These data are characterized by being generated at irregular intervals with precision of milliseconds or even higher frequencies, which causes high volumes of data containing multiple variables. In other ways, we are not dealing with usual time series, since these data come from buy and sell orders that arrive to the market system, where some of these orders intersect, giving rise to what are called trades, and others remain queued, whose processing consists of the reconstruction of the so-called limit order book. The final data contain multiple variables that evolve over time, which has contributed to the fact that these data have been the subject of a multitude of research works from a statistical point of view (Hautsch, 2012). However, the very nature of these data causes that their distributions are not maintained over time, so that this circumstance has created a suitable framework to address the resolution of forecasting problems on these data with other kinds of methods, based on artificial intelligence, in which it is not necessary to make assumptions on the distributions of these data (Nousi, Tsantekidis, Passalis, Ntakaris, Kanniainen, Tefas, Gabbouj & Iosifidis, 2019).

The progressive scientific interest in the combination of knowledge fields related to high frequency data analysis and artificial intelligence methods for forecasting purposes is evident in the multiple references cited in this scientific article, most of which have been published in recent years. This growing body of literature recognizes the importance of using machine learning methods to analyse, modeling, and forecast financial markets. Moreover, there is strong private sector interest in this research topic, on account of the fact that investment banks, hedge funds and other wealth management firms are in the race for successful forecasts, in addition to higher returns at lower risks.

In order to apply machine learning techniques on raw data from financial markets, and solve multiple high frequency forecasting problems, it is necessary to extract relevant features from these data, so they must be previously analyzed and treated, using techniques that allow to achieve this objective. The literature reviewed contemplates feature extractions on sets of observations with a constant number of elements, making forecasts on a fixed forecast horizon. This is the situation of the usual forecasting problems in this research field, such as volatility estimation and directional forecasting linked to price movements. In such cases, the cited forecasts are made based on a previously defined sampling scheme and forecast horizon. However, volatility is a variable that presents clustering in its values and does not have symmetry with bullish and bearish movements (Engle & Patton, 2001), so it seems reasonable to group volatility values by intraday trend movements. In addition, high frequency quotes tend to experience intraday trends with irregular duration. These particularities arouse interest in knowing what would be the evolution of a relative volatility linked to intraday trends durations, or what direction or duration a future intraday trend would have. But the posed problem is not limited to these variables, there are many others whose evolution could be linked to intraday trends.

The research objective is to extract features from intraday trends in the trades series and from the limit order book states at every time in the trend movements. Therefore, a methodology that meets this objective was developed, which

[⋆] This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

pmantilla1@alumno.uned.es (P. Mantilla); sebas@dia.uned.es (S. Dormido-Canto)

provides answers to the questions previously formulated, since it allows to analyze variables over time intervals in which intraday trends occur, and also allows to estimate how a variable will behave in future intervals, using values from variables in previous intervals, treating them as regressors that explain the variation in the response variable using artificial intelligence models.

The sampling and forecasting scheme in this methodology constitutes a different approach from the traditional one, which usefulness is aimed at solving a specific problem with high frequency financial data, where the purpose is to analyze variables over time intervals in which intraday trends occur and estimate dependent variables in future intervals. Therefore, the main contribution is a methodology to extract features from subsets with variable composition.

The general structure of this document is organized in seven sections, including this introductory part. The state of the art related to this experimental research is reviewed in Section 2. In Section 3, the research problem is stated and it is raised the proposal. Section 4 includes the application of the developed methodology to forecasting problems, in which the artificial intelligence algorithm *extreme gradient boosting* is applied. Section 5 contemplates the extensive and laborious experimentation conducted in this research, in which the designed methodology is executed and evaluated and in which the developed feature engineering is applied to three specific forecasting problems. Results are discussed in Section 6. Finally, the concluding remarks are stated in Sección 7.

2. Related work

In general terms, the state of the art linked to this research is framed in the scientific field related to the application of artificial intelligence methods to high frequency financial data. Specifically, it is a feature engineering for high frequency data, where the existing methods start from data subsets with fixed composition, from which the corresponding features are extracted. On the contrary, the developed feature engineering contemplates an additional previous step, which consists of constructing data subsets with variable composition, according to a specific criterion. For this reason, it is considered that it has posed a new problem in artificial intelligence using high frequency financial data. However, there are works related, in some way, to this research proposal, which are commented below.

The methodology exposed in this article is applied to forecast intraday volatility linked to high frequency trend movements, where it is used a *dynamic blocking scheme* without overlapping, that is, it is sequentially computed volatility over intervals with variable duration, which did not share observations between them. It has not noted that this type of sampling scheme is collected in the literature on high frequency volatility. Nevertheless, volatility has been treated with artificial intelligence models.

An example of volatility forecasting with machine learning models is found in (Ramos-Pérez, Alonso-González & Núñez-Velázquez, 2019), where they used *Gradient Descent Boosting*, *Random Forest (RF)*, *Support Vector Machines (SVM)* and *Artificial Neural Networks (ANN)* to forecast the volatility of the S&P500 index. In this publication, they used the standard deviation of returns as a proxy for volatility, calculated with a window size equal to 5 observations. Also in (Liu, 2019) were used SVM, *Long Short-Term Memory (LSTM)* networks and daily data from the cited index and *Apple Inc.*, where the objective was to forecast volatility on 1 and 3 days forecast horizon.

Other authors developed machine learning models to forecast intraday volatility. In (Guo, Bifet & Antulov-Fantulin, 2018), they forecasted short-term volatility using historical volatility and order book data. They used RF, *Extreme Gradient Boosting (XGBoost)* and LSTM neural networks. In (Peng, Albuquerque, Camboim de Sá, Padula & Montenegro, 2018), they forecasted cryptocurrency volatility using *Support Vector Regression (SVR)* and daily and intraday data, on 1 hour forecast horizon for the intraday case. In (Doering, Fairbank & Markose, 2017), they forecasted high frequency volatility for a 20 events forecast horizon. They used a binary classification model with *Convolutional Neural Networks (CNN)*, and used *limit order book (LOB)* data from *Barclays PLC* stock, which is listed on the London Stock Exchange.

In high frequency financial data analysis, directional forecasting is a common problem addressed with machine learning approaches. This issue has gained importance in light of recent results, which prove that machine learning methods are the most suitable option to deal with this problem. Studies over the last decade have provided important insights into this challenge, in which directional forecasts are made based on historical data. Directional forecasting can consist of estimating future price movements on the next observation, but it can also forecast at a higher forecast horizon, on multiple observations.

Previous research works have focused their forecasting on one observation horizon, on a fixed number of observations or over a constant time interval, feeding their machine learning algorithms with features extracted from high frequency data over a constant number of observations or in a fixed time interval. Therefore, no previous works have been found whose objective has been to use machine learning techniques in high frequency financial data to make directional forecasts about future intraday trends, extracting data from subsets formed by previous trend movements. Hence, the methodology presented in this article is applied to high frequency directional forecasting with an unusual approach. Below, multiple works that somehow performed directional forecasting with machine learning methods are reviewed.

In (Fletcher & Shawe-Taylor, 2013), they proposed SVM to classify price direction into three classes and different forecast horizons. The input consisted of features extracted from order book updates, with sampling at a frequency of 1 second. They used EURUSD data for one day. Since then, other articles have covered the use of SVM to address directional forecasting, as in (Kercheval & Zhang, 2015), where they used a multiclass SVM to forecast the mid-price and the bid-ask spread direction, with a trading day data for 5 stocks listed on the NASDAQ. The input was multiple price and volume features at different levels from the order book. They made forecasts on a 5, 10, 15 and 20 events forecast horizon.

Other publications have raised the use of SVM and neural networks. In (Tsantekidis, Passalis, Tefas, Kanniainen, Gabbouj & Iosifidis, 2017a), they compared methods based on CNN, *Multilayer Perceptron (MLP)* and SVM to perform a multi-class classification of future mid-price movements on three different forecast horizons: 10, 20 and 50. They made data labeling comparing the average of k previous mid-prices with the average of the following k mid-prices, considering a certain threshold that split three classes. They ran experiments with data composed of 10 LOB levels from 5 stocks traded on the Finnish stock market, during the period from June 1 to June 14, 2010. Also in (Nousi, Tsantekidis, Passalis, Ntakaris, Kanniainen, Tefas, Gabbouj & Iosifidis, 2019), they forecasted the mid-price direction with a multiclass SVM model and with neural networks. They used forecast horizons equal to 1, 5 and 10 mid-prices. In the last two cases, they forecasted the average price. They extracted features every 10 limit order events, and used data from 5 listed stocks on the Finnish stock market, relative to 10 LOB levels and 10 trading days, as of June 14, 2010. They extracted features from raw data as inputs for their machine learning models, in addition to others obtained by *Autoencoders* and *Bag-of-features* models. In (Tsantekidis, Passalis, Tefas, Kanniainen, Gabbouj & Iosifidis, 2017b), they used LSTM networks to forecast the future mid-price direction with 10 levels of LOB data, corresponding to 5 shares listed on the Finnish stock market, during a 10-day trading period. They made labeling data with three classes, comparing the average of previous mid-prices with the average of next mid-prices, considering a specific threshold. They conducted experiments for three different forecast horizons: 10, 20, and 50. Finally, they compared their results with those obtained with a Linear SVM and with an MLP network.

Multiple publications discuss different neural networks types to forecast price direction. In (Dixon, Klabjan & Bang, 2017), they implemented a deep neural network to classify the future price direction over a future time interval. The data contained mid-prices in 5-minute intervals, related to 43 commodity and currency futures, from March 1991 to September 2014. The input features contained price differences and lags from 1 to 100, price moving averages with window sizes from 5 to 100, pairwise correlations between returns and returns. In (Doering, Fairbank & Markose, 2017), in addition to forecast volatility, as discussed above, they chose a deep CNN to classify the price trend direction in three classes on a 20 events future horizon, where they estimated the direction of arithmetic returns. The data consisted of *Barclays PLC* message books from the London Stock Exchange, between June 2007 and June 2008. In total, 217 trading days. The input data were the previous order book states and the events flow. In (Arévalo, Nino, León, Hernandez & Sandoval, 2018), they suggested a deep neural network with *wavelet* to forecast the next *pseudo-log-return* in 1 minute, from which they got the average price in the next minute. They used wavelets because high frequency data shows simultaneous transactions, as well as price jumps and low variance. The wavelets had length equal to eight and compressed the corresponding *tick-by-tick* transactions in a given minute. They conducted experiments with tick-by-tick data from 19 randomly selected companies listed on the Dow Jones Industrial Average index, during the period January 2015 to July 2017. The neural network input consisted of 27 variables: 1 minute pseudo-log-returns and wavelet vectors compressed in 1 minute, both with lag equal to 3. In (Dixon, Polson & Sokolov, 2019), they used a multi-layer deep learning network to classify the mid-price movement over a future time interval. The data corresponded to the E-mini S&P500 message book during August 2016. They preprocessed their data with an elastic network and extracted features from 5 levels of LOB states. In (Sirignano & Cont, 2019), they fed LSTM networks in sequences of 100 and 5000 lags with historical prices and order flows relative to multiple values from the US stock market, in order to forecast whether the next price direction would be increasing or decreasing. They built multiple models using data from 1000 NASDAQ stocks. In (Ntakaris, Mirone, Kanniainen, Gabbouj & Iosifidis, 2019), they used MLP, CNN and LSTM models to forecast the mid-price movement direction, and they estimated the order book events generated until the direction change occurred. The neural networks input included technical and quantitative indicators, time-sensitive and insensitive features, as well as features extracted automatically. They made two different forecasts. On the one hand, they estimated the price direction, up or down, and when the next event would occur, through classification and regression, respectively. With this approach, they extracted features with a sliding window of size equal to 10 events and step equal to 1 event. On the other hand, they made a multiclass classification every 10 events, for which they used an average with 10 future events, with feature extraction every 10 observations without overlapping. The data set corresponded to 10 trading days, 5 stocks from the Finnish stock market and 2 stocks from the US stock market, for 10 LOB levels in milliseconds time frame. In (Passalis, Tefas, Kanniainen, Gabbouj & Iosifidis, 2019), they combined the Bag-of-Features method with deep neural networks to solve a multiclass classification problem regarding the future average mid-price direction. They considered the 15 most recent feature vectors for every *timestep* and two forecast horizons: 10 and 50 timesteps. The data consisted of 10 LOB levels from 5 shares listed on the Helsinki Stock Exchange, during the period from June 1, 2010 to June 14, 2010, which is 10 trading days. In (Zhang, Zohren & Roberts, 2019), they designed a deep learning model based on convolutional layers and LSTM units to forecast future price movements on three different forecast horizons: 20, 50 and 100. They used two different data sets. The first set had 10 LOB levels relative to 5 shares from the London Stock Exchange during the period of 1 year. The second set was the public data set called FI-2010 (Ntakaris, Magris, Kanniainen, Gabbouj & Iosifidis, 2018). They selected 40 features for each event: price and volume of each level, for both LOB sides, from 100 previous order book states.

