A machine learning ensemble approach to predict financial markets manipulation

Tatiana Franus[a], Malvina Marchese*[a], Richard Payne[a]

[a]*Bayes Business School, City University of London, 106 Bunhill Row, London EC1Y 8TZ, UK*

Abstract

Forecasting spoofing manipulation in financial markets is the cornerstone of regulators to improve trading surveillance systems. This paper introduces a new data-driven approach to forecast the market conditions associated with episodes of spoofing manipulation. Our approach reduces the importance of model selection through forecasts combination of different traditional machine learning methods. We apply the forecasting algorithm to a unique dataset of suspicious spoofing cases detected on the Moscow Stock Exchange. Our results show that learning from the limit order book and past suspected spoofing cases generates an effective manipulation prediction measure at short horizons in a high frequency data environment. The Real-Time Spoofing Probability measure proposed is shown to be an effective real-time indicator of risk to trade in manipulative environment, which can be used by exchangers, market participants and regulators in surveillance systems.

Keywords: Price Manipulation, Spoofing, Forecast Combination, Random Forest, XGBoost, Ensamble

---

1. Introduction

Electronic markets and high-speed and automated trading have transformed the financial market landscape and introduced new possibilities for disruptive practices. Such practices have the potential to enable traders to make considerable profits through the artificial manipulation of market perceptions, which, in turn, may detrimentally impact other market participants. The strategic manipulation of markets can lead to price distortions, posing a substantial threat to the trustworthiness and integrity of capital markets. The resulting mispricing may undermine investor confidence

Email addresses: `tatiana.franus.2@city.ac.uk` (Tatiana Franus), `malvina.marchese@city.ac.uk` (Malvina Marchese*), `richard.payne.1@city.ac.uk` (Richard Payne)

and damage market participation, efficiency, liquidity and the overall development of financial markets (Guiso et al., 2008; Imisiker and Tas, 2013; Punniyamoorthy and Thoppan, 2013).

One prominent form of trade-based manipulation is 'spoofing'. Spoofing involves the placement and subsequent withdrawal of limit orders to feign supply or demand interest. In spoofing strategies, spurious orders are typically situated close to the best bid and ask prices within the order book, a positioning adopted by manipulators to help them to masquerade as traders with legitimate buy and sell interests. In the aftermath of the enactment of the Dodd-Frank Act, which outlawed such activities (Dodd-Frank, 2010), substantial investments have been made in the enhancement of automated surveillance systems tasked with its detection. The Commodity Futures Trading Commission (CFTC) responded to the threat posed by spoofing with the establishment of a dedicated Task Force in 2018, signaling the critical importance of thwarting such manipulation for regulatory bodies. Notably, financial institutions such as Citigroup and J.P. Morgan Chase have been subjected to sizable fines amounting to \$25 million and \$920 million, respectively, for engaging in spoofing within the US Treasury futures market between the years 2017 and 2020.

The threat of manipulation and the necessity to police it has prompted the emergence and rapid expansion of a trade surveillance industry dedicated to the oversight of client transactions and the identification of illicit trading activities (Cumming et al., 2011; Aitken et al., 2015). However, these surveillance systems still lack effective tools for predicting spoofing activities — a challenge that impedes regulators' real-time monitoring efforts.

This paper provides four innovative contributions to the literature on spoofing manipulation in financial markets. First and foremost, we develop a novel data-driven approach to forecast the market conditions associated with episodes of spoofing manipulation. Second, we identify the link between market states and limit order book variables. Third, we offer an extensive and systematic comparison of the out-of-sample performance of multiple machine learning techniques applied to spoofing detection. Finally, we propose an effective spoofing prediction measure, the Real-Time Spoofing Probability (RTSP), based on the combination of supervised machine learning algorithms. Our approach is evaluated using a unique and novel data set of suspected spoofing cases detected by the Moscow Exchange (MOEX).

In the last decade, there has been a growing interest in applying machine learning to the challenge of detecting market manipulation. Öğüt et al. (2009) investigate the effectiveness of artificial neural networks and support vector machines to identify manipulation on the Istanbul Stock Exchange. Golmohammadi and Zaiane (2015) advocate for contextual anomaly detection algorithms and demonstrate their superior performance over k-Nearest Neighbor (kNN) techniques in a broad stock market manipulation identification. Cao et al. (2014) employ a selection of machine learning algorithms to ascertain the presence of price manipulation within financial markets. Their outcomes suggest that the k-Nearest Neighbour algorithm and the One-Class Support Vector Machine are capable of detecting manipulative behaviors with a notable degree of effectiveness. Expanding upon this line of inquiry, Cao et al. (2015) evaluate the performance of the Adaptive Hidden Markov Model against traditional machine learning paradigms, including Gaussian Mixture Models and the k-Nearest Neighbour algorithm to identify manipulative trade activities on the NASDAQ and the London Stock Exchange. The results from their analysis indicated a superior detection capability. Tuccella et al. (2021) use Gated Recurrent Unit (GRU) based architecture for spoofing detection on several cryptocurrency exchanges while Tao et al. (2022) proposes a quantification instrument to monitor real-time spoofing behaviour on the TMX market using a conditional Wasserstein distance.

All these studies suffer from certain limitations. First, they identify spoofing orders using their algorithm, which questions the robustness of the detection method. In contrast, we identify spoofing using suspected spoofing incidents recognized by exchange authorities. Moreover, most studies rely on synthetically generated data only to validate their findings. We rely on a novel and extensive dataset with complete historical information of orders placed on the market. In the empirical application, we create a unique dataset by combining data provided by MOEX with suspected spoofing orders identified by the exchange's internal algorithm on ten liquid stocks and data of all orders and trades over the same period. Those ten stocks are not the most traded stocks on the Moscow exchange, so the choice of sample stocks is in line with the results of Williams and Skrzypacz (2020), which suggest that spoofing should be most prevalent in markets which are sufficiently liquid but not too liquid. Using files from MOEX with all orders and trades, we track all orders placed on the market, identify whether they are filled or cancelled and reconstruct an entire limit order book at each tick, thus replicating the MOEX's spoofing identification methodology, albeit with more constrained data sets. To the best of our knowledge, we are the first to use this level of granularity to address the spoofing problem. All of our algorithms are rigorously trained on a dataset comprising suspicious spoofing orders, flagged by the Moscow Exchange. Our results suggest that using machine learning algorithms trained on public domain data readily can effectively supplant the need for exclusive data privileged to market participants, such as trader identities and their respective trading styles.

An additional issue in the current literature is the dependence of the predictive results on the breadth and depth of the training data. Achieving a sufficient volume of trading data that is adequately labeled for the purposes of supervised learning presents a significant difficulty. Existing research on manipulation detection largely utilizes first and second-level data within the order book, often restricted to the top five tiers of depth, which may not have the sufficient informational depth to detect spoofing. We circumvent this limitation by adopting a more comprehensive analytical perspective, leveraging a dataset that encompasses up to fifty tiers within the order book. This allows us to examine a broader range of potential market state indicators, inclusive of novel metrics such as the order book filling ratio and the order book spread. We categorize our set of indicators into three distinct groups: market quality indicators, trade-related metrics, and operational state descriptors of the order book. Using data-driven feature importance selection, we distill hundreds of variables down to those of significance in forecasting the optimal market conditions for manipulators to introduce orders. This approach allows us to encapsulate key economic dynamics prevalent in limit order markets, yet remains manageable through the implementation of dimensionality reduction techniques.

Most studies in the current literature focus on the applications of just one or two machine learning algorithms. In contrast, in this paper we conduct a comprehensive analysis of the out-of-sample performance of multiple ML techniques, i.e. regularization (lasso, elastic net), tree-based algorithms (Random Forest, XGBoost and Decision Tree), and Neural Networks. We aim not only to do a complete comparison across ML methods but also provide evidence of how and why some of these methods improve the accuracy of forecasting spoofing manipulation. In the comparison, we calibrate the algorithms using the data from five preceding trading days, and then conduct predictive analysis for the subsequent 10-, 30- and 60-minute windows. Through the aggregation of probabilities yielded by each individual machine learning algorithm, we formulate a composite real-time spoofing likelihood score. Our results confirm the superior forecasting performance of our methodology with respect to other machine learning methods, thereby affirming the real-time responsiveness and effectiveness of our RTSP metric in identifying market states with elevated spoofing propensities. We hope that our approach may be of interest to regulatory bodies, enhancing the detection of spoofing within the limit order book (LOB) while simultaneously augmenting market quality. By incorporating an RTSP-based spoofing risk assessment, market participants can also adjust their trading strategies, thereby altering the market dynamics and rendering conditions less amenable for low-risk manipulator gains, thus inherently improving market quality.