Other methods have also been used to solve the same problem. In (Felker, Mazalov & Watt, 2014), they presented a method based on the feature-weighted Euclidean distance to the centroid of a training cluster. They used multiple technical indicators to forecast price changes, 10 to 2000 milliseconds before they occurred. Their forecasts were made when the classification confidence was high enough, from the record of every new market event. In (Tran, Magris, Kanniainen, Gabbouj & Iosifidis, 2017), they used *Multilinear discriminant Analysis* and *Weighted Multichannel Time-series Regression* to forecast the mid-price movement. The input was a time series tensor representation. Labeling corresponded to three mid-price movement classes in the next 10 order events. They utilized LOB data from 5 stocks listed on the Finnish stock market, relative to 10 trading days.

3. Methodology

Time series trends have a variable number of observations. It is intended to extract features from intraday trends and from the limit order book states at the times of these trends. This is the problem proposed to solve, whose solution allows to analyze multiple variables in subsets defined by time intervals in which intraday trends occur, and also allows to estimate response variables in future trends, by feeding artificial intelligence models with the extracted features.

This problem can be satisfactorily solved through a multi-stage feature engineering. In the first step, the transaction time series are partitioned, and segments with a variable number of observations are obtained, which are conditioned by the irregular durations of intraday trends. The intervals obtained form subsets of values which correspond to variables related to trades. In the next stage, the order book states at trades times are synchronized, to obtain variables associated with each order book state, which form the second subset comprised in the aforementioned time interval.

The methodology scheme is shown in figure 1. On the variables set in each segment, several transformations to obtain the cited features are performed, which form the input of artificial intelligence models.

3.1. Segmentation

The appropriate technique to obtain these high frequency intervals is time series segmentation. Generally speaking, the trades series are segmented to obtain the *breakpoints* at which intraday trends change their direction in order to accurately obtain the intraday trends that occur between these breakpoints. In these segments, features from trades and order book states are extracted, which are linked to intraday trend directions.

The scientific literature collects different ways of approaching the problem. Reviews of segmentation methods

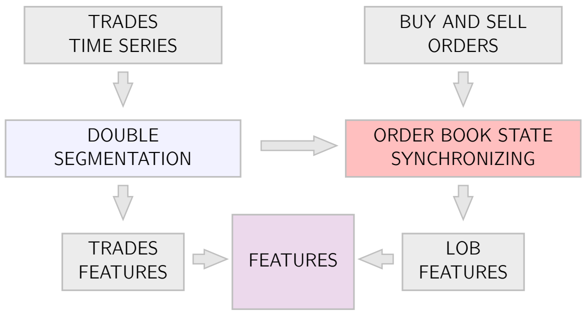


Figure 1: Feature engineering scheme

can be found in (Keogh, Chu, Hart & Pazzani, 2004) (Lovrić, Milanović & Stamenković, 2014). Also in (Si & Yin, 2013), different methods to address the segmentation problem are reviewed. The technique that obtains an optimal or exact solution to the segmentation problem uses dynamic programming (Bai & Perron, 2003) (Zeileis, Kleiber, Krämer & Hornik, 2003). It is very precise, but its main disadvantage is its high execution time, due to the fact that this segmentation method has algorithmic complexity of quadratic order, so the execution time increases at most quadratically as the number of observations increases.

High frequency data is recorded by days independently and we could consider applying the exact method directly, but most days have such a high number of observations that it would not be feasible to apply the optimal method directly, since the computational time would be considerably high. To solve this problem, we apply the exact method in two phases. First, we add the original data to a period of lower frequency, reducing the number of observations and respecting the main trends of the series. That is, we divide each daily series into constant intervals of specific periodicity and select the last observation of each interval. Then, we apply the exact method to the resulting series and obtain the breakpoints that delimit the trends of the reduced series. Next, we move these breakpoints to the original series, obtaining primary segments. Each of these segments is a time series that can be re-segmented. We apply the exact method again and obtain the final breakpoints, which delimit the final trends on which we are going to extract the features.

The method used to obtain the breakpoints in the two performed segmentations was implemented in (Zeileis, Leisch, Hornik & Kleiber, 2002), where the optimal position of the breakpoints is achieved by minimizing the residual sum of squares and the optimal number of breakpoints is obtained by minimizing the Bayesian Information Criteria. An example of the final result is shown in Figure 2.

3.2. Feature processing

We start from the trades series segmented. For time series with n observations and m breakpoints, we have a set of $m+1$ segments, fragments or time intervals from this time series

$$S_1, S_2, ..., S_{m+1} \tag{1}$$

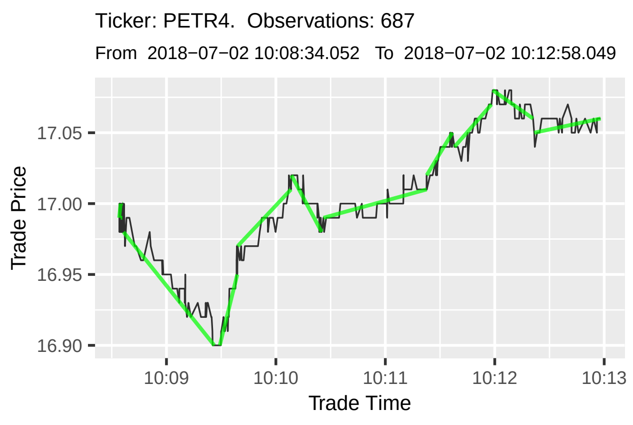


Figure 2: Segmentation example

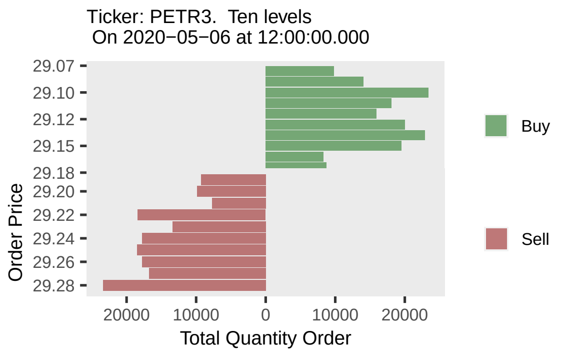


Figure 3: LOB State

Each interval has a variable number of observations e_j and a set of p trades features is extracted in each segment S_j,

$$X_{j_1}, X_{j_2}, ..., X_{j_p} \tag{2}$$

We obtain each feature X_{j_t} from the prices y_i, times x_i, transactions T_i or volumes v_i contained in an interval S_j, such that

$$X_{j_t} = f_t(\{y_i\}_{i=i_{j-1}+1}^{i_j}, \tag{3}$$

$$\{x_i\}_{i=i_{j-1}+1}^{i_j}, \tag{4}$$

$$\{T_i\}_{i=i_{j-1}+1}^{i_j}, \tag{5}$$

$$\{v_i\}_{i=i_{j-1}+1}^{i_j})$$

The trades feature X_{j_t} in each interval S_j with $t = 1, 2, ..., p$ is a function f_t that can be any combination of the variables: prices, timestamps, transactions or volumes.

As we can see in figure 4, we extract other features from the LOB states at times set by transaction prices. For an interval S_j, we have the q features from the LOB states

$$Z_{j_1}, Z_{j_2}, ..., Z_{j_q} \tag{6}$$

Figure 3 represents an example of the LOB state at time x_i, with depth of ten levels on each side of the market.

The best buy and sell orders, from the point of view of being easily crossed, are those with the highest and lowest prices, respectively. These orders are located in the middle of the chart, separated by the *spread*, which is the difference in price between said orders. Level 1 in the order book corresponds to these orders, and the next levels are the blocks that follow the best buy and sell orders, forming what is called the LOB depth. In mathematical terms, the LOB status at every time x_i from the trades series is a matrix with the prices and volumes from all levels formed by buy and sell orders, that is, the buy and sell depths, named as d_B and d_S, respectively.

The feature extraction process using each LOB state has three steps, as shown in figure 4. Let an interval S_j formed by observations i from the trades series, and let the LOB states i at every time x_i from each segment observation. Every LOB state is a matrix with prices and volumes from buy and sell orders ($P_{i_{B_l}}$ $V_{i_{B_l}}$ $P_{i_{S_l}}$ $V_{i_{S_l}}$) corresponding to each level l existing at time x_i. From these four variables, we obtain other variables with single values at every time, obtaining vectors $u_{i_{j-1}+1}, u_{i_{j-1}+2}, ..., u_{i_j}$ referred to each time x_i. The resulting set of vectors forms the matrix M_j, which contains the variables referring to times x_i in the interval S_j. Finally, we obtain the values related to the set of features $Z_{j_1}, Z_{j_2}, ..., Z_{j_q}$ corresponding to said interval.

Ultimately, the features depend on the prices and volumes corresponding to the buy and sell orders, but we establish different types of features, depending on the input data. In general terms, a feature Z_{j_o} is obtained from a subset with a number of input buy levels $1, 2, ..., d_B$ and sell levels $1, 2, ..., d_S$ within the interval S_j. According to this criterion, we define the function g_o that obtains the feature Z_{j_o} in a segment S_j with observations i, for $o = 1, 2, 3, ..., q$

$$Z_{j_o} = g_o(\{\{P_{i_{B_l}}, V_{i_{B_l}}\}_{l=1}^{d_B}, \{P_{i_{S_l}}, V_{i_{S_l}}\}_{l=1}^{d_S}\}_{i=i_{j-1}+1}^{i_j}) \tag{7}$$

The variables $P_{i_{B_l}}$ and $V_{i_{B_l}}$ are the buy prices and buy volumes from the LOB level l at time x_i in the interval S_j. Likewise, $P_{i_{S_l}}$ and $V_{i_{S_l}}$ are referred to sell orders.

Therefore, in each time interval two sets of different features are extracted, according to the scheme represented in figure 5.

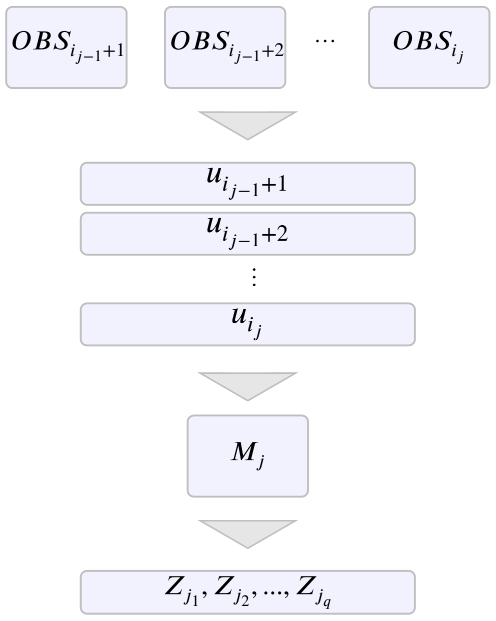
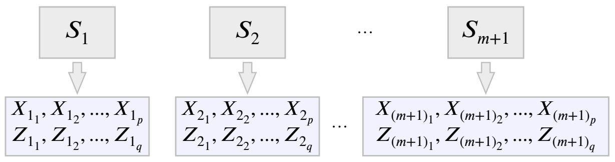


Figure 4: LOB feature extraction process



S_1
S_2
$\cdots$
S_{m+1}
$X_{1_1}, X_{1_2}, ..., X_{1_p}$ $Z_{1_1}, Z_{1_2}, ..., Z_{1_q}$
$X_{2_1}, X_{2_2}, ..., X_{2_p}$ $Z_{2_1}, Z_{2_2}, ..., Z_{2_q}$
$\cdots$
$X_{(m+1)_1}, X_{(m+1)_2}, ..., X_{(m+1)_p}$ $Z_{(m+1)_1}, Z_{(m+1)_2}, ..., Z_{(m+1)_q}$

Figure 5: Segment features

4. Application

In order to apply the developed methodology to a forecasting problem, models based on an artificial intelligence method for classification problems are built, although the designed feature engineering could be used with innumerable machine learning methods.

The input features needs prior preparation to feed these models, so once we obtain the features, we proceed to labeling and embedding, leaving the input ready for use in modeling, as shown in figure 6.

4.1. Labeling

Once we have segmented the trades series and extracted the respective features, we classify the segments according to the corresponding response variable. It is constructed a labeling to decide the groups in which it was going to classify the segments. Said labeling depends on the values of the response variable, and is composed of different labels, referring to classes of the dependent variable, where each one identifies a region of values. Thresholds or limits for these regions are established in advance. The criterion was to obtain three balanced regions from the train subset data, so it was defined the limits using quantiles with probabilities of 0.3333 and 0.6666 of the values of the dependent variable.

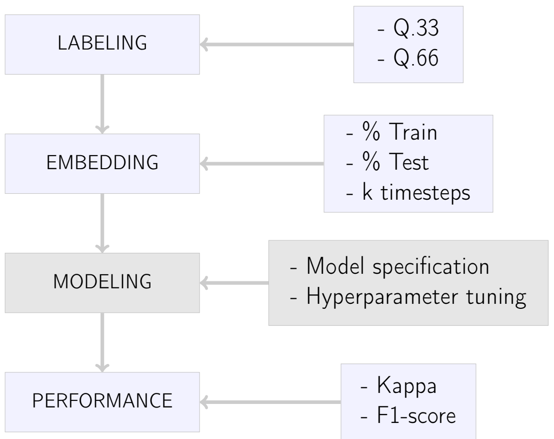


Figure 6: Machine learning scheme

4.2. Embedding

We have vectors of features that come from the trades series and the order book states, which they were extracted in each time interval obtained by segmenting this series. From these vectors, it is built the set of samples to perform the embedding. In the case of time series, it is usual to include variables with lags, and that the advance over the segments series has a specific step. It is a sliding window with size equal to the number of lags and advance equal to the considered step. Let k be the number of lags. Then, it was chosen a step equal to 1 and a number of lags equal to k, so the samples structure is the one shown in figure 7.

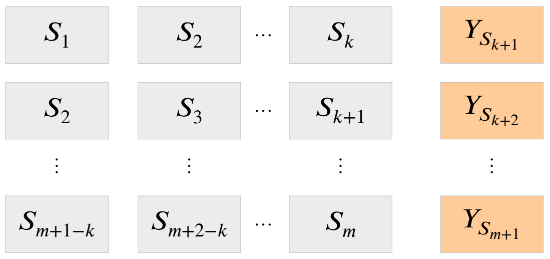


Figure 7: Embedding scheme

The first sample is a vector formed by the features corresponding to the first k segments, which represent the explanatory part. The response variable corresponds to the label that this variable has in the time interval $k + 1$. We advance a step in the segments series and build the second sample, starting with the features relative to the second segment and with an explained variable equal to the segment label $k + 2$, and so on.

4.3. Modeling

It was used the algorithm called *Extreme Gradient Boosting* or *XGBoost* (Chen & Guestrin, 2016). In this section, it is briefly reviewed theoretical foundations related to this method, which is based on the Gradient Boosting method (Friedman, 2001) and is an implementation of *Gradient Boosted Trees*.

Let X_i and Y_i be the regressors and the response variable, respectively. The objective is to forecast the response variable $\hat{Y}_i$, which in classification models represents the labels for each class. The explanatory variables X_i constitute the b features extracted from the data, each of which has a weight in the model, defined by the θ parameters. The model is trained to learn and obtain the parameters that provide the value for the response variable closest to a given one. The difference between the estimated value corresponding to the dependent variable and the observed one constitutes the residual.

If we have a samples and b features, with $X_i \in \mathbb{R}^b$ and $Y_i \in \mathbb{R}$, we forecast the output through K additive functions for a set of trees.

$$\hat{Y}_i = \phi(X_i) = \sum_{k=1}^{K} f_k(X_i)\,; \quad f_k \in \mathcal{F}, \tag{8}$$

where $\mathcal{F}$ is the space of trees , T the number of leaves in each tree and q the tree structure. Each function f_k refers to a tree structure q and leaf weights w.

The function that measures the error is the loss or cost function. The target is to minimize the objective function, which corresponds to the sum of the loss function and the regularization function, which measures the model complexity. This last function is introduced in order to penalize overfitting. If the regularization term were zero, we would have the classic gradient tree boosting.

$$\begin{gathered} \mathcal{L}(\phi) = \sum_{i=1}^{a} l(\hat{Y}_i, Y_i) + \sum_{k=1}^{K} \Omega(f_k) \\ \Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2, \end{gathered} \tag{9}$$

where γ and λ are regularization parameters.

Optimization is done from training in an additive way. *Gradient Boosted Trees* combine *boosting* and *gradient descent*. In the former, the decision trees are built sequentially, so that the next tree is built on the residuals of each tree, reducing the error in each step. Finally, the objective function is the sum of those obtained in each tree.

The gradient descent is an optimization method that is based on the vector generalization of the derivative, whose objective is to optimize the total minimum. The purpose is to obtain the best parameters θ to have the minimum residual.

The variable importance is referring to the relative contribution of each feature to the XGBoost model, based on the participation of each feature in each tree division, which is weighted over a total value equal to 1. If a specific feature, like an X_t feature from trades, has a higher contribution than other features, then the feature X_t is more relevant for making forecasts. The total sum of the scores associated with each feature is equal to 1.

4.4. Performance metrics

In this research, they were used multiclass classification models to forecast the corresponding *labels* in future time intervals. With the resulting labels, and those corresponding to the test data, the confusion matrices were calculated, where they were compared the observed classes with those estimated. The diagonal of the confusion matrix includes the correctly estimated cases of each class, while the rest of the elements in the matrix correspond to errors in each class.

From these matrices, we can calculate multiple performance metrics, including *kappa* and *F1-score* metrics. The metric *kappa* considers the precision that could be obtained by chance, penalizing this aspect in the model precision (Kuhn & Johnson, 2013). These performance metrics were calculated with the procedure below.

Let A be the confusion matrix for a three-class classification problem.

		Observed		
		High	Medium	Low
Forecasted	High	a_{11}	a_{12}	a_{13}
	Medium	a_{21}	a_{22}	a_{23}
	Low	a_{31}	a_{32}	a_{33}

Let S_{ij} be the sum of all the elements a_{ij} from A, $tr(A)$ the trace of A and let O and E be the observed precision and the expected, respectively.

$$S_{ij} = \sum_{i,j=1}^{3} a_{ij}\,; \quad tr(A) = \sum_{i=1}^{3} a_{ii} \tag{10}$$

The observed precision O was obtained from the following expression.

$$O = \frac{tr(A)}{S_{ij}} \tag{11}$$

Using the total sum and the sums of the elements in each column and each row, it was attained the expected precision E.

$$E = \frac{S_{i1}}{S_{ij}}.\frac{S_{1j}}{S_{ij}} + \frac{S_{i2}}{S_{ij}}.\frac{S_{2j}}{S_{ij}} + \frac{S_{i3}}{S_{ij}}.\frac{S_{3j}}{S_{ij}} \tag{12}$$

From the observed and expected precision, it was obtained the *Cohen's kappa* (Cohen, 1960).

$$kappa = \frac{O - E}{1 - E}, \tag{13}$$

which can have values within the range $[-1, 1]$. A value equal to zero means that there is no agreement between the observed classes and the forecasts. A value equal to one represents the perfect forecast, while negative values indicate that the forecast is in the opposite direction to the truth.

In order to have a single measure for each class, the $F1-score$ was calculated as follows. Let the true positives TP, false positives FP, true negatives TN and false negatives FN for the first class, where TN corresponds to those elements that do not belong to column and row one. The procedure is similar for the other two classes, except that the TP are on the matrix diagonal for each class, so the TN related to the second class would correspond to the elements not included in the column and row 2, and so on.

		Observed		
		High	Medium	Low
Forecasted	High	TP	FP_1	FP_2
	Medium	FN_1	TN_1	TN_2
	Low	FN_2	TN_3	TN_4

$$TP = a_{11} \tag{14}$$

$$FP = a_{12} + a_{13} \tag{15}$$

$$TN = a_{22} + a_{23} + a_{32} + a_{33} \tag{16}$$

$$FN = a_{21} + a_{31} \tag{17}$$

First, it was obtained the metric *precision*, which measures the proportion of correct forecasts over the total of positive forecasts,

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

Second, it was calculated the metric *recall*, which represents the proportion of positives observed with a correct forecast,

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

Finally, the $F1 - score$ was achieved, as the harmonic mean of the metrics *precision* and *recall*,

$$F1 = \frac{2PR}{P + R}, \tag{20}$$

where P and R correspond to the *precision* and *recall* metrics, respectively.

It was followed the same procedure to obtain the *F1-score* relative to the other two classes, with the particularity that TP, TN, FP and FN were obtained according to the positions of the respective matrix for each class, as indicated above.

5. Experimentation

In this research, real data and supercomputing resources have been used, which are described in the following subsections, as well as the different stages followed to obtain the results shown in section 6.

5.1. Data

Detailed studies of the characteristics relating to high frequency financial data are found in (Dacorogna, Gençay, Müller, Olsen & Pictet, 2001) (Russell & Engle, 2010). In this research, experimentation was conducted using data from 20 assets listed on the *Brazil Stock Exchange (B3)*, identified by their *tickers* or *instrument symbols*. These data were stored in two types of compressed text files. One of these refers to trades, while the other type corresponds to buy and sell orders. The data and description of each type of file were obtained from the financial market server `ftp://ftp.bmf.com.br/MarketData/` in May 2019. The period used in this experimentation starts on July 2, 2018 and ends on May 6, 2019. In total, 206 trading days. Both trades and buy and sell orders were recorded with millisecond precision. The data correspond to trading hours: from 10:00 to 16:55. Details about the Brazilian market schedule, as well as studies related to it, can be found in (Perlin & Ramos, 2016).

Transaction data contain multiple variables. Their specifications, as well as that relating to the orders data, were in files with *.txt* extension from the aforementioned server. The file with the specification of the variables contained in the trade data is called *NEG_LAYOUT*. In relation to the orders data, these files are named as: *OFER_CPA_LAYOUT* and *OFER_VDA_LAYOUT*. Additionally, more information was obtained in (B3 Brasil, Bolsa, Balcão, 2018). Table 1 lists the trades variables names used in this research.