The empirical results show that no individual algorithm consistently outperforms the other. We show that the RSTP measure, a forecast combination of the two best performing learners , the Random Forest and XGBoost, achieves a superior out-of-sample predictive performance with respect to any algorithm. Our analysis provides strong evidence of the model appropriateness for imbalanced datasets, such as the real-time tick data used in our empirical study and facilitates the identification of intervals featuring potentially suspicious and fraudulent activities within the order book.

The remainder of the paper is organized as follows. Section 2 describes the data and the variables. The following parts of the paper are dedicated to the methodology we use in our study (Section 3) and the results that we obtain to forecast spoofing (Section 4). In Section 5, we introduce the novel forecasting measure RTSP and show its predictive power (Section **??**). Finally, some conclusive remarks are offered in Section 6.

2. Data collection and processing

In our empirical investigation, we use high-frequency tick market data for six consecutive months from January till June 2019 for a total of 103.412 observations. Our data is constructed using two dataset obtained from the Moscow Exchange (MOEX).

The first dataset contains information on 51,706 cancelled orders of 10 liquid stocks, identified by MOEX, according to their internal algorithm, as potential spoofing orders. This dataset provides information on security id, time, buy/sell, volume and price. Data providers do not release the algorithms used for spoofing detection by the exchange surveillance department, however they track potential spoofing orders by observing how the order influences the market, whether it tightens the spread, which is its lifetime and using several other indicators. In addition, the exchange tracks the trading style of each trader and can mark potential spoofers. Suspicious spoofing orders in this dataset are identified based on the information regarding proven detected spoofing cases from 2010 to 2019 in the USA, Europe, and the UK.

The second dataset consists of all orders placed on the stock market and includes for each order: record number, security code, type of order (sell or buy), time in the format of microseconds, order number, type of action (placed, cancelled, executed), price, volume, trade number if executed, trade price if executed. This dataset includes all orders on MOEX, all trades, and best bid and ask prices. This information allows us to reconstruct the full limit order book (LOB) up to maximum available level at any time and for any stock. We construct the book 50 levels deep.

We construct our dataset by matching suspicious spoofing orders identified by MOEX with the second dataset of orders and trades. This allows us to track the manipulative order's lifetime and to identify its order book price level and time of cancellation. We call the spoofing orders "True" orders. We then randomly pick non-spoofing orders during the same period and denote them as "False" orders. Table 1 includes description and general statistics of the dataset with the spoofing orders.

Table 1: Description and general statistics of the spoofing orders

format:	security id, time, buy/sell, volume, price
time:	6 months from January till June 2019
10 stocks:	FIVE', 'GMKN', 'MGNT', 'NVTK', 'PLZL', 'POLY', 'ROSN', 'SNGS','TATN', 'YNDX'
number of orders:	51 706 spoofing orders
lifetime:	on average the lifetime of spoofing order is 0.0045 min with maximum lifetime 0.33 min
price level:	on average spoofing order appears on the 5th price level below or above best bid or ask in the order book
max price level:	the maximum price level is 48
trades information:	on average 84% of spoofing orders do not have any trade during their lifetime
orders frequency:	spoofing orders appear every day on each stock
buy or sell:	55% of spoofing orders are sell orders
time of the day:	orders appear equally during the day, except YNDX, which spoofing orders appear only after 4pm

For the ten stocks, we reconstruct the limit order book every 10 seconds (Table 1). Each reconstructed order book consists of 50 ask prices and bid prices. We use price levels depending on the stock's minimum price increment, including zero volume price levels. We only consider normal trading hours[1] and omit the first and last half hours of each trading day, since these are often periods of high volatility and increased spreads (Cartea et al., 2019). Every price level in the reconstructed order books represents the volume of orders placed on the market from the beginning of the trading day minus cancelled and executed orders on the same price level. The data structure enables us to identify the exact time when the suspicious spoofing order was placed and the market

[1]Normal trading hours on MOEX are from 10:00 till 18:45.

state just before that order, consisting of the reconstructed limit order book, the trading price and volume.

In the first part of our analysis , the dataset is split into two parts. A training and validation sample (in-sample) which includes 75% of the full sample, for a total of and a testing sample which includes the remaining 25% of the sample. The model development process is performed in the in-sample dataset and model validation and assessment is conducted on the testing sample. In the second part of our analysis, we conduct a forecasting assessment of the algorithms on the full sample, considering real-time prediction of spoofing at the next tick, based on information from the last five days of trading only.

In developing our model specification, we examine an extended set of variables that follow under the classification categories of market quality variables, variables of trades and their frequency, and order book variables. As market quality variables we consider

1. Spread measures:
   - QS. Quoted spread is $(a-b)/m$, where a is the best ask, b is the best bid, and m is the midquote;
   - QS_delta. The difference in quoted spreads measures ten and twenty seconds before the spoofing order.
   - QS_delta_t2. The difference in quoted spreads measures ten and thirty seconds before the spoofing order.
   - ES. Effective (half) spread is $(p-m)/m$, where p is the trade price and m is the prevailing midquote (prior to execution) (Lee, 1993; Blume and Goldstein, 1992).
2. Short-term volatility measures:
   - VOL_1_Nmin. N-minute price volatility is a standard deviation of the midquotes within the interval divided by the last midquote (Aitken and Frino, 1996). $N \in \{1, 2, 5, 10\}$ minutes.
   - VOL_2_Nmin. N-minute price volatility is measured as a standard deviation of price returns, where $N \in \{1, 2, 5, 10\}$ minutes.
   - VOL_3_Nmin. Average realized volatility in the time interval is measured as $1/n\Sigma_{s=1}^{N}\left|ln(m_{t-s} - m_{t-s-1})\right|$, where m_s is the midquote at end of minute s, and $N \in$

$\{1, 2, 5, 10\}$ minutes.

In constructing variables of trade and its frequency, the interval choice should be aligned with the main research question. As we have data with orders that live on average less than a second, we consider short intervals of less than 10 minutes. One-minute intervals coincide with research on quote stuffing events (Egginton et al., 2016), while 10-minute intervals are used by Cartea et al. (2019) and Hasbrouck and Saar (2013). For a better picture, we include 1-, 2-, 5-, 10-minute intervals and consider the following:

1. $Hour$ is a trading hour from 10 am till 6 pm in which the tick prevailing the spoofing order occurred.
2. UF. Ultra-fast activity measure is a fraction of limit orders submitted and cancelled very quickly out of all cancelled orders (Cartea et al., 2019). UF_X_N is an ultra-fast measure cancelled within X ms, where $X \in \{1, 10, 50, 100, 600\}$ ms in one minute, and N indicates how many minutes before spoofing order ultra-fast activity happened, where $N \in [1; 10]$ minutes. For example, spoofing order was placed at 10:30, so UF_10_5 is a fraction of the limit orders cancelled within 10 ms measured in the minute from 10:25 to 10:26.

The order book state is a "snapshot" of the LOB at every point in time when at least one action (trades, placement or cancellation of limit orders) is present. Two neighbouring order book states could have the time index at least a one-time increment possible on the exchange. [2] However, neighbouring order book states may be several seconds apart during slow trading times. We consider the order book state just before the spoofing order is placed [3] and compute separate order book variables for the LOB's ask and bid sides. To determine the order book state before the manipulative order, we have to consider that spoofing orders appear on one side of the LOB. Does it mean that the market state on this side is the one that attracts manipulators? While separating the dataset into two subsets of buy and sell orders, we address this question. Also, we calculate the order book depth variables separately for the ask and bid sides of the LOB.

1. Order book imbalance. Volume imbalance of the limit order book is the ratio of volume posted at the best bid price minus the volume posted at the best ask price to the sum of the volume at the best bid and best ask price (Cartea et al., 2020). We compute those imbalances using:

[2]MOEX provides data in the format HHMMSSZZZXXX, which gives us the minimum time increment of 0.000001 seconds.

[3]The mean lifetime of spoofing orders is 0.005 seconds.

- Total order book imbalance:

  $IMB = (SB - SA)/(SA + SB)$,

  where SB is a sum of the orders' volume on all bid price levels, SA is a sum of the orders' volume on all ask price levels.

- Order book imbalance until i price level:

  $IMB\_i = (SB_i - SA_i)/(SA_i + SB_i)$,

  where $i \in \{0, 1, 2, 3, 4, 5\}$ is a price level in the order book, where $i = 0$ is the best ask and best bid price levels.

- Order book imbalance until the price level of the spoofing order:

  $IMB\_order = (SB^* - SA^*)/(SA^* + SB^*)$,

  where SB^* and SA^* are the sums of the orders' volume from the bid and ask sides of the order book, respectively, that lie within the price band defined by the spoofing order.

- Difference in the order book imbalance:

  $IMB\_delta = IBM\_10 - IBM\_20$,

  where IBM_10 is an order book imbalance measured ten seconds before the spoofing order. IBM_20 is the order book imbalance variables for the order book twenty seconds before the spoofing order.