Session Date	Instrument Symbol
Trade Price	Traded Quantity
Trade Time	

Table 1: Transaction Data Variables

High frequency financial data contain multiple errors (Dacorogna, Gençay, Müller, Olsen & Pictet, 2001) (Falkenberry, 2002) (Hautsch, 2012) and a procedure for cleaning this type of data is described (Barndorff-Nielsen, Hansen, Lunde & Shephard, 2009). In relation to these errors, (Brownlees & Gallo, 2006) established a procedure to detect and eliminate erroneous observations. Subsequently, cancelled operations and not executed orders were eliminated, and those trades with a *Trade Price* or *Traded Quantity* less than or equal to zero were rejected (Hautsch, 2012). In addition, the variables *Trade Price* and *Traded Quantity* were transformed, due to the existence of simultaneous orders with the same *Trade Price*. Table 2 shows the extent of this incidence.

The number of observations varies depending on the ticker, with figures between just over 1 million observations until figures reaching around 5 million observations. The columns *PSO* and *PSODP* refer to the percentage of simultaneous operations and the percentage of simultaneous operations with different prices, respectively. In the former, there are percentages that exceed 60% of *PSO*, while in the latter a ticker exceeds 8% of *PSODP*, followed by figures below 2.5% in most of the assets. In order to solve this problem, the work of (Brownlees & Gallo, 2006) was used as

Ticker	Trades	POS (%)	PSODP (%)
B3SA3	4621343	51.31	1.69
BBAS3	5097897	55.63	2.42
BBDC3	2014462	53.29	1.87
BBSE3	3038689	63.04	1.12
BRFS3	3425237	53.48	1.84
BRML3	3240840	53.88	0.45
CIEL3	4477212	60.94	0.65
CSNA3	2734771	55.46	0.73
GGBR4	3817163	50.72	1.00
GOAU4	2593701	38.67	0.35
HYPE3	1655824	46.09	1.73
JBSS3	4014349	58.22	0.50
KROT3	4023300	55.04	0.56
LAME4	2828343	48.91	1.20
MGLU3	1159513	58.39	8.53
PETR3	3165573	44.01	1.80
RAIL3	3595665	55.71	0.79
RENT3	3370796	58.86	1.26
TIMP3	1975100	45.22	0.62
USIM5	3078855	48.63	0.50

Table 2: Raw data. Simultaneous transactions

a reference, considering a weighted average and performing the following transformations. Let y_i be the prices, v_i the amounts traded and $i = 1, 2, ..., k$ the number of equal timestamps. Let be a timestamp x with several simultaneous trades. Then, for a timestamp x, we have

$$y_x = \frac{\Sigma_{i=1}^{k} y_i v_i}{\Sigma_{i=1}^{k} v_i} \qquad v_x = \Sigma_{i=1}^{k} v_i \tag{21}$$

The result is a volume-weighted average *Trade Price* and a total sum of the *Traded Quantity* for every group of simultaneous trades. In order to lose as little information as possible, it was created a variable with the number of simultaneous transactions in every group, as shown in (Brownlees & Gallo, 2006), and what has been called *Transactions*. Finally, the variables selected for the segmentation part were the following: *Trade Time*, *Trade Price*, *Traded Quantity* and *Transactions*.

It was used *tick data* or *tick time* with the cited tickers. The description can be found in (Griffin & Oomen, 2008). Raw high frequency data contain sequences in which the price does not change, so the associated series of returns for those sequences would be composed of zeros. If we reduce these sequences to the first value, we obtain tick data or tick time, which is the name used when there is a difference of at least one *tick* between one trade and the next one, defined as the smallest variation that can occur at the listed price (Hautsch, 2012). The tick time data used can represent a reduction of around 75% with respect to the raw data, according to the figures for the number of trades in tables 2 and 3.

Tick data was obtained from the previously processed tickers. All sequences with constant *Trade Price* were reduced to the first value, but the variables *Transaction* and *Traded Quantity* are the sum of their values in each sequence.

An outlier treatment was also conducted, but under a very conservative criterion. The purpose was to reduce the loss of information and maintain the essence of the dynamics of high-frequency financial data. The adaptive filter suggested in (Brownlees & Gallo, 2006) was used to detect inconsistencies in the data. Let be a time series with observations $i = 1, 2, ..., n$ and prices y_i, then we consider that

$$|y_i - \overline{y}_i(k)| < 3s_i(k) + \gamma \tag{22}$$

The terms $\overline{y}_i(k)$ and $s_i(k)$ are referred to the *δ-trimmed sample mean* and the *sample standard deviation* from a neighborhood of k observations around i. The γ parameter is called *granularity*. With this filter, the false observation is removed, while it allows the true observation i to remain. The *neighborhood* corresponding to the first observation of each daily series contains the first k observations, the neighborhood related to the last observation is composed of the k last transactions and for the observation in the middle of the series, it was obtained the $k/2$ previous transactions and the $k/2$ subsequent observations, and so on. In this way, the reference to decide the validity of an observation is the closest valid observations. It was selected a percentage of *trimming* δ equal to 0.1 and a window length k equal to 30.

The granularity avoids the zero variance provoked when sequences of k equal prices are produced. Following (Brownlees & Gallo, 2006), γ should be chosen as a multiple of the minimum price change for each asset. According to this premise, it was chosen γ as the mean of the absolute values corresponding to 5% and 95% quantiles computed over the price differences series.

Ticker	Trades	Ticker	Trades
B3SA3	1244902	HYPE3	530391
BBAS3	1341834	JBSS3	765707
BBDC3	582892	KROT3	820254
BBSE3	621448	LAME4	762381
BRFS3	874510	MGLU3	463454
BRML3	622923	PETR3	859144
CIEL3	785091	RAIL3	731211
CSNA3	511758	RENT3	836196
GGBR4	885488	TIMP3	472230
GOAU4	574962	USIM5	633536

Table 3: Clean tick data

5.2. Segmentation

Trades time series are recorded by days, so we have a first partition of these series. Therefore, it was executed the segmentation on each day, independently. The execution time will depend on the number of daily observations, so it is interesting to know how these observations are distributed in each ticker. In table 4, it is shown descriptive statistical summaries of the variable *number of daily observations*, where the maximums range from figures close to 7500 to figures above 25000.

Ticker	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
B3SA3	2516	4309.25	5420.0	6043.06	6980.50	25390
BBAS3	1959	4703.00	5796.0	6513.69	7442.00	23324
BBDC3	1108	2012.50	2504.5	2829.25	3152.25	14618
BBSE3	1120	2011.00	2806.5	3016.51	3576.25	9704
BRFS3	1476	2819.25	3686.5	4245.11	5122.00	13638
BRML3	837	1960.00	2679.0	3023.71	3808.00	8512
CIEL3	1189	2539.75	3298.5	3811.05	4660.75	9530
CSNA3	561	1337.00	1988.0	2484.25	3176.50	12072
GGBR4	1438	3039.25	3855.0	4298.47	5113.00	13937
GOAU4	702	1665.50	2488.0	2791.07	3396.75	12936
HYPE3	735	1819.25	2259.5	2574.40	3194.25	7441
JBSS3	928	2279.75	3019.5	3716.94	4568.00	14586
KROT3	1198	2749.00	3664.0	3981.73	4805.50	10383
LAME4	966	2392.75	3099.0	3700.68	4512.00	11932
MGLU3	830	1635.75	2126.5	2249.72	2689.00	7913
PETR3	1473	2774.75	3674.0	4170.52	4822.00	14984
RAIL3	1015	2457.50	3154.0	3549.36	4239.50	10047
RENT3	1317	3128.50	3844.0	4058.91	4802.25	11003
TIMP3	667	1476.75	2012.5	2292.28	2866.75	9774
USIM5	786	1882.25	2605.5	3075.41	3718.75	11204

Table 4: Daily trades summary

Once the procedure to obtain the clean tickers was performed, the data was aggregated to a 1-minute period and the segmentation was executed with the optimal method. The resulting breakpoints were moved to the original data, separating each day into different time series fragments. On these fragments, a second segmentation was executed with the optimal method, obtaining the daily segments of each ticker. As mentioned above, each segment corresponds to an intraday trend, and the number of resulting daily segments is between 21 and 681 segments, depending on each ticker.

5.3. Limit order book reconstruction

In order to extract features from buy and sell orders, you first have to rebuild the order book and obtain its status at certain times. In (Gould, Porter, Williams, McDonald, Fenn & Howison, 2013), they reviewed literature related to the analysis and modeling of limit order books, in addition to describing the so-called *stylized facts* of financial markets, defined as non-trivial statistical regularities present in data from such markets. In (Christensen & Woodmansey, 2013), they defined the reconstruction of the limit order book as the process by which we start from the recorded data and regenerate it to obtain the multidimensional limit order book.

The *limit order book* is reconstructed by combining and processing the data related to the buy and sell orders files corresponding to the main market. The orders data variables used in this research are listed in table 5, and are described in the corresponding text files provided by the aforementioned market.

Session Date	Order Price
Instrument Symbol	Total Quantity Order
Order Side	Traded Quantity Order
Sequential Order Number	Secondary Order ID
Order Datetime Entry	Order Status

Table 5: Orders Variables

In figure 8, it is shown boxplots that represent summaries relative to the number of orders per day and ticker, which are variable. Orders generated in one day are added to those that remain active from previous periods. The asset with ticker BBDC3 has a day maximum number of orders exceeding 700000, with a daily average orders slightly higher than 250000, which reflects the computational effort required to obtain the order book states for the times of 582892 trades that has the BBDC3 trades series, as indicated in table 3. The rest of the tickers present lower figures, but the computational effort to process these tickers, although lower than in the BBDC3 ticker, is still considerably high.

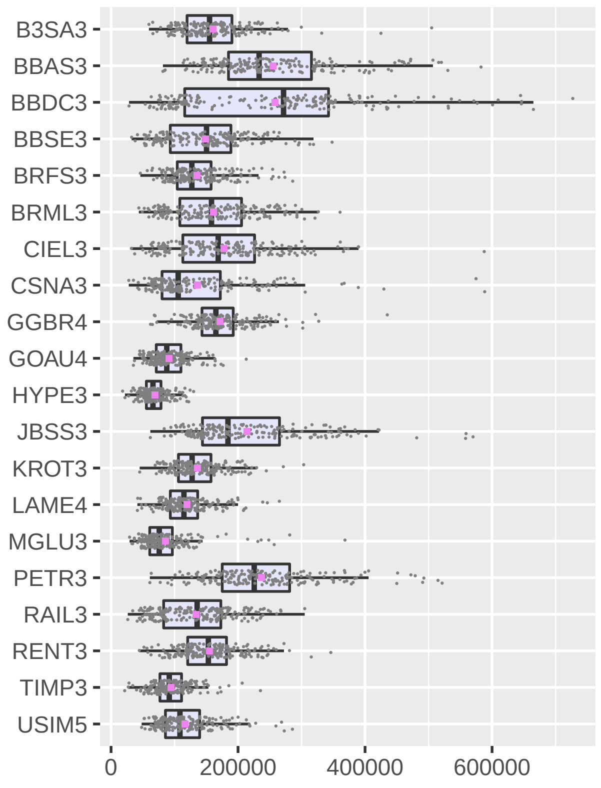


Figure 8: Raw data. Limit orders by day

It was preprocessed buy and sell orders data and eliminated those observations with errors, such as negative or null *Total Quantity Order* and *Order Price* , in addition to those for which the *spread* was negative, as raised in (Barndorff-Nielsen, Hansen, Lunde & Shephard, 2009).