  $IMB\_delta\_t2 = IBM\_10 - IBM\_30$,

  where IBM_30 is the variable for the order book thirty seconds before the spoofing order.

2. Order book spread. Bid-ask spread is a classical liquidity measure. It is a trade-based effective cost estimate that has been widely used in studies of market quality. We introduce a measure of order book spread to indicate hidden liquidity inside the LOB that could be seen not as a snapshot of market quality but as potential longer-lasting liquidity.

   - Order book spread:

     $DistNormN = (AvAskN - AvBidN)/m$, where $AvAskN$ and $AvBidN$ are a mean ask and bid prices respectively that lies within $N \in \{10, 20, 50\}$ ask and bid price levels including those with zero volume; m is a midquote;

- Clean order book spread:

  $DistNormCleanN$ is the same measure as $DistNormN$, but we do not consider price levels with zero volume. $N \in \{10, 20, 50\}$ not including zero volume levels.

3. Order book depth. Biais et al. (1995) find that thin books elicit orders and thick books result in trades. Ranaldo (2004) uses similar measures like pending volume in the number of shares divided by 10,000 at the best quote on the same market side as the incoming trader and on the opposite side of the market for the incoming trader. We follow a similar logic by introducing the following variables:

   - Average order book depth:

     $DistVolN = (VolAsk + VolBidN)/Vol\_m$ is the average cumulative volume resting in the LOB within $N \in \{10, 20, 50\}$ basis points of the best bid and ask (Cartea et al., 2019). $VolAskN$ and $VolAskN$ are cumulative volumes on the order book's ask and bid sides, resting within the N price level.

     $Vol\_m = (VolAsk + VolBid)/2$ is an average volume on the best bid and ask price levels. This measure accounts for the tick size by measuring the depth at given distances relative to the current best bid and ask. We normalize this measure by average volume on the best bid and ask price levels.

   - Average clean order book depth:

     $DistVolCleanN$ is the same measure as $DistVolN$, but we do not consider price levels with zero volume. $N \in \{10, 20, 50\}$ not including zero volume levels. For example, $N = 10$, but we observe two price levels from the ask side with zero volume. So, we must count cumulative volume from the ask side until the twelfth price increment from the best ask.

4. Order book filling ratio. The order book filling ratio is the number of filled price levels with the limit orders divided by the number of all price levels. The order book has several visible price levels, however, the difference between neighbouring levels could be more significant than a minimum price increment. In other words, the order book could be filled with orders on each price level or many empty price levels. We propose the following novel variables to capture this ratio:

- $FRA = FA/TotalA$ and $FRB = FB/TotalB$, where FA and FB are the counts of price levels in the order book with at least one sell and buy limit order, respectively. $TotalA$ and $TotalB$ are the maximum price levels from the order book's ask and bid sides.
- $FRA\_delta = FRA\_10 - FRA\_20$ and
  $FRB\_delta = FRB\_10 - FRB\_20$, where FRA_10 and FRB_10 are order book filling ratios measured ten seconds before the spoofing order. FRA_20 and FRB_20 are order book filling ratios for the order book twenty seconds before the spoofing order.
  $FRA\_delta\_t2 = FRA\_10 - FRA\_30$ and
  $FRB\_delta\_t2 = FRB\_10 - FRB\_30$,
  where FRA_30 and FRB_30 are order book filling ratios for the order book thirty seconds before the spoofing order.

The variable generation process leads to a large set of predictors as potential candidates for our modeling procedures. Our decision to explore such a large set of predictors is motivated by the fact that the current literature that has inconclusive evidence on which variables to include in predicting spoofing manipulation (Cao et al., 2015; Tao et al., 2022). That is, our choice is driven by our intention not to miss any important information.

We first reduce the dimensionality space of the candidate variables that articulate the current state of the market, so that the core models are developed on a sub sample of variables that includes the most relevant ones for the purposes of the analysis. As a first step , to ensure a consistent framework for comparison across different stocks and to facilitate interpretability, each variable is standardized by removing outliers, characterized by deviations exceeding three standard deviations from the mean of each feature. Then, considering the large number of explanatory variables, we proceed by excluding the ones that convey, to a large extent, the same type of information in predicting the dependent variable using regularisation techniques. We use LASSO (Least Absolute Shrinkage and Selection Operator) as a regularization technique for variable selection, as detailed by (Tibshirani, 1996; Hastie et al., 2009) (see Appendix 7.1) to assess simultaneously the predictive power of features. However, while LASSO is capable of accommodating pre-specified non-linearities and interactions, it may exhibit sub-optimal performance in the presence of highly correlated predictors (Zou and Hastie, 2005). The sample correlation matrix of our candidate predictors is provided in

Appendix 7.2. Given the substantial correlation among predictors, in line with the findings of Zou and Hastie (2005), we rely on the Elastic Net to robustify our findings from LASSO. We choose a parameterization with an alpha of 50% (Appendix 7.1)). The regressors to be included in our model are selected based on their capacity to minimize the Root Mean Square Error (RMSE). As the outcome of the variable selection process, we retain 67 regressors as predictors of spoofing manipulation trading activity. The full list of predictors is reported in documented in Appendix 7.3. At this point, it is critical to ensure that the final system of variables remains robust for different model specifications. To this end, we extend our analysis by also applying a Stepwise Logit (SL) model specification. Detailed results are available upon request from the authors. The variables selected are robust to the choice of model specification, so our main conclusions regarding the predictive performance of each model seem not to be biased against the variable selection algorithm used.

3. Model Development

To predict the probability of spoofing manipulation at the next tick , we consider several machine learning methods, including deep learning ones, and conduct and extensive comparison. Namely, the following classical machine learning models are included in our analysis: Logistic Regression (LogR), Support Vector Machines (SVM), k-Nearest Neighborhoods (KNN), Naive Bayes (NB), Stochastic Gradient Descent (SGD), Decision Trees (DT), Random Forest (RF), Gradient Boosting(GB), XGBoost and Adaptive Markov Chain. For the Neural Networks (NNs), we include the Perceptron and the LSTM and the GRU. All these models are well-documented in the literature.

The *Logistic Regression* is used as a benchmark. It is a non-linear fully parametric approach and has several advantages: it is easy to fit and to interpret. Statistical significance tests can be easily performed. However, parametric methods have a disadvantage: by construction, they make strong assumptions and if the specified functional form is far from the truth, then the parametric method will perform poorly. The simplest non-parametric method we consider is the *k-Nearest-Neighbors (k-NN)*. k-NN categorizes objects based on the classes of their nearest neighbors in the dataset. Distance metrics are used to find the nearest neighbor. The k-NN algorithm searches for k closest training instances in the feature space and uses their average prediction. k-NN predictions assume that objects near each other are similar. When mobilizing k-NNs, memory usage and prediction speed of the trained model are of lesser concern to the modeler.

Support Vector Machines (SVM) map inputs to higher-dimensional feature spaces. The SVM classifies data by finding the linear decision boundary (e.g., hyperplane) that separates all data points of one class from those of the other class. This machine-learning algorithm separates the attribute space with a hyperplane, maximizing the margin between the instances of different classes or class values. It can be used when the researcher needs a classifier that is simple, easy to interpret, and accurate. A significant number of studies point the usefulness of SVMs in market manipulation detection (Öğüt et al., 2009; Khodabandehlou and Golpayegani, 2022), since they reduce the possibility of overfitting and alleviate the need of tedious cross-validation for the purpose of appropriate hyper parameter selection. Their main limitation is the lack of intepretability.

Decision trees are resilient to variable scaling and robust against outliers and have been used by several authors as tools to predict spoofing.For instance, when the trading volume exceeds the 95th percentile, a split is made based on whether the quoted spread is above or below the median, otherwise, the split is based on the 25th percentile of order book imbalance. Each of the resulting four leaves is then assigned an expected frequency of spoofing activity, in line with the historical averages. In recent years, ensemble algorithms, particularly tree-based algorithms, have emerged as pivotal tools for addressing prediction and classification challenges across diverse domains, yielding notable achievements (Zhou, 2012). Rather than relying on a singular model, tree-based methods amalgamate multiple individual tree models to attain optimal prediction performance. The *Random Forest*, introduced by Ho (1995), operates as a supervised ensemble learning approach based on decision trees. It aggregates results from multiple random decision trees (Breiman, 2001), averaging many trees, each built on a bootstrapped sample from a random subset of all predictors originating from the original dataset.