All orders have a lifecycle that can go through different states. For the purposes of this research, it has represented the lifecycle of an order in figure 9, taking into account the

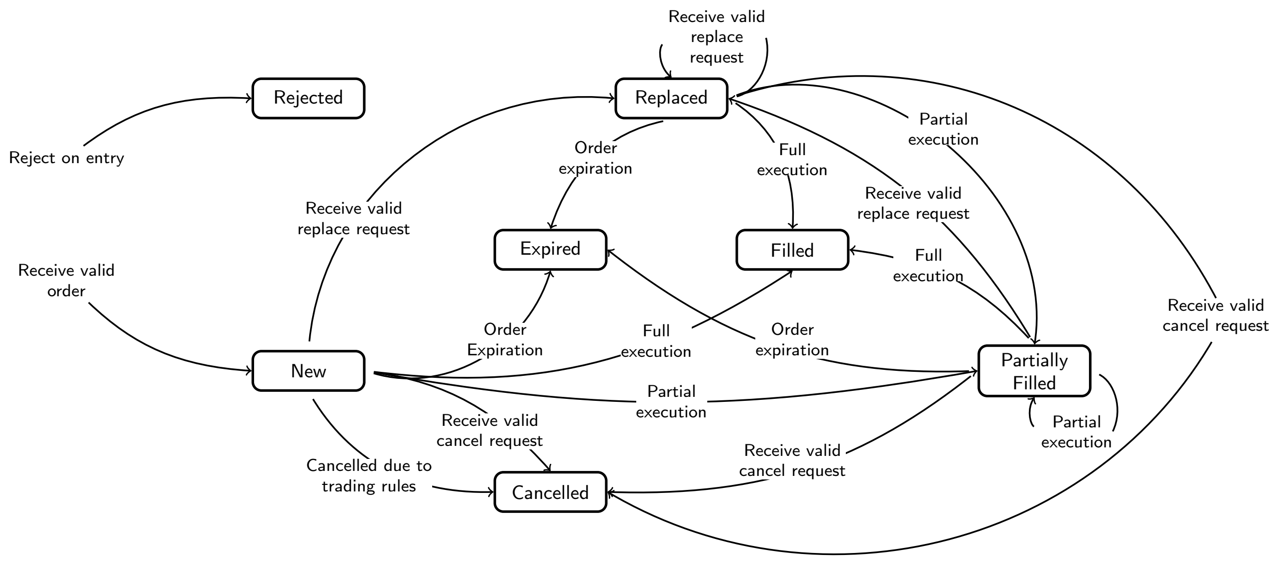


Figure 9: Order Lifecycle. *Source:* (*B3 Brasil, Bolsa, Balcão, 2018*)

cycle shown in (B3 Brasil, Bolsa, Balcão, 2018). The graph shows all possible combinations between different states. There are 7 *Order Status*: *New*, *Partially Filled*, *Filled*, *Cancelled*, *Replaced*, *Rejected* and *Expired*. The final states can be: *Filled*, *Cancelled*, *Replaced*, *Rejected* and *Expired*.

In figure 9, the final states are arrow receivers, never emitters. The origin of an order is registered with *Order Status New*. If said order is rejected at origin, it would go to the final state *Rejected*. Otherwise, the original order could be modified, cancelled, partially or totally crossed or expired. The modified order could go into the same states as an original order, and it could also be modified again. When an order is partially executed, this execution could be repeated, but it could also go into the same states as a *New* or *Replaced* order.

Each order with *New* status is linked to other states through the *Sequential Order Number*, which allows knowing if the initial order has moved to different states. If we were analyzing the LOB at a given time and there were several orders with the same *Sequential Order Number*, linked to an order with *New* status, the active order at the time would be the one with the most recent timestamp, and if there were several simultaneous orders, the active order would be the one with the highest *Secondary Order ID*. The LOB status at time x includes the orders currently active. These orders would be those with the state *New*, *Replaced* or *Partially Filled*, which were not previously crossed and were not cancelled, rejected or expired. From the active orders at each time of the trades series, the buy and sell prices and volumes are obtained for each level of the order book. Finally, the features are extracted from said prices and volumes for each segment.

5.4. Variable selection

The features extracted in the intervals from the high frequency segmentation were used to estimate the future behavior of three response variables. One of these is the duration corresponding to the time intervals resulting from the completed segmentation, which it was measured directly, from the difference between the final and initial timestamp of each interval. Regarding the other two response variables, proxies were used in order to estimate the volatility and direction associated with trend movements in future time intervals. These variables are part of the set of features extracted from each segment, to which it was added some of those collected in the reviewed works, as well as others that could be explanatory of the variance in the response variables.

High frequency scientific literature collects evidence about the influence of certain variables on price behavior. Proof of this is what is stated in (Cont, Kukanov & Stoikov, 2013), where they suggested that changes in prices, during short periods of time, are mainly driven by the *order flow imbalance*, defined as the imbalance that occurs between supply and demand for the prices from the best buy and sell orders. Also in (Xu, Gould & Howison, 2018), they studied the *multi-level order flow imbalance*. In (Cao, Hansch & Wang, 2009), they considered the imbalance in the order book length as a function of the difference in aggregated quantities of shares on the buy and sell sides, divided by the sum of these. Similarly, those features that reflect an imbalance in the order book by level grouping were extracted in this research.

Table 6 contains a proposal of features extracted in each interval. These features were divided into two blocks, depending on their origin: features from the trades series and features from the limit order book. In the next subsection,

Trades	
	Average price
	Price variance
	Fitted price variance
	Residual variance
	Mean absolute error
	Interval duration
	Average trade duration
	Trade duration variance
	Interval observations
	Interval transactions
	Interval transaction variance
	Value per second
	Return per second
	Volatility per unit of time
	Squared log-return per second
	Total squared log-returns per second
	Squared log-return per second variance
LOB	
	Average buy market
	Average sell market
	Average buy volume
	Average sell volume
	Average five levels OBI
	Average ten levels OBI
	Average total levels OBI

Table 6: Feature selection

the three response variables are described and the Appendix contains the corresponding description of the remaining features.

5.5. Response variables

The features extracted from the high frequency time intervals are the regressors used to estimate the volatility, duration and direction of future trend movements. In the next three subsections, the variables employed to effect the labeling corresponding to these response variables are described.

5.5.1. *Volatility per unit of time*

Volatility is an unobserved variable, so proxies are used for its study. Taking into account the volatility adjusted for time duration between transactions present in (Engle, 2000), it was formulated the following proxy variable to estimate the volatility associated with each trend movement contained in a segment j with observations i and number of observations e_j,

$$\sigma_j^2 = \frac{1}{e_j - 2} \sum_{i_{j-1}+1}^{i_j-1} \left(\frac{r_i}{\sqrt{d_i}} - \mu_{r_x} \right)^2 \tag{23}$$

The variable r_i represents the logarithmic returns between two consecutive trades, d_i is the duration between these trades and μ_{r_x} refers to the mean of *log-returns per square root duration* in each segment j with observations i,

$$d_i = x_{i+1} - x_i \tag{24}$$

$$r_i = log(y_{i+1}) - log(y_i) \tag{25}$$

$$\mu_{r_x} = \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j-1} \frac{r_i}{\sqrt{d_i}} \tag{26}$$

In figure 10, it is shown an example of the evolution of volatility per unit of time through all the intervals in which the quotes series were segmented for a given day.

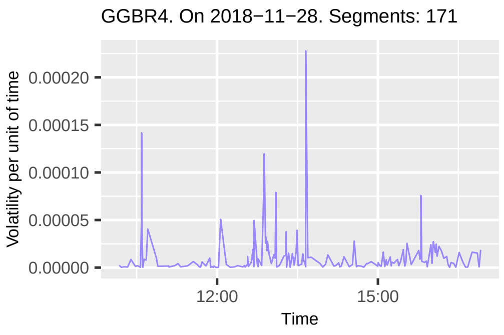


Figure 10: Volatility evolution

5.5.2. *Interval duration*

Total duration in seconds of each interval associated with segment j,

$$D_j = x_{i_j} - x_{i_{j-1}+1} \tag{27}$$

The elements x_{i_j} and $x_{i_{j-1}+1}$ are the final and initial timestamps of a given interval, respectively. Figure 11 contains the graphical representation corresponding to an example of the time evolution of the total duration on a specific day.

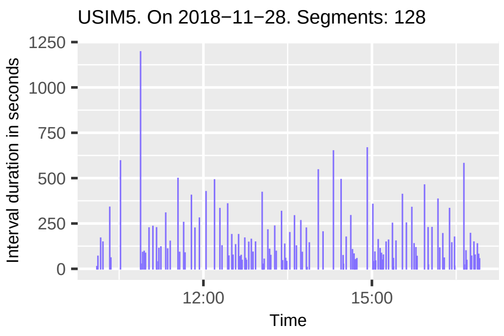


Figure 11: Interval duration evolution

5.5.3. *Return per second*

It was formulated a proxy variable for the intraday trend direction in each time interval achieved with the developed segmentation method, which is a function of the total return generated in each interval. But the trades series used in the experiments are unevenly spaced, so it could be that two intervals had the same return and the durations were different. To differentiate this aspect, it was considered the return in each interval adjusted for its duration. That is, the arithmetic return per second in each segment j, with y_{i_j} and $y_{i_{j-1}+1}$ the final and initial price in each interval, respectively

$$R_j = \frac{\frac{y_{i_j} - y_{i_{j-1}+1}}{y_{i_{j-1}+1}}}{x_{i_j} - x_{i_{j-1}+1}} \tag{28}$$

In this expression, the divisor corresponds to the *interval duration* D_j, and the elements x_{i_j} and $x_{i_{j-1}+1}$ represent the final and initial time in each interval, respectively. In figure 12, it is shown an example of the series of returns per second relative to all the intervals coming from the partition of a specific day.

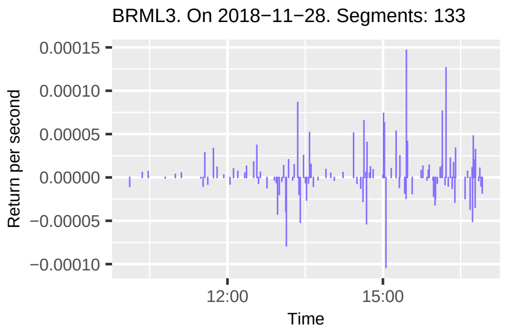


Figure 12: Return per second evolution

5.6. Machine learning

The segmented trades series are formed by the trades variables, to which the limit order book variables were added, obtained in each timestamp from the trades series. These variables form the data set corresponding to each ticker, from which the features in each segment were extracted. These sets were separated into subsets train and test, in the proportion *70/30*, so that the subset train was composed of the segments relative to the first 144 days from each ticker, while the subset test corresponded to the segments present in the last 62 days. It was executed the split of the initial sets into train and test subsets respecting the time sequence, so that the observations from the test subset occurred temporarily after the observations from the train subset. The next step was to extract features in each interval from the aforementioned subsets, and calculate the quantiles with probabilities 0.3333 and 0.6666 relative to the response variables, accomplished from the values of these variables in the train subset. The resulting values divided the three labeling regions of the response variables. Next, it was constructed the samples to prepare the embedding for the machine learning method used. These samples are composed of an ordered sequence with the features from k segments, where each block is a timestep. In the conducted experiments, a value for k equal to 8 was selected, which provided a slight improvement in the results than those obtained with values of 3, 5, 10, 20, 50 and 100.

The last term in each sample is the label of the corresponding response variable, in such a way that each sample has part in X and part in Y, forming the divisions: *trainX*, *trainY*, *testX* and *testY*. The samples were consecutively built by days. In table 7, it is shown the number of segments and samples from each subset for every ticker. It is noted that the developed segmentation method provides a sufficient number of samples to reliably train the artificial intelligence models, with these data. Furthermore, the number of test samples is a sufficient size to achieve a reliable validation of the models.

	Segments		Samples	
Ticker	Train	Test	Train	Test
B3SA3	29081	13503	27929	7426
BBAS3	37140	18858	35988	12263
BBDC3	22960	12003	21808	6000
BBSE3	20473	9332	19321	8536
BRFS3	21450	11202	20298	10229
BRML3	13693	6089	12541	3176
CIEL3	16636	6576	15484	5036
CSNA3	10439	9088	9287	5084
GGBR4	20429	7229	19277	5120
GOAU4	9969	2929	8817	2003
HYPE3	17521	8230	16369	5521
JBSS3	13837	9629	12685	7363
KROT3	16604	7278	15452	5217
LAME4	19108	7953	17956	4668
MGLU3	20521	7450	19369	5059
PETR3	24845	10332	23693	4533
RAIL3	17437	8756	16285	4408
RENT3	26343	13293	25191	9744
TIMP3	10912	4083	9760	2659
USIM5	12948	5023	11796	3602

Table 7: Segments and samples

The *trainX* and *trainY* divisions from the samples were used to train artificial intelligence models. Next, the design characteristics of these models and their respective hyperparameters are attached.

5.6.1. *XGBoost*

It was built a model for every response variable in each ticker. The parameters selected are classified into three blocks: general parameters, *booster* parameters and learning parameters. The first block refers to the type of *boosting* used, which was of type tree. Regarding the second group, it was chosen values for the parameters relative to the learning rate η equal to 0.001, the minimum loss reduction γ to perform an additional partition in a tree node equal to 3, the maximum tree depth equal to 5 and the training subsampling ratio equal to 0.75. The learning parameters selected refer to the learning objective with the *softprob* function for multiclass classification, and to the evaluation metric for data validation, called error rate for multiclass classifications and equal to the proportion of failed cases between total cases. Finally, it was set the values of two parameters related to computational execution: the maximum number of boosting iterations equal to 1000 and the number of rounds to stop training, if performance does not improve, equal to 50.

The input of the models contained the features listed in table 6, repeated a specific number of timesteps. The features importance in these models was calculated considering each variable from the timesteps sequence, for which these variables were grouped to compute the total contribution of each selected feature.

The usual procedure for making forecasts with a classification model and evaluating its result is to train the model on a train subset, then we make forecasts with a new input *testX* and, finally, we compare the output with the response variable *testY*. This comparison is evaluated with the confusion matrix and its associated metrics, whose result provides us with the model validation.

5.7. Computing resources

The code related to this experimental research was developed in *R* language (R Core Team, 2020). The *strucchange* package (Zeileis, Leisch, Hornik & Kleiber, 2002) was used to segment time series. The packages *xgboost* (Chen, He, Benesty, Khotilovich, Tang, Cho, Chen, Mitchell, Cano, Zhou, Li, Xie, Lin, Geng & Li, 2020) and *caret* (Kuhn, 2020) were used to conduct the experiments with artificial intelligence. The tidyverse package collection (Wickham, Averick, Bryan, Chang, McGowan, François, Grolemund, Hayes, Henry, Hester, Kuhn, Pedersen, Miller, Bache, Müller, Ooms, Robinson, Seidel, Spinu, Takahashi, Vaughan, Wilke, Woo & Yutani, 2019), packages *plyr* (Wickham, 2011) and *data.table* (Dowle & Srinivasan, 2020), as well as packages *quantmod* (Ryan & Ulrich, 2020), *zoo* (Zeileis & Grothendieck, 2005), *tidyquant* (Dancho & Vaughan, 2020a), *timetk* (Dancho & Vaughan, 2020b) and *tibbletime* (Vaughan & Dancho, 2020) were used to analyze and explore the data. The last five, related to quantitative financial modeling and analysis and the specific treatment of time series. The *PerformanceAnalytics* package (Peterson & Carl, 2020) was used for performance and risk analysis issues. The packages *lubridate* (Grolemund & Wickham, 2011) and *hms* (Müller, 2020) were used for aspects related to date and time formats. The packages *DBI* (R Special Interest Group on Databases (R-SIG-DB), Wickham & Müller, 2019) and *dbplyr* (Wickham & Ruiz, 2020) were used for the construction, connection and database management. The statistical tests were performed using the package *scmamp* (Calvo & Santafé, 2016).

The experiments whose results are presented in this scientific article were conducted using the resources provided by the *Centro de Supercomputación y Visualización de Madrid (CeSViMa)*, at the *Universidad Politécnica de Madrid*. The jobs were run on *Magerit-3*, a supercomputer consisting of 68 nodes *Lenovo ThinkSystem SD530* , each of which has 2 processors *20-core Intel Xeon Gold 6230 at @2.10 GHz (1,344 GFLOPS)*, *192 GB RAM* and intraday *480 GB SSD*. The nodes are interconnected by two *25 Gbps low-latency networks*, one of them with a *flat-tree* architecture dedicated exclusively to the messages passage.

6. Results and discussion

The values of the performance metrics obtained from the forecasts made with each *machine learning* model built for forecasting the variables volatility, duration and direction, are presented in tables 8, 9 and 10, respectively. The results are sorted by *tickers* and performance metric: kappa and F1-score for each of the three classes. Additionally, the values that delimit each class region are shown in the columns named *Q33* and *Q66*. The best results are highlighted in bold for each measure.

	Performance metrics					
		F1-score			Delimiters	
Ticker	kappa	high	med	low	Q33	Q66
B3SA3	0.25	0.50	0.28	0.67	1.45e-06	6.850e-06
BBAS3	0.24	0.52	0.33	0.62	6.30e-07	2.980e-06
BBDC3	0.19	0.35	0.45	0.55	2.20e-07	3.010e-06
BBSE3	0.18	0.46	0.29	0.56	8.60e-07	6.060e-06
BRFS3	0.25	0.56	0.47	0.49	1.87e-06	1.070e-05
BRML3	0.25	0.52	0.47	0.54	1.94e-06	1.645e-05
CIEL3	0.22	**0.65**	0.39	0.37	3.42e-06	2.185e-05
CSNA3	0.16	0.56	0.35	0.38	9.90e-07	8.090e-06
GGBR4	0.24	0.47	0.38	0.60	1.66e-06	1.018e-05
GOAU4	0.22	0.41	0.48	0.57	2.98e-06	2.610e-05
HYPE3	0.21	0.47	0.36	0.58	7.40e-07	5.630e-06
JBSS3	0.24	0.51	**0.50**	0.54	2.39e-06	1.677e-05
KROT3	0.30	0.61	0.47	0.52	2.35e-06	1.814e-05
LAME4	0.24	0.48	0.44	0.58	1.23e-06	9.640e-06
MGLU3	0.19	0.48	0.22	0.61	1.76e-06	8.180e-06
PETR3	**0.33**	0.61	0.30	**0.68**	5.30e-07	2.950e-06
RAIL3	0.24	0.54	0.44	0.53	2.00e-06	1.343e-05
RENT3	0.17	0.40	0.36	0.57	1.90e-06	8.280e-06
TIMP3	0.28	0.54	0.47	0.55	1.30e-06	1.028e-05
USIM5	0.25	0.47	**0.50**	0.54	1.74e-06	1.746e-05

Table 8: Volatility results. Performance metrics and class regions delimiters

The best results were obtained in volatility estimation, followed by duration and direction, according to the *kappa* values from tables 8, 9 and 10. In relation to the first two variables, the best performances were achieved in the forecasting of the classes *high* and *low*. Regarding direction, the best performances were accomplished for the *medium* class, as indicated by the *F1-score* figures obtained.

In relation to the values of the quantiles with probabilities 0.3333 and 0.6666, *Q33* and *Q66*, which delimit the regions of the three classes, they refer to the volatility per unit of time, to the duration of the intraday trend in seconds and return per second, depending on the corresponding response variable.

In relation to volatility, the best result for kappa was achieved for the ticker PETR3, with a value of 0.33. Regarding the F1-score, the best value was obtained for the low class and the same ticker, equal to 0.68. For the high class, the best F1-score figure was reached with the CIEL3 ticker, while for the medium class the best figure was equal to 0.5.

Regarding duration, the best kappa value was obtained with the ticker USIM5, equal to 0.26. In relation to the F1-score, values of 0.69 and 0.77 were reached for the high and low classes and tickers GOAU4 and CSNA3, respectively. For the medium class, the best figures were equal to 0.37.

	Performance metrics				Delimiters	
		F1-score				
Ticker	kappa	high	med	low	Q33	Q66
B3SA3	0.18	0.52	0.33	0.48	50.857	121.824
BBAS3	0.20	0.49	0.33	0.56	40.395	95.004
BBDC3	0.17	0.44	0.33	0.56	61.709	143.653
BBSE3	0.17	0.52	0.34	0.46	63.140	164.148
BRFS3	0.23	0.49	0.32	0.61	63.015	164.962
BRML3	0.16	0.50	**0.37**	0.44	107.544	245.928
CIEL3	0.18	0.59	0.36	0.38	73.197	201.058
CSNA3	0.25	0.43	0.27	**0.77**	139.053	323.325
GGBR4	0.19	0.63	0.28	0.41	76.467	177.293
GOAU4	0.22	**0.69**	0.29	0.39	163.862	356.536
HYPE3	0.17	0.50	0.28	0.52	65.058	184.884
JBSS3	0.19	0.37	0.26	0.69	103.490	248.070
KROT3	0.17	0.56	0.26	0.46	86.393	206.133
LAME4	0.21	0.55	0.26	0.55	77.553	182.603
MGLU3	0.20	0.60	0.30	0.45	51.846	160.148
PETR3	0.21	0.60	0.31	0.47	60.671	138.369
RAIL3	0.17	0.44	**0.37**	0.53	82.613	198.650
RENT3	0.18	0.47	0.33	0.55	53.666	131.865
TIMP3	0.19	0.61	0.30	0.41	133.463	327.919
USIM5	**0.26**	0.63	0.32	0.52	116.463	267.837

Table 9: Duration results. Performance metrics and class regions delimiters

	Performance metrics				Delimiters	
		F1-score				
Ticker	kappa	high	med	low	Q33	Q66
B3SA3	0.14	0.34	0.55	0.38	-5.90e-06	6.01e-06
BBAS3	0.16	0.40	0.49	0.42	-6.72e-06	6.80e-06
BBDC3	0.15	0.42	0.47	0.41	-6.21e-06	6.11e-06
BBSE3	0.14	0.34	0.54	0.38	-4.47e-06	4.50e-06
BRFS3	0.15	0.38	0.52	0.38	-5.49e-06	5.34e-06
BRML3	0.11	0.28	0.58	0.31	-3.91e-06	4.19e-06
CIEL3	0.13	0.35	0.55	0.30	-5.16e-06	5.01e-06
CSNA3	0.12	0.44	0.38	**0.44**	-4.46e-06	4.61e-06
GGBR4	0.14	0.33	0.59	0.30	-5.22e-06	4.95e-06
GOAU4	0.16	0.32	**0.63**	0.30	-3.88e-06	3.36e-06
HYPE3	0.13	0.37	0.53	0.35	-4.87e-06	5.10e-06
JBSS3	0.10	**0.47**	0.36	0.37	-4.30e-06	4.66e-06
KROT3	0.14	0.34	0.57	0.28	-5.36e-06	5.62e-06
LAME4	0.14	0.36	0.54	0.34	-5.25e-06	5.25e-06
MGLU3	0.14	0.33	0.58	0.31	-6.78e-06	6.43e-06
PETR3	**0.17**	0.31	0.62	0.36	-6.61e-06	6.66e-06
RAIL3	0.14	0.38	0.52	0.36	-4.94e-06	4.87e-06
RENT3	0.13	0.40	0.47	0.39	-6.41e-06	6.58e-06
TIMP3	0.15	0.29	0.57	0.38	-3.00e-06	2.91e-06
USIM5	0.16	0.32	0.58	0.38	-4.70e-06	4.68e-06

Table 10: Direction results. Performance metrics and class regions delimiters

Finally, the results obtained for the direction variable were significantly lower than those obtained for the volatility and duration variables. The maximum value for the kappa metric was equal to 0.17, achieved with the PETR3 ticker. In the case of this variable and the measure F1-score, the best result was achieved with the medium class, reaching a value of 0.63 with the GOAU4 ticker. Regarding the other two classes, values of 0.47 and 044 were obtained for the high and low classes and the JBSS3 and CSNA3 tickers, respectively.