The Gradient Boosting (XGBoost), effectively implementing Gradient Tree Boosting, as outlined in Chapter 10 of Hastie et al. (2017). While Random Forest relies on averaging over random trees (referred to as bagging), in Gradient Tree Boosting, each new tree focuses on examples that previous trees found challenging (known as Boosting). Generally, Boosting yields more accurate forecasts than bagging, albeit the estimation process is considerably slower. XGBoost effectively addresses this slowdown, making Gradient Tree Boosting nearly as fast as Random Forest. Moreover, it recognizes the propensity of trees to overfit and imposes penalties on trees with excessive leaves, favoring simpler and more concise trees—a process commonly referred to as regularization. As discussed in Chen et al. (2015), the XGBoost algorithm is equally adept at providing computational efficiency, particularly when dealing with sizable datasets. Moreover, XGBoost, akin to Random Forest, is adept atleveraging hyperparameters which aligns well with our data requirements, specifically our need to wield various order book variables for spoofing prediction, incorporating clusters, rates, and other parameters within a unified algorithm.

4. Model Validation

To compare models performance we conduct an extensive validation exercise in term of predictive ability in the out of sample (testing) period. To ensure robustness of our findings to the validation procedure, we consider several different cross-validation approaches in controlling for overfitting.

4.1. Validation measures

To discriminate between competing forecasts, we compare the predictions using a series of metrics, namely *accuracy*, *precision*, *recall* and $f1 - score$. These measures overcome any issues of model performance misinterpretation, which can arise in cases where one uses traditional model performance metrics that are based on overall model accuracy. We rely on *accuracy*, *precision*, *recall* and the $f1 - score$, defined as follows:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}, \ Precision = \frac{tp}{tp + fp},$$

$$Recall = \frac{tn}{tn + fp}, \ F1 - score = \frac{2 * Recall * Precision}{Recall * Precision},$$

where, tp is a number of true positive values, tn is a number of true negative values, fp is a number of false positive values, and fn is a number of false negative values. The *accuracy* measure denotes the proportion of correctly forecasted market states to the total number of all forecasted market states. In other words, it measures the portion of all testing samples classified correctly. The *precision* measures the proportion of all correctly identified samples in a population of samples which are classified as positive labels. Thus it answers the question: Of all "True" predicted market states with spoofing orders, how many actually had spoofing orders in the LOB? For example, if *precision* of "True" spoofing orders equals 0.82, it means that 82% of all predicted market states with spoofing orders in the LOB were correctly identified by the ML algorithm. Similarly, if *precision* of "False" spoofing orders equals 0.74, it means that 74% of predicted market states without spoofing orders in the LOB were correctly identified. *Recall* or sensitivity measures the ability of a classifier to identify positive labels correctly. *Recall* answers the question: Of all actual "True" market states with spoofing orders, how many were predicted correctly by the ML algorithm? For example, if *recall* of "True" spoofing orders equals 0.71, it means that 71% of market states with spoofing orders in the LOB were identified correctly by the ML algorithm. Similarly, if *recall* of "False" spoofing orders equals 0.84, it means that 84% of market states without spoofing orders in the LOB were identified correctly. The $f1 - score$ combines *precision* and *recall* into a single metric as the harmonic mean of *precision* and *recall*, By definitions, it is never higher than the geometrical mean and tends towards the least number, minimizing the impact of the large outliers and maximizing the impact of small ones. $F1-score$, therefore, tends to privilege balanced systems. To test for equal predictive ability of the models we use the Diebold Mariano test on model pairs for each of the accuracy measures. Finally, we rely on the Model Confidence set to rank models forecasting performances.

4.2. Validation Results

The fully balanced nature of our original sample may facilitate the training of the models. To investigate the robustness of our approach to the more realistic scenario of an unbalanced data set, we analyze model performance for the case of 1:2 True-False proportion. To this end, we create a new training sample, including further 51,706 randomly selected False orders. We assess model performance in the "balanced" (original) and "unbalanced" dataset. As already mentioned, the validation period refers to 25% of the full data. Our analysis reveals an average prediction accuracy ranging from 50% and 76% (cf. Table 2) for the balanced sample and indicates the Random Forest as the model with best performance. These finding our supported also by the Diebold-Mariano test which allows us to reject the null of equal predictive ability of all models against the Random Forest. In addidition, the DM test indicates that the Decision Tree and XGBoost models either outperform or exhibit comparable performance to the other models (see Appendix 8.4). Results from the Model Confidence set confirm the Random Forest model as the optimal choice in term of prediction accuracy . Subsequently, in quest of identifying the second-best model, we exclude the Random Forest from consideration, leading us to identify two other models, Decision Trees and XGBoost that exhibit similar and commendable performance levels (refer to Appendix 8.5) Following this first step of analysis, the Random Forest, Decision Tree, and XGBoost undergo calibration to achieve optimization of the hyperparameters (refer to Appendix 7.6). Table 2: Prediction accuracy of ML models. The table shows results of commonly used ML models with True and False orders proportion of 1:1, splitting the data into 75

Table 2: Prediction accuracy of ML models. The table shows results of commonly used ML models with True and False orders proportion of 1:1, splitting the data into 75% training and 25% validating sets.

Logistic regression:	52%
Support Vector Machines (SVM):	57%
K-nearest neighbors (KNN):	68%
Naive Bayes (NB):	50%
Stochastic Gradient Descent (SGD):	52%
Decision Tree (DT):	68%
Random Forest (RF):	76%
Gradient Boosting (GB) :	67%
XGBoost:	68%
Adaptive Markov Chains	50%
LSTM Neural Network (10 hidden layers)	55%
LSTM Neural Network (5 hidden layers)	55%
GRU neural network	51%
Percepton neural network	53%

Overall three algorithms on the first step of the analysis predict correctly from 66% to 78% of spoofing cases for balanced and un blanced data, the Random Forest, the Decision Tree and the XGboost.

4.3. Robustness

We investigate robustness of the results to various cross validation setting in controlling for overfitting. Cross-validation is a resampling procedure used to evaluate the consistency of prediction quality of machine learning models on a limited data sample. The process has a single parameter k that refers to the number of groups a given data sample will be split into. We start with the 5-fold cross-validation for each ML model and obtain average prediction accuracy. It generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train-test split. The general procedure is as follows: firstly, the dataset is shuffled randomly; secondly, the dataset is split into k groups to take each group as a hold-out or test data set and then fit a model on the training set and evaluate it on the test set; and finally, retain the evaluation score and discard the model. Then we summarize the model's skill using the sample of model evaluation scores. Then we assess the model forecasting performance based on the mean accuracy of the prediction of each fold. We use K-fold, Stratified K-fold, and Shuffle cross-validations, splitting the spoofing events into K non-overlapping parts in calendar time. We run cross-validation with different Ks (3, 4, 5, 6, 8, 10) to test for the stability of the outcomes. In K- fold cross validation, every fold is used as a validation set and other left-outs as a training set. This validation technique is not considered suitable for imbalanced datasets as the model will not get adequately trained owing to the proper ratio of each class data, thus on the imbalced data we train the models using the shuffling and block approach. The typical approach when using k-fold cross-validation is to shuffle the data randomly and split it in K equally-sized folds. By shuffling the dataset, we ensure that the model is exposed to a different sequence of samples in each part, which can help to prevent it from memorizing the order of the training data and overfitting to specific patterns. The blocked k-folds cross-validation procedure is similar to the standard form described above. The difference is that there is no initial random shuffling of observations. The stratified K-fold cross-validation , an enhanced version of the k-fold cross-validation technique, splits the dataset into K equal folds. Each fold has the same ratio of instances of target variables. This method enables us to work with imbalanced datasets. Finally, we use the shuffle cross-validation technique that involves splitting the whole data in the percentage of your choice and keeping the train-test split percentage different. This method allow us to get the test errors and access the outcomes.

4.4. Forecasting Results for balanced and imbalanced datasets

In the case of a balanced dataset, the three machine learning algorithm models chosen on the first step of the analysis predict correctly from 66% to 78% . Tables 3, 4, 5, 6 below show the results.

Table 3: Accuracy of ML models. The table shows results of Random Forest, XGBoost and Decision Tree ML models with True and False orders proportion are equal to 1:1, splitting the data into 75% training and 25% validating sets. *Accuracy* measures the portion of all testing samples classified correctly. *Recall* measures the ability of a classifier to correctly identify positive labels. And *Precision* measures the proportion of all correctly identified samples in a population of samples which are classified as positive labels. $F1 - score$ combines *precision*, *recall* into a single metric, $F1 - score = (2 * Recall * Precision)/(Recall * Precision)$.

	Random Forest			XGBoost			Decision Tree		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
False	0.74	0.84	0.79	0.78	0.63	0.69	0.64	0.72	0.68
True	0.82	0.71	0.76	0.70	0.82	0.75	0.69	0.60	0.64
Accuracy		0.78			0.73			0.66	

Table 4: K-fold cross-validation check for of ML models using balanced data with 1:1 True-False proportion. The table shows the results of two cross-validation approaches: *1. Shuffle*: the complete dataset is shuffled in random k-parts; *2. Blocks*: the dataset is sorted by time into blocks without shuffling.

	1. Shuffle			
K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	76.5	72.5	66.0
	Max accuracy	76.9	72.7	66.6
	Average accuracy	76.7	72.6	66.3
4	Min accuracy	77.1	72.4	65.8
	Max accuracy	77.4	72.9	66.4
	Average accuracy	77.2	72.7	66.1
5	Min accuracy	77.2	72.4	65.6
	Max accuracy	77.7	73.3	66.4
	Average accuracy	77.5	72.9	66.1
6	Min accuracy	77.3	72.0	65.7
	Max accuracy	78.2	73.4	66.4
	Average accuracy	77.7	72.9	66.1
8	Min accuracy	77.4	72.0	65.9
	Max accuracy	78.3	73.2	66.8
	Average accuracy	77.9	72.7	66.1
10	Min accuracy	77.4	72.4	65.5
	Max accuracy	78.4	73.3	66.9
	Average accuracy	78.0	73.0	66.3
	2. Blocks			
K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	56.6	62.0	62.8
	Max accuracy	61.0	66.4	66.7
	Average accuracy	58.4	64.3	64.7
4	Min accuracy	54.9	61.0	59.0
	Max accuracy	61.9	67.9	65.8
	Average accuracy	57.9	64.7	62.2
5	Min accuracy	55.9	63.0	60.1
	Max accuracy	64.2	70.2	69.2
	Average accuracy	59.3	65.9	64.4
6	Min accuracy	55.3	60.0	59.3
	Max accuracy	63.4	68.5	68.4
	Average accuracy	58.6	64.8	64.2
8	Min accuracy	55.3	56.5	56.1
	Max accuracy	68.7	68.6	69.1
	Average accuracy	58.9	64.5	62.8
10	Min accuracy	54.7	59.6	57.2
	Max accuracy	70.3	71.9	71.3
	Average accuracy	59.8	65.9	64.1

Table 5: Stratified K-fold cross-validation check for of ML models using balanced data with 1:1 True-False proportion.

K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	76.4	72.0	65.7
	Max accuracy	76.9	73.0	66.3
	Average accuracy	76.6	72.5	66.1
4	Min accuracy	77.0	72.2	65.3
	Max accuracy	77.3	72.9	66.8
	Average accuracy	77.1	72.7	66.0
5	Min accuracy	77.2	71.9	65.8
	Max accuracy	78.0	73.4	66.8
	Average accuracy	77.5	72.8	66.2
6	Min accuracy	77.4	72.5	65.8
	Max accuracy	78.0	73.5	66.2
	Average accuracy	77.6	72.8	66.0
8	Min accuracy	77.5	72.4	65.5
	Max accuracy	78.4	73.5	67.2
	Average accuracy	77.9	73.0	66.2
10	Min accuracy	77.5	71.7	65.1
	Max accuracy	79.2	73.9	67.1
	Average accuracy	78.0	72.9	66.2

Table 6: Shuffle K-fold cross-validation check for of ML models using balanced data with 1:1 True-False proportion, training on 60% of the data and validating on 20% of the data

K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	76.0	71.9	65.8
	Max accuracy	76.3	72.7	66.1
	Average accuracy	76.2	72.4	65.9
4	Min accuracy	76.1	72.2	65.5
	Max accuracy	76.6	72.6	66.8
	Average accuracy	76.3	72.4	65.9
5	Min accuracy	75.8	72.1	65.8
	Max accuracy	76.7	73.0	66.4
	Average accuracy	76.2	72.6	66.2
6	Min accuracy	75.6	71.7	65.8
	Max accuracy	76.6	72.7	66.4
	Average accuracy	76.2	72.3	66.1
8	Min accuracy	75.7	71.8	65.7
	Max accuracy	76.7	72.8	66.4
	Average accuracy	76.3	72.2	66.1
10	Min accuracy	75.9	71.6	65.6
	Max accuracy	76.6	72.9	66.6
	Average accuracy	76.4	72.3	66.1

Different cross-validation approaches illustrate that the models performance is robust to the validation approach. It is interesting to notice however that according to the K-fold cross-validation with blocks the XGBoost outperforms the Random Forest algorithm, while the shuffle k-fold cross-validation suggests dominance of the Random Forest. There is no uniquely dominant algorithm across different cross validation approaches. This suggest that combining different algorithms may improve the overall forecasting performance (Claeskens et al., 2016). In the case of imbalanced data, the Random Forest model achives 82.3% accuracy. However since the dataset is imbalanced, this implies that only 57% of True orders are predicted correctly, while for False orders, prediction accuracy is 95%. The XGBoost model gives overall 78.2% prediction accuracy with 59% accuracy in predicting True orders and 88% False orders. The decision tree also shows similar results with little less forecasting accuracy (Tables 7, 8, 9, 10).

Table 7: This table shows results of Random Forest, XGBoost and Decision Tree ML models with True and False orders proportion are equal to 1:2, splitting the data into 75% training and 25% validating sets. *Accuracy* measures the portion of all testing samples classified correctly. *Recall* measures the ability of a classifier to correctly identify positive labels. And *precision* measures the proportion of all correctly identified samples in a population of samples which are classified as positive labels. $F1-score$ combines *precision*, *recall* into a single metric, $F1-score = (2 * Recall * Precision)/(Recall * Precision)$.

	Random Forest			XGBoost			Decision Tree		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
False	0.81	0.95	0.88	0.81	0.88	0.84	0.76	0.85	0.80
True	0.85	0.57	0.69	0.71	0.59	0.65	0.61	0.46	0.53
Accuracy		0.82			0.78			0.72	

Table 8: Stratified K-fold cross-validation check for of ML models using imbalanced data with 1:2 True-False proportion.

K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	81.6	77.9	71.3
	Max accuracy	81.9	78.0	72.0
	Average accuracy	81.7	77.9	71.6
4	Min accuracy	81.8	77.9	71.7
	Max accuracy	82.2	78.4	72.1
	Average accuracy	82.1	78.1	71.9
5	Min accuracy	81.9	77.8	71.5
	Max accuracy	82.5	78.4	72.3
	Average accuracy	82.3	78.2	71.9
6	Min accuracy	82.0	77.6	71.2
	Max accuracy	82.7	78.4	72.2
	Average accuracy	82.3	78.2	71.8
8	Min accuracy	82.1	77.8	71.3
	Max accuracy	83.0	78.5	72.2
	Average accuracy	82.6	78.3	71.8
10	Min accuracy	82.1	77.8	71.2
	Max accuracy	83.1	78.6	72.3
	Average accuracy	82.7	78.3	71.8

Table 9: K-fold cross-validation check for of ML models using imbalanced data with 1:2 True-False proportion. We show the results of two cross-validation approaches: *1. Shuffle*: the complete dataset is shuffled in random k-parts; *2. Blocks*: the dataset is sorted by time into blocks without shuffling.

	1. Shuffle			
K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	81.7	77.8	71.8
	Max accuracy	81.8	77.9	72.0
	Average accuracy	81.7	77.9	71.9
4	Min accuracy	81.9	77.9	71.7
	Max accuracy	82.3	78.2	72.2
	Average accuracy	82.0	78.0	71.9
5	Min accuracy	82.0	77.8	71.6
	Max accuracy	82.5	78.3	72.1
	Average accuracy	82.2	78.1	71.8
6	Min accuracy	82.2	77.9	71.7
	Max accuracy	82.7	78.4	72.1
	Average accuracy	82.5	78.2	71.9
8	Min accuracy	82.2	78.1	71.3
	Max accuracy	83.0	78.7	72.3
	Average accuracy	82.6	78.3	71.9
10	Min accuracy	82.3	78.0	71.4
	Max accuracy	83.2	78.6	72.1
	Average accuracy	82.7	78.3	71.8
	2. Blocks			
K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	66.4	68.9	69.3
	Max accuracy	66.5	70.3	70.7
	Average accuracy	66.4	69.7	69.8
4	Min accuracy	66.1	67.6	67.5
	Max accuracy	66.7	72.1	71.6
	Average accuracy	66.5	69.8	69.6
5	Min accuracy	64.7	69.2	68.0
	Max accuracy	70.6	72.7	71.3
	Average accuracy	67.3	70.8	69.7
6	Min accuracy	65.8	68.1	66.2
	Max accuracy	66.9	71.6	71.7
	Average accuracy	66.5	69.7	69.6
8	Min accuracy	64.0	66.0	66.2
	Max accuracy	68.3	73.4	73.7
	Average accuracy	66.6	69.7	69.6
10	Min accuracy	65.0	66.9	66.2
	Max accuracy	74.8	77.3	74.5
	Average accuracy	67.4	71.0	69.8

Table 10: Shuffle K-fold cross-validation check for of ML models using imbalanced data with 1:2 True-False proportion,training on 60% of the data and validating on 20% of the data.