Figure 13 contains boxplots that represent the summaries of the importance of variables with values between 0 and 1. In each response variable, as can be seen in figure 13, there is uniformity regarding the variable importance for each *ticker*.

In figure 13, referring to the importance of the variables used as regressors to forecast volatility linked to future intraday trends, the most important variables are the volatility per unit of time and the total squared log-returns per second, followed by the average trade duration and the variance of squared log-returns per second in each interval. This is a logical result, since the first two variables and the last one are volatility proxies variables, while duration is directly related to volatility.

Regarding duration, the highest values are reached with the different duration metrics: the interval duration, the average and the variance of durations between trades in each interval. In addition, the value per second and the squared returns log-returns per second also are highlighted as important variables. The result obtained is also logical, since the interval duration variable is the most important and

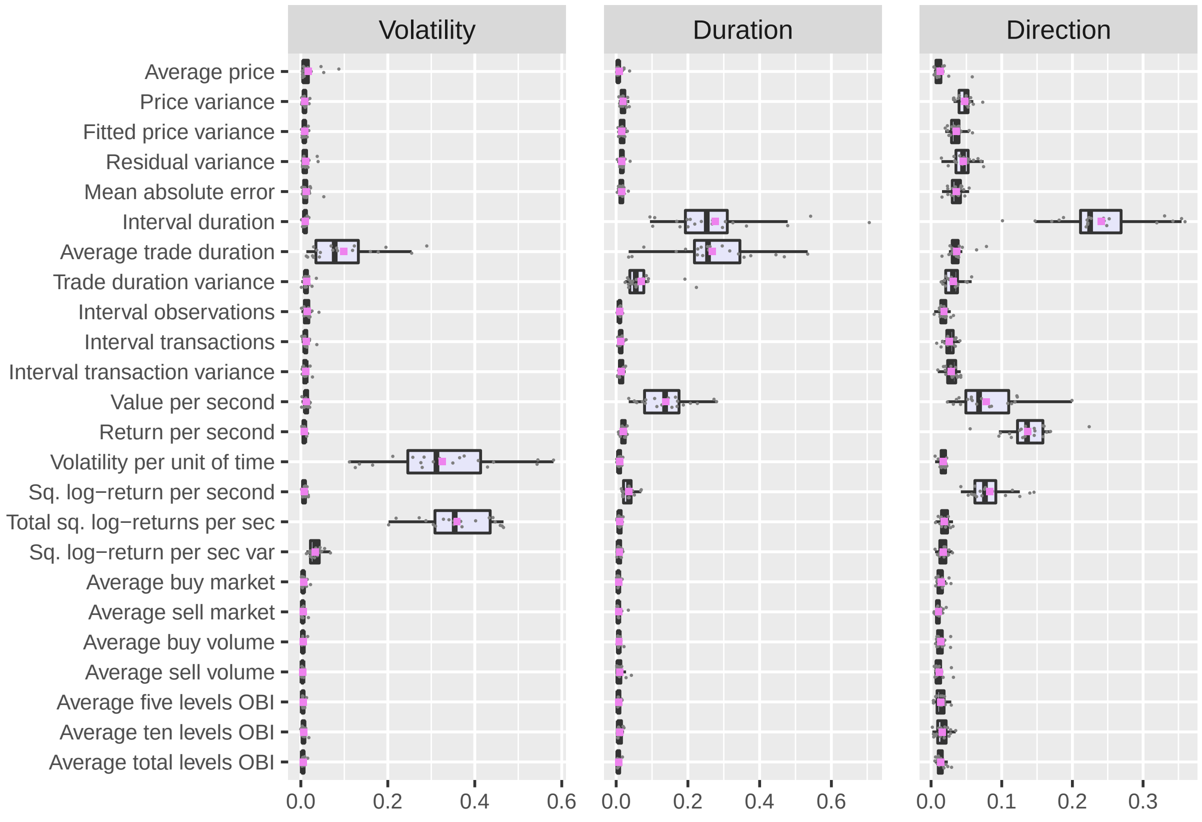


Figure 13: Variable importance

it is the response variable, and the durations between trades are very closely related to the response variable.

In relation to the direction, the figure 13 contains, as variables with greater importance, the interval duration, the return per second, the squared return per second and the value per second. In this case, it is highlighted that the response variable is not the most important, since this place is occupied by the interval duration. However, it must be considered that the return per second is a function of the interval duration, since the return of the interval is divided by said variable. It is also highlighted that in the case of direction, a large part of the features are of relative importance, which did not occur with the other two response variables.

In general terms, it is highlighted the greater importance of the variables extracted from the series of *trades*, for all the *tickers* and response variables, against the features extracted from the states of the limit order books, which show less importance. Only in the case of direction, the averages of the order book imbalance for the first 5 and 10 levels present a slightly greater importance than that shown in the other two response variables.

7. Concluding remarks

Using a time series segmentation method, it was developed a feature engineering that allows extracting features in high frequency intervals where intraday trends occur. It was extended the intervals resulting from this segmentation to order book data, in order to extract features from their states in these intervals. This methodology facilitates the analysis of variables in intraday trends and their estimation in future intervals through artificial intelligence models.

This methodology was applied to forecast three response variables. To do this, features among those highlighted by the high frequency financial literature were selected, to which it was added others that were considered of interest to explain the variation in each response variable. For the purpose of extracting features from limit orders, the LOB at each time of the corresponding trades series was reconstructed.

Artificial intelligence models with features extracted from each interval were fed, as a way to estimate the volatility, duration and direction associated with intraday price trends in future intervals. The best results were obtained for volatility forecasting, follow by duration and direction. The importance of the selected variables was obtained, and the trades variables provided a better explanation of the variation in the response variables than the LOB variables.

The feature engineering developed in this research was applied to extract particular features and forecast three defined response variables, with a certain artificial intelligence method and using specific data. However, this methodology is general-purpose for high frequency time series and could be applied in a broader scope, depending on the problem to be solved and within the method specifications.

Acknowledgment

The authors gratefully acknowledge the *Universidad Politécnica de Madrid (www.upm.es)* for providing computing resources on *Magerit Supercomputer*.

References

Arévalo, A., Nino, J., León, D., Hernandez, G., & Sandoval, J. (2018). Deep learning and wavelets for high-frequency price forecasting. In *International Conference on Computational Science* (pp. 385–399). doi:10.1007/978-3-319-93701-4_29.

B3 Brasil, Bolsa, Balcão (2018). *Entry point messaging guidelines. Version 2.9.4*. URL: http://www.b3.com.br/data/files/BD/67/C3/FF/0DBB3610DF40D936790D8AA8/EntryPointMessagingGuidelines2.9.4.pdf.

Bai, J., & Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, *18*, 1–22. doi:10.1002/jae.659.

Barndorff-Nielsen, O. E., Hansen, P. R., Lunde, A., & Shephard, N. (2009). Realized kernels in practice: trades and quotes. *The Econometrics Journal*, *12*, C1–C32. URL: http://www.jstor.org/stable/23116045.

Brownlees, C., & Gallo, G. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, *51*, 2232 – 2245. doi:10.1016/j.csda.2006.09.030.

Calvo, B., & Santafé, G. (2016). scmamp: Statistical comparison of multiple algorithms in multiple problems. *The R Journal*, *8*, 248–256. doi:10.32614/RJ-2016-017.

Cao, C., Hansch, O., & Wang, X. (2009). The information content of an open limit-order book. *Journal of Futures Markets*, *29*, 16–41. doi:doi.org/10.1002/fut.20334.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (p. 785–794). doi:10.1145/2939672.2939785.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., & Li, Y. (2020). *xgboost: Extreme Gradient Boosting*. URL: https://CRAN.R-project.org/package=xgboost.

Christensen, H., & Woodmansey, R. (2013). Prediction of hidden liquidity in the limit order book of GLOBEX futures. *The Journal of Trading*, *8(3)*, 68 – 95. doi:10.3905/jot.2013.8.3.068.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*, 37–46. doi:10.1177/001316446002000104.

Cont, R., Kukanov, A., & Stoikov, S. (2013). The price impact of order book events. *Journal of Financial Econometrics*, *12*, 47–88. doi:10.1093/jjfinec/nbt003.

Dacorogna, M. M., Gençay, R., Müller, U. A., Olsen, R. B., & Pictet, O. V. (2001). *An Introduction to High-Frequency Finance*. Academic Press. doi:10.1016/B978-0-12-279671-5.X5000-X.

Dancho, M., & Vaughan, D. (2020a). *tidyquant: Tidy Quantitative Financial Analysis*. URL: https://CRAN.R-project.org/package=tidyquant.

Dancho, M., & Vaughan, D. (2020b). *timetk: A Tool Kit for Working with Time Series in R*. URL: https://CRAN.R-project.org/package=timetk.

Dixon, M., Klabjan, D., & Bang, J. (2017). Classification-based financial markets prediction using deep neural networks. *Algorithmic Finance*, *6*, 67–77. doi:10.3233/AF-170176.

Dixon, M. F., Polson, N. G., & Sokolov, V. O. (2019). Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. *Applied Stochastic Models in Business and Industry*, *35*, 788–807. doi:10.1002/asmb.2399.

Doering, J., Fairbank, M., & Markose, S. (2017). Convolutional neural networks applied to high-frequency market microstructure forecasting. In *2017 9th Computer Science and Electronic Engineering (CEEC)* (pp. 31–36). doi:10.1109/CEEC.2017.8101595.

Dowle, M., & Srinivasan, A. (2020). *data.table: Extension of data.frame*. URL: https://CRAN.R-project.org/package=data.table.

Engle, R., & Patton, A. (2001). What good is a volatility model? *Quantitative Finance*, *1*, 237–245. doi:10.1088/1469-7688/1/2/305.

Engle, R. F. (2000). The econometrics of ultra-high-frequency data. *Econometrica*, *68*, 1–22. URL: http://www.jstor.org/stable/2999473.

Falkenberry, T. N. (2002). High frequency data filtering. *Technical Report. Tick Data, Inc*, . URL: https://s3-us-west-2.amazonaws.com/tick-data-s3/pdf/Tick_Data_Filtering_White_Paper.pdf.

Felker, T., Mazalov, V., & Watt, S. M. (2014). Distance-based high-frequency trading. *Procedia Computer Science*, *29*, 2055 – 2064. doi:10.1016/j.procs.2014.05.189.

Fletcher, T., & Shawe-Taylor, J. (2013). Multiple kernel learning with fisher kernels for high frequency currency prediction. *Computational Economics*, *42*, 217–240. doi:10.1007/s10614-012-9317-z.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*, 1189–1232. URL: http://www.jstor.org/stable/2699986.

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance*, *13*, 1709–1742. doi:10.1080/14697688.2013.803148.

Griffin, J. E., & Oomen, R. C. A. (2008). Sampling returns for realized variance calculations: Tick time or transaction time? *Econometric Reviews*, *27*, 230–253. doi:10.1080/07474930701873341.

Grolemund, G., & Wickham, H. (2011). Dates and times made easy with lubridate. *Journal of Statistical Software*, *40*, 1–25. URL: http://www.jstatsoft.org/v40/i03/.

Guo, T., Bifet, A., & Antulov-Fantulin, N. (2018). Bitcoin volatility forecasting with a glimpse into buy and sell orders. In *2018 IEEE International Conference on Data Mining (ICDM)* (pp. 989–994). doi:10.1109/ICDM.2018.00123.

Hautsch, N. (2012). *Econometrics of Financial High-Frequency Data*. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-21925-2.