K	Accuracy (%)	Random Forest	XGBoost	Decision Tree
3	Min accuracy	81.0	77.8	71.6
	Max accuracy	81.3	78.0	71.9
	Average accuracy	81.2	77.9	71.8
4	Min accuracy	81.1	77.7	71.3
	Max accuracy	81.6	78.0	72.1
	Average accuracy	81.4	77.9	71.6
5	Min accuracy	81.2	77.7	71.7
	Max accuracy	81.5	78.1	72.2
	Average accuracy	81.4	78.0	71.9
6	Min accuracy	81.2	76.9	71.3
	Max accuracy	81.6	78.2	72.2
	Average accuracy	81.4	77.6	71.8
8	Min accuracy	81.0	77.3	71.4
	Max accuracy	81.7	78.0	72.1
	Average accuracy	81.3	77.7	71.7
10	Min accuracy	80.9	77.4	71.3
	Max accuracy	81.9	78.4	72.3
	Average accuracy	81.5	77.8	71.8

To shed some more light on the performance of the models in the imbalanced dataset, we run the chosen algorithms increasing the proportion of False orders up to 15 by cutting the number of True spoofing orders. The results are presented in Table 11 and confirm that validity of the method strongly relies on the proportion of the data trained.

Various state-of-the-art learning techniques have been suggested in the past few years to address classification problem in imbalanced dataset, including algorithm level methods and data level methods. The algorithm driven approach or classifier level approach keeps the training dataset invariable and adjusts the inference algorithm to facilitate learning specifically related to the minority class. To illustrate this approach, we run the Random Forest for imbalanced datasets called "BalanceRandomForestClassifier" (Appendix 7.6), which randomly under-samples each bootstrap sample to balance it. We obatain a higher predictive accuracy of 80% with a True order prediction of 71%. For the XGBoost to balance order weights, we significantly change the parameter 'scale_pos_weight' from 1 to 500 (Appendix 7.6), which puts 500 times more priority on True orders prediction and we obtain 81% overall accuracy. Results are reported in Table 12.

Table 11: This table shows the results of training ML algorithms on imbalanced data, achieved by cutting the initial dataset of True spoofing orders to achieved the proportion presented in the first column. We split the data into 75% training and 25% validating sets. We report *Accuracy* and *Recall* measures. *Accuracy* measures the portion of all testing samples classified correctly. *Recall* measures the ability of a classifier to correctly identify positive labels. In other words, *Recall* could be seen as an accuracy for True and False orders separately.

True:False	Random Forest			XGBoost			Decision Tree		
	Accuracy	Recall False	Recall True	Accuracy	Recall False	Recall True	Accuracy	Recall False	Recall True
1:1	0.77	0.84	0.70	0.72	0.63	0.82	0.66	0.72	0.59
1:2	0.80	0.95	0.50	0.77	0.88	0.57	0.72	0.90	0.38
1:3	0.83	0.97	0.41	0.81	0.95	0.42	0.77	0.98	0.14
1:4	0.86	0.98	0.36	0.84	0.97	0.35	0.81	0.99	0.13
1:5	0.88	0.99	0.33	0.87	0.99	0.29	0.84	0.99	0.09
1:6	0.89	0.99	0.33	0.89	0.99	0.27	0.86	0.99	0.11
1:7	0.90	0.99	0.28	0.90	0.99	0.23	0.88	0.99	0.09
1:8	0.91	0.99	0.28	0.91	1.00	0.23	0.89	1.00	0.05
1:9	0.92	0.99	0.27	0.92	1.00	0.22	0.90	1.00	0.05
1:10	0.92	0.99	0.24	0.92	1.00	0.20	0.91	1.00	0.06
1:11	0.93	0.99	0.25	0.93	1.00	0.21	0.92	1.00	0.04
1:12	0.93	0.99	0.22	0.93	1.00	0.19	0.92	1.00	0.6
1:13	0.94	0.99	0.24	0.94	1.00	0.21	0.93	1.00	0.3
1:14	0.94	0.99	0.23	0.94	1.00	0.18	0.93	1.00	0.3
1:15	0.95	1.00	0.21	0.94	1.00	0.17	0.94	1.00	0.6

Table 12: This table shows results of modified parameters for Random Forest model using 'BalanceRandomForest-Classifier' (1); and XGBoost ML model with 500 and 10 modification in 'scale_pos_weight' parameter, (2) and (3) respectively. We use imbalanced True and False orders proportion equal to 1:2, splitting the data into 75% training and 25% validating sets. *Accuracy* measures the portion of all testing samples classified correctly. *Recall* measures the ability of a classifier to correctly identify positive labels. And *precision* measures the proportion of all correctly identified samples in a population of samples which are classified as positive labels. $F1 - score$ combines *precision*, *recall* into a single metric, $F1 - score = (2 * Recall * Precision)/(Recall * Precision)$.

	Balanced RF (1)			XGBoost (500) (2)			XGBoost (10) (3)		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
False	0.85	0.86	0.86	0.83	0.91	0.87	0.79	0.86	0.82
True	0.72	0.70	0.71	0.77	0.63	0.70	0.66	0.55	0.60
Accuracy		0.81			0.81			0.75	

5. A real-time spoofing probability measure

Results from the previous sections, suggest that the XGBoost and the Random Forest are the best performing models in term of Precision and Recall respectively. Furthermore cross-validation robustness checks show these two algorithms emerge as the dominant ones in different cross validation approaches.

This section introduces a new forecasting measure, the Real-Time Spoofing Probability (RTSP), based on the combination of the XGBoost and Random Forest algorithms. The RTSP aims at providing a real-time indicator of the risk of interacting with a spoofer. Unlike liquidity measures such as effective or quoted bid-ask spreads, a spoofing manipulation measure is not easy to identify. Manipulators usually hide behind retail trades' order flow to avoid being detected. Empirical measures are typically motivated by theoretical studies of spoofing trading to overcome these limitations. For example, Cartea et al. (2020) model the trading strategy of an investor who spoofs the limit order book (LOB) and computes the LOB volume imbalance as a measure that is important for spoofing strategy to be profitable. We propose a data-driven approach to measuring trading risk when spoofing manipulation is highly probable.

To test the forecasting performance of the RSTP measure on out-of-sample data, we train models on five consecutive trading days and the current trading day before 14:00. As the lifetime of spoofing order is less then a minute according to Table 1. Moreover, taking into account that features in our model are based on limit order book information, that changes quickly, we need to add new information to the training set often. So, we forecast spoofing orders for the next 10, 30, and 60 minutes from 14:00 to 18:00. We forecast each market state or tick. We keep the variety of 10, 30, and 60 minutes forecast for the robustness of our findings. For example, in a 10-minute setting, we train data on the previous five trading days and today's morning, then forecast spoofing risk for each tick, and every 10 minutes, we retrain the model with new data. The model outcomes ranges from zero to one and represents the probability of the spoofing order occurring in the next tick. So, the RTSP is higher when spoofing trading is more likely. We do not use rounding to compute RTSP measure . We train models on the balanced dataset with a 1:1 True-False orders proportion. Then we predict the imbalanced dataset with a 1:2 True-False orders proportion as it is closer to the actual trading environment. We use two validation approaches where we add more data to train the dataset. Fig. 3 shows the logic of the expanding validation approach when we add more data for the training period while keeping the testing period 10, 30 or 60 minutes. Moreover, we also run a rolling validation approach when we keep the training period the same for all the splits (Fig.4). We show the results of both approaches in Table 13. We forecast 70,347 orders including of 19,971 True and 50,376 False suspect spoofing orders. Table 13 shows the forecasting results of the three best performing models and RTSP measures with the two validation approaches illustrated in Fig. 3 and Fig. 4.

Overall, we do not observe significant differences in predictive quality across
10-, 30- and 60-minute rolling forward predictions. So, whether we re-estimate our model every 10

Table 13: Forecasting performance of RTSP. The table shows the results of ML models and the RTSP measure with 10-, 30- and 60-minute forecasting re-estimation settings, where column *All correct* shows the percentage of correctly predicted all market states, while columns *True correct* and *False correct* show the percentage of correctly predicted market states with True and False spoofing orders respectively. The first number shows the result using the expanding validation approach (Fig. 3) when we add new data to the training dataset while keeping the testing time frame the same (10, 30, or 60 minutes). The second number after the slash shows the results using the rolling validation approach illustrated in Fig. 4.

10-minute re-estimation frequency			
Model	All correct (%)	True correct (%)	False correct (%)
XGBoost	67.2 / 66.4	74.4 / 46.9	75.0 / 75.3
Random Forest	68.7 / 68.8	22.2 / 22.1	87.1 / 87.3
Decision Tree	65.0 / 64.5	49.5 / 47.4	71.1 / 71.3
RTSP	68.2 / 68.3	42.5 / 41.9	78.4 / 78.8
30-minute re-estimation frequency			
Model	All correct (%)	True correct (%)	False correct (%)
XGBoost	66.8 / 66.7	47.1 / 46.1	74.6 / 74.8
Random Forest	68.2 / 68.5	21.4 / 21.9	86.8 / 86.9
Decision Tree	65.3 / 64.6	50.2 / 48.7	71.3 / 70.9
RTSP	68.1 / 68.3	42.7 / 42.4	78.1 / 78.6
60-minute re-estimation frequency			
Model	All correct (%)	True correct (%)	False correct (%)
XGBoost	66.3 / 66.4	47.0 / 46.2	74.0 / 74.4
Random Forest	67.9 / 68.2	21.3 / 21.9	86.4 / 86.5
Decision Tree	65.0 / 64.3	50.1 / 48.2	70.9 / 70.7
RTSP	67.9 / 67.8	42.7 / 42.1	77.9 / 78.0

minutes or every hour does not significantly improve the results. XGBoost and Decision Tree models predict around 50% of True and 70-75% of False spoofing cases correctly with an overall predictive quality of approximately 67%. The Random Forest predicts only 20-25% of True cases; however, 87% of False cases, with overall approximately 68% events predicted correctly. The RTSP measure overperfroms all three alogorithms. Different forecasting validation approaches show similar results.

Additionally, we test our method on buy and sell suspect spoofing orders separately. Table 14 shows that forecasting accuracy for market states with buy spoofing orders is slightly better. This example indicates that one could train models differently depending on the required task; for example, the exchange wants to know the risk of spoofing occurrence only from the bidding side of the limit order book.

The RTSP shows a probability in real time that the market state is preferable for the spoofers to place their order. Out-sample forecasting reveals that a forecast combination is better than

Table 14: The table shows the performance of ML models and the RTSP measure for buy and sell spoofing risk with 10-, 30- and 60-minute forecasting re-estimation settings. The column *All Buy* shows the percentage of correctly predicted market states with buy spoofing orders, while columns *True Buy* and *False Buy* show the percentage of correctly predicted market states with True and False buy orders, respectively. The same logic is applied for forecasting market state with sell spoofing orders.

10-minute re-estimation frequency						
Model	All Buy (%)	True Buy (%)	False Buy (%)	All Sell (%)	True Sell (%)	False Sell (%)
XGBoost	70.4	40.9	80.2	67.9	40.3	77.0
Random Forest	72.2	24.1	88.2	70.5	22.7	86.3
Decision Tree	65.9	43.5	73.2	63.6	45.2	69.7
RTSP	70.5	37.6	81.4	68.4	37.9	78.5

30-minute re-estimation frequency						
Model	All Buy (%)	True Buy (%)	False Buy (%)	All Sell (%)	True Sell (%)	False Sell (%)
XGBoost	70.6	41.6	79.8	67.7	41.1	76.6
Random Forest	72.1	24.1	87.9	70.4	22.8	86.2
Decision Tree	65.6	42.4	73.3	63.6	45.5	69.6
RTSP	70.2	37.0	81.2	68.3	38.1	78.2

60-minute re-estimation frequency						
Model	All Buy (%)	True Buy (%)	False Buy (%)	All Sell (%)	True Sell (%)	False Sell (%)
XGBoost	70.3	41.9	79.6	67.6	41.3	76.4
Random Forest	72.0	24.0	87.8	65.1	22.9	85.9
Decision Tree	65.0	42.9	72.3	63.2	45.7	69.0
RTSP	70.0	37.6	80.7	68.2	38.7	77.9

an individual forecast. Table 13 shows that Random Forest slightly overperforms RTSP measure while assessing all spoofing and non-spoofing market states forecast. However, for example, in 10-minute re-estimation frequency, we observe RTSP measure significantly overperforming in forecasting "True" spoofing orders, which is a high priority for the regulator and traders as final users and beneficiaries of the developed algorithm. The aim of the machine learning application is equally important to forecast the market state when spoofing is likely to occur and the market state which is safe to trade without manipulative orders. RTSP smoothes the prediction of the individual machine learning algorithms, making the forecasting outcomes more reliable.

When comparing the forecasting performance of RTSP to the performance of other ML algorithms, we observe that our designed methodology performs better in the given environment. Table 15 shows the forecasting power of alternative models, and we observe worse results than those from the RTSP measure.

Finally, we illustrate one of many possible applications that could be developed using the RTSP

Table 15: Forecasting results of alternative models. The table shows 60-minute forecasting results using alternative models, where column *All correct* shows the percentage of correctly predicted all orders, while columns *True correct* and *False correct* show the percentage of correctly predicted True and False orders, respectively.

Model	All correct (%)	True correct (%)	False correct (%)
Logistic Regression	57.7	37.4	65.7
Logistic Regression (Lasso)	57.9	37.6	65.9
Logistic Regression (Ridge)	57.7	37.4	65.7
K-nearest neighbors	58.5	38.2	66.5
Stochastic Gradient Descent	53.9	43.8	57.9
Support Vector Machines	29.9	17.5	34.8
LSTM with 10 layers	54.6	44.0	58.9
GRU	61.4	25.9	75.5
Percepton	46.6	55.4	43.1

measure, as a real-time risk indicator for authorities or exchanges showing the market conditions when spoofing orders may appear in the order book leading to a worse market state for the order execution. Fig. 1 shows an example of the graphical spoofing risk indication. We set a critical value for the RTSP measure as 0.4 and marked it as a horizontal black dotted line on the bottom chart. We observe that RTSP rises above the critical value at the end of the trading day around 18:30. On the top chart, we see that many True suspicious spoofing orders appear at that time, which we mark as green dotted lines. Another example on the graph is the time between 17:30 and 17:45 when RTSP significantly rises above 0.4 and correctly forecasts the spoofing order placement. The user may change the critical value parameter and adjust the signal appearance as a number, warning, or trading feature on the chosen platform. So, the RTSP measure could be incorporated into trading platforms or surveillance systems. The model is self-training; no adjustments are needed to detect other manipulative practices.

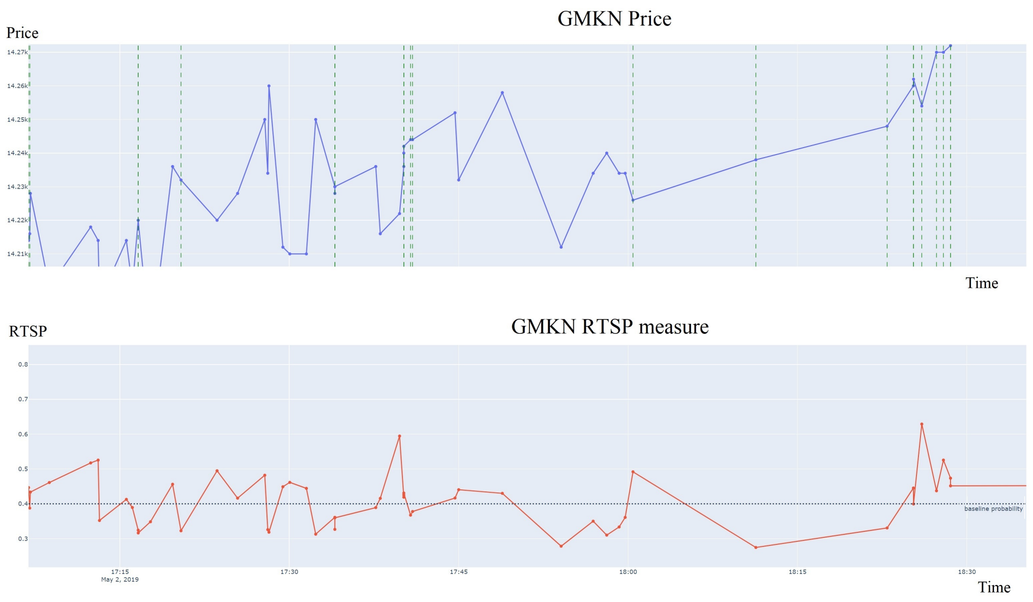

Figure 1: RTSP implementation. The figure illistates an example of RTSP measure signals on GMKN stock. The vertical green dotted lines on the top chart indicate when True spoofing orders happened. The horizontal black dotted line on the bottom chart indicates a critical value for the RTSP measure of 0.4.