Keogh, E., Chu, S., Hart, D., & Pazzani, M. (2004). Segmenting time series: a survey and novel approach. In *Data Mining in Time Series Databases* (pp. 1–21). doi:10.1142/9789812565402_0001.

Kercheval, A. N., & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, *15*, 1315–1329. doi:10.1080/14697688.2015.1032546.

Kuhn, M. (2020). *caret: Classification and Regression Training*. URL: https://CRAN.R-project.org/package=caret.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer, New York. doi:10.1007/978-1-4614-6849-3.

Liu, Y. (2019). Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. *Expert Systems with Applications*, *132*, 99 – 109. doi:10.1016/j.eswa.2019.04.038.

Lovrić, M., Milanović, M., & Stamenković, M. (2014). Algorithmic methods for segmentation of time series: an overview. *Journal of Contemporary Economic and Business Issues*, *1*, 31–53. URL: https://journals.ukim.mk/index.php/jeccf/article/view/124.

Müller, K. (2020). *hms: Pretty Time of Day*. URL: https://CRAN.R-project.org/package=hms.

Nousi, P., Tsantekidis, A., Passalis, N., Ntakaris, A., Kanniainen, J., Tefas, A., Gabbouj, M., & Iosifidis, A. (2019). Machine learning for forecasting mid-price movements using limit order book data. *IEEE Access*, *7*, 64722–64736. doi:10.1109/ACCESS.2019.2916793.

Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2018). Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*, *37*, 852–866. doi:10.1002/for.2543.

Ntakaris, A., Mirone, G., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). Feature engineering for mid-price prediction with deep learning. *IEEE Access*, *7*, 82390–82412. doi:10.1109/ACCESS.2019.2924353.

Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). Deep temporal logistic bag-of-features for forecasting high frequency limit order book time series. In *ICASSP 2019 - 2019 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 7545–7549). doi:10.1109/ICASSP.2019.8682297.

Peng, Y., Albuquerque, P. H. M., Camboim de Sá, J. M., Padula, A. J. A., & Montenegro, M. R. (2018). The best of two worlds: Forecasting high frequency volatility for cryptocurrencies and traditional currencies with support vector regression. *Expert Systems with Applications*, *97*, 177 – 192. doi:10.1016/j.eswa.2017.12.004.

Perlin, M., & Ramos, H. (2016). Gethfdata: A R package for downloading and aggregating high frequency trading data from bovespa. *SSRN Electronic Journal*, . doi:10.2139/ssrn.2824058.

Peterson, B. G., & Carl, P. (2020). *PerformanceAnalytics: Econometric Tools for Performance and Risk Analysis*. URL: https://CRAN.R-project.org/package=PerformanceAnalytics.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. URL: https://www.R-project.org/.

R Special Interest Group on Databases (R-SIG-DB), Wickham, H., & Müller, K. (2019). *DBI: R Database Interface*. URL: https://CRAN.R-project.org/package=DBI.

Ramos-Pérez, E., Alonso-González, P. J., & Núñez-Velázquez, J. J. (2019). Forecasting volatility with a stacked model based on a hybridized artificial neural network. *Expert Systems with Applications*, *129*, 1 – 9. doi:10.1016/j.eswa.2019.03.046.

Russell, J. R., & Engle, R. F. (2010). Analysis of high-frequency data. In Y. Aït-Sahalia, & L. P. Hansen (Eds.), *Handbook of Financial Econometrics: Tools and Techniques* (pp. 383–426). North-Holland. doi:https://doi.org/10.1016/B978-0-444-50897-3.50010-9.

Ryan, J. A., & Ulrich, J. M. (2020). *quantmod: Quantitative Financial Modelling Framework*. URL: https://CRAN.R-project.org/package=quantmod.

Si, Y.-W., & Yin, J. (2013). Obst-based segmentation approach to financial time series. *Engineering Applications of Artificial Intelligence*, *26*, 2581–2596. URL: https://www.sciencedirect.com/science/article/pii/S0952197613001723. doi:https://doi.org/10.1016/j.engappai.2013.08.015.

Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, *19*, 1449–1459. doi:10.1080/14697688.2019.1622295.

Tran, D. T., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Tensor representation in high-frequency financial data for price change prediction. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1–7). doi:10.1109/SSCI.2017.8280812.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017a). Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)* (pp. 7–12). doi:10.1109/CBI.2017.23.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017b). Using deep learning to detect price change indications in financial markets. In *2017 25th European Signal Processing Conference (EUSIPCO)* (pp. 2511–2515). doi:10.23919/EUSIPCO.2017.8081663.

Vaughan, D., & Dancho, M. (2020). *tibbletime: Time Aware Tibbles*. URL: https://CRAN.R-project.org/package=tibbletime.

Wickham, H. (2011). The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, *40*, 1–29. URL: http://www.jstatsoft.org/v40/i01/.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., & Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*, 1686. doi:10.21105/joss.01686.

Wickham, H., & Ruiz, E. (2020). *dbplyr: A dplyr Back End for Databases*. URL: https://CRAN.R-project.org/package=dbplyr.

Xu, K., Gould, M. D., & Howison, S. D. (2018). Multi-level order-flow imbalance in a limit order book. *Market Microstructure and Liquidity*, *04*, 1950011. doi:10.1142/S2382626619500114.

Zeileis, A., & Grothendieck, G. (2005). zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, *14*, 1–27. doi:10.18637/jss.v014.i06.

Zeileis, A., Kleiber, C., Krämer, W., & Hornik, K. (2003). Testing and dating of structural changes in practice. *Computational Statistics & Data Analysis*, *44*, 109 – 123. doi:10.1016/S0167-9473(03)00030-6.

Zeileis, A., Leisch, F., Hornik, K., & Kleiber, C. (2002). strucchange: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, *7*, 1–38. URL: http://www.jstatsoft.org/v07/i02/.

Zhang, Z., Zohren, S., & Roberts, S. (2019). Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, *67*, 3001–3012. doi:10.1109/TSP.2019.2907260.

Appendix. Features description

For a segment j with observations i, we have the following features.

1. Average price
   Arithmetic mean of prices y_i in each segment.

$$\bar{y}_j = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} y_i \tag{29}$$

2. Price variance
   Variance of prices y_i in each segment.

$$\sigma^2_{y_j} = \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j} \left(y_i - \bar{y}_j\right)^2 \tag{30}$$

3. Fitted price variance
   Variance of fitted prices $\hat{y}_i$ from the linear regression in each segment.

$$\sigma^2_{\hat{y}_j} = \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j} \left(\hat{y}_i - \bar{y}_j\right)^2 \tag{31}$$

4. Residual variance
   Variance of residuals ϵ_i from the linear fit in each segment.

$$\sigma^2_{\epsilon_j} = \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j} {\epsilon_i}^2 \tag{32}$$

5. Mean absolute error
   Arithmetic mean of absolute residuals ϵ_i in each segment.

$$mae_j = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} |\epsilon_i| \tag{33}$$

6. Average trade duration
   Arithmetic mean of trade durations d_i in seconds for each segment.

$$\bar{d}_j = \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j-1} d_i \tag{34}$$

$$d_i = x_{i+1} - x_i \tag{35}$$

7. Trade duration variance
   Variance of trades durations d_i in each segment.

$$\sigma^2_{d_j} = \frac{1}{e_j - 2} \sum_{i_{j-1}+1}^{i_j-1} \left(d_i - \bar{d}_j\right)^2 \tag{36}$$

8. Interval observations
   Number of observations e_j in each segment j.

$$e_j = i_j - i_{j-1} \tag{37}$$

9. Interval transactions
   The simultaneous observations corresponding to some prices were registered as a new variable when the data cleaning procedure was executed, so the number of transactions in each segment is greater than or equal to the number of observations in it, since there may be segments without simultaneous observations. This variable corresponds to the sum of the transactions executed in each time interval.

$$T_j = \sum_{i_{j-1}+1}^{i_j} T_i \tag{38}$$

10. Interval transaction variance
    We can have one or more transactions for each observation in a segment. The variance of transactions in each segment is calculated as follows.

$$\sigma^2_{T_j} = \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j} \left(T_i - \bar{T}_j\right)^2 \tag{39}$$

11. Value per second
    Total value $v_i y_i$ of trades in each segment j per second, where the divisor represents the *interval duration* D_j.

$$V_j = \frac{1}{x_{i_j} - x_{i_{j-1}+1}} \sum_{i_{j-1}+1}^{i_j} v_i y_i \tag{40}$$

12. Squared log-return per second
    Squared log-return divided by the interval duration, where D_j is the *interval duration* and r_j is the log-return of the segment j.

$$r^2_{D_j} = \left(\frac{r_j}{\sqrt{D_j}}\right)^2 = \frac{r_j^2}{D_j} \tag{41}$$

$$r_j = log(y_{i_j}) - log(y_{i_{j-1}+1}) \tag{42}$$

13. Total squared log-returns per second
    Sum of squared log-returns per second in each segment.

$$S_{r_{ji}} = \sum_{i_{j-1}+1}^{i_j-1} \left(\frac{r_i}{\sqrt{d_i}}\right)^2 = \sum_{i_{j-1}+1}^{i_j-1} \frac{r_i^2}{d_i} \tag{43}$$

14. Squared log-return per second variance
    Variance of squared log-returns per second in each segment.

$$\sigma^2_j = \frac{1}{e_j - 2} \sum_{i_{j-1}+1}^{i_j-1} \left(\left(\frac{r_i}{\sqrt{d_i}}\right)^2 - \mu_{r_x^2}\right)^2 \tag{44}$$

$$\begin{aligned} \mu_{r_x^2} &= \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j-1} \left(\frac{r_i}{\sqrt{d_i}}\right)^2 \\ &= \frac{1}{e_j - 1} \sum_{i_{j-1}+1}^{i_j-1} \frac{r_i^2}{d_i} \end{aligned} \tag{45}$$

15. Average buy market
    For every time x_i, we have buy prices and volumes for each level of the order book. We multiply each price by its corresponding volume, then we add the products obtained from all levels. Finally, we calculate the arithmetic mean of the sums obtained for timestamps x_i in each segment j, where $P_b V_b$ is the sum of the price-volume products from all order book levels at times x_i.

$$\bar{V}_{b_j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} (P_b V_b)_i \tag{46}$$

16. Average sell market
    Calculated in the same way as the *average buy market*, but for the sell side of the order book.

$$\bar{V}_{s_j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} (P_s V_s)_i \tag{47}$$

17. Average buy volume
First, we add the buy volumes from the order book for each timestamp x_i from the trades series. Lastly, we calculate the arithmetic mean of these sums, where V_b is the sum of buy volumes at each timestamp.

$$\bar{V}_{b_j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} (V_b)_i \tag{48}$$

18. Average sell volume
Calculated as in the feature *average buy volume*, but with sell volumes.

$$\bar{V}_{s_j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} (V_s)_i \tag{49}$$

19. Average five levels OBI
It is a type of order book imbalance (OBI). First, we add the volumes from the first five levels for each side of the order book at each timestamp from each segment. Then we divide the difference between the volumes obtained, buy minus sell, by the sum of these volumes. Finally, we calculate the arithmetic mean of these fractions at each timestamp from each segment.

$$\overline{OBI}_{5_j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} \frac{V_{5b_i} - V_{5s_i}}{V_{5b_i} + V_{5s_i}} \tag{50}$$

20. Average ten levels OBI
It is calculated in a similar way as the average five levels OBI, but with ten levels.

$$\overline{OBI}_{10_j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} \frac{V_{10b_i} - V_{10s_i}}{V_{10b_i} + V_{10s_i}} \tag{51}$$

21. Average total levels OBI
Similarly as in the two previous features, the average total levels OBI is obtained for all levels from the order book in each segment.

$$\overline{OBI}_{j} = \frac{1}{e_j} \sum_{i_{j-1}+1}^{i_j} \frac{V_{b_i} - V_{s_i}}{V_{b_i} + V_{s_i}} \tag{52}$$