6. Conclusions

In this paper, we introduce a data-driven methodology to identify and forecast market states associated with spoofing events in real time. Our approach is holistic, ranging from the selection of the most significant order book variables, which can predict spoofing events, to the choice of the appropriate machine learning algorithms that aggregate all critical information into a single score, the Real-Time Spoofing Probability (RTSP) measure, designed to signify the market vulnerability to intra-day manipulative activity. State-of-the-art ML techniques help us learn from the data and find how the Limit order book when suspected spoofers trade differ from the Limit Order book when they do not trade. This flexible approach accounts for non-linearities and interactions between variables, while cross-validation and regularization avoid overfitting the data. Our analysis provides strong evidence of the model appropriateness for imbalanced datasets, such as the real-time tick data used in our empirical study. This performance consistency implies a much stronger generalization capacity compared to the state-of-the-art models, which renders our approach much more attractive to researchers and regulators working in financial institutions.

The main contributions of this empirical study and its stark differences from other studies in the related literature can be summarized in three layers. First, to the best of our knowledge, this study is the first to extensively implement a broad selection of the most widely applied ML methods and compare their predictive performance to forecast periods with a potential of suspicious activities. Second, our approach provides a comprehensive characterization of the limit order book state by identifying the relevant financial microstructure indicators and establishes its pivotal role in detecting fraudulent trading activities. Third, we focus on suspected spoofing incidents recognized by exchange authorities, ensuring replication of their spoofing identification methodology, albeit with a more constrained data sets.

Our approach facilitates the identification of intervals featuring potentially suspicious and fraudulent activities within the order book. Additionally, by leveraging our methodology, exchanges can proactively pinpoint periods with elevated risk of fraudulent activity, drawing from insights on past suspicious orders, typically identified retrospectively through comprehensive analysis of traders' IDs and their historical trading strategies. By integrating our method into their surveillance systems, exchanges stand to achieve substantial enhancements in market quality, coupled with a notable reduction in instances of market manipulation.

One aspect that this work does not consider is whether allowing for our model to account for market shocks, news, dividend activities, or macroeconomic variables can improve prediction performance. In the future, we aim to perform our analysis on an enriched dataset with high frequency tick data and lower frequency macro-level data using a MIDAS approach.In addition, our framework does not consider sequential aggressive spoofing orders. Nevertheless, the results of this analysis provide valuable information to regulators and exchanges to mitigate mitigate artificial, manipulative orders, improving the overall market quality.

7. Appendices

7.1. Appendix 1. Regularization for variables choice

Table 16: Lasso regularization for variables choice. This table shows the results of the lasso regularization technique. We remove from the analysis those predictors that have the value of the coefficient less than 0.001 (column *Coef.*).

Feature	**Coef**	**Std.Err.**	z	**P>** $\lvert z\rvert$	**[0.025**	**0.975]**
IMB	-0.3212	0.10522	-3.05267	0.002268	-0.52743	-0.11497
IMB_order	-0.00843	0.027257	-0.30911	0.75724	-0.06185	0.044997
FRA	-1.60787	0.144537	-11.1243	9.56E-29	-1.89116	-1.32458
FRB	0.880062	0.14531	6.056448	1.39E-09	0.595259	1.164864
QS	383.8753	34.00574	11.28854	1.49E-29	317.2252	450.5253
ES	122.714	49.63106	2.472524	0.013416	25.43888	219.9891
IMB_0	0.006483	0.014956	0.433459	0.664682	-0.02283	0.035795
IMB_1	0.097356	0.014714	6.616393	3.68E-11	0.068516	0.126196
IMB_2	0.077485	0.015252	5.080211	3.77E-07	0.047591	0.107378
IMB_3	0.078047	0.015426	5.059438	4.20E-07	0.047813	0.108282
IMB_4	-0.07434	0.015573	-4.77395	1.81E-06	-0.10487	-0.04382
IMB_5	-0.03868	0.01498	-2.58226	0.009816	-0.06804	-0.00932
UF_1ms_1	0.000422	0.000366	1.155149	0.24803	-0.00029	0.001139
UF_1ms_2	-0.00073	0.000496	-1.46944	0.141714	-0.0017	0.000243
UF_1ms_3	0.000657	0.000566	1.159823	0.246121	-0.00045	0.001766
UF_1ms_4	0.001032	0.000529	1.950864	0.051073	-4.80E-06	0.002068
UF_1ms_5	-0.00035	0.000502	-0.69271	0.48849	-0.00133	0.000636
UF_1ms_6	0.000565	0.000578	0.976952	0.328593	-0.00057	0.001698
UF_1ms_7	3.50E-05	0.000778	0.044997	0.96411	-0.00149	0.001559
UF_1ms_8	-0.00119	0.000701	-1.70177	0.088799	-0.00257	0.000181
UF_1ms_9	0.001496	0.000532	2.814629	0.004883	0.000454	0.002538
UF_1ms_10	-0.00107	0.000321	-3.32851	0.000873	-0.0017	-0.00044
UF_10ms_1	-0.00077	0.000741	-1.03415	0.301067	-0.00222	0.000686
UF_10ms_2	0.001162	0.000977	1.189622	0.234195	-0.00075	0.003076
UF_10ms_3	0.001405	0.001069	1.314504	0.188677	-0.00069	0.0035
UF_10ms_4	-0.00322	0.001053	-3.05752	0.002232	-0.00528	-0.00116
UF_10ms_5	0.003332	0.001106	3.012135	0.002594	0.001164	0.0055
UF_10ms_6	-0.00368	0.001174	-3.13422	0.001723	-0.00598	-0.00138
UF_10ms_7	0.002428	0.001261	1.924943	0.054237	-4.40E-05	0.0049
UF_10ms_8	0.001425	0.001161	1.227215	0.219742	-0.00085	0.0037
UF_10ms_9	-0.00373	0.000965	-3.86287	0.000112	-0.00562	-0.00184
UF_10ms_10	0.002529	0.000562	4.500931	6.77E-06	0.001428	0.00363
UF_50ms_1	0.007226	0.0018	4.015637	5.93E-05	0.003699	0.010754
UF_50ms_2	-0.00568	0.002153	-2.63767	0.008348	-0.0099	-0.00146
UF_50ms_3	0.000266	0.002007	0.132356	0.894703	-0.00367	0.004199
UF_50ms_4	0.000947	0.002137	0.443233	0.657597	-0.00324	0.005137
UF_50ms_5	-0.00443	0.00232	-1.90867	0.056305	-0.00897	0.000119
UF_50ms_6	0.01228	0.00243	5.053454	4.34E-07	0.007518	0.017043
UF_50ms_7	-0.0053	0.002431	-2.1814	0.029154	-0.01007	-0.00054
UF_50ms_8	-0.00452	0.001985	-2.27897	0.022669	-0.00841	-0.00063
UF_50ms_9	0.00126	0.001604	0.785475	0.432175	-0.00188	0.004405
UF_50ms_10	-0.001	0.001095	-0.91578	0.359781	-0.00315	0.001143
UF_100ms_1	-0.00551	0.001667	-3.30508	0.000949	-0.00878	-0.00224
UF_100ms_2	0.004442	0.002013	2.20594	0.027388	0.000495	0.008388
UF_100ms_3	-0.00302	0.001863	-1.61985	0.105264	-0.00667	0.000634
UF_100ms_4	0.002616	0.001967	1.330229	0.183443	-0.00124	0.006471
UF_100ms_5	0.000583	0.002108	0.276438	0.782211	-0.00355	0.004714
UF_100ms_6	-0.00897	0.002183	-4.11074	3.94E-05	-0.01325	-0.00469
UF_100ms_7	0.004389	0.002155	2.036936	0.041656	0.000166	0.008612
UF_100ms_8	0.001557	0.001819	0.856017	0.391988	-0.00201	0.005123
UF_100ms_9	0.004167	0.00153	2.722724	0.006475	0.001167	0.007166
UF_100ms_10	-0.0023	0.001034	-2.22182	0.026296	-0.00432	-0.00027
UF_600ms_1	-0.00046	0.000486	-0.93692	0.348798	-0.00141	0.000497

UF_600ms_2	0.000218	0.000583	0.37335	0.708888	-0.00092	0.00136
UF_600ms_3	0.001369	0.000599	2.283564	0.022397	0.000194	0.002544
UF_600ms_4	-0.00127	0.000594	-2.13249	0.032967	-0.00243	-0.0001
UF_600ms_5	0.000823	0.000608	1.35409	0.175708	-0.00037	0.002015
UF_600ms_6	0.000205	0.00062	0.330751	0.740833	-0.00101	0.001421
UF_600ms_7	-0.00108	0.000626	-1.7199	0.08545	-0.0023	0.00015
UF_600ms_8	0.001556	0.000608	2.559037	0.010496	0.000364	0.002748
UF_600ms_9	-0.00257	0.000598	-4.30355	1.68E-05	-0.00374	-0.0014
UF_600ms_10	0.00148	0.000354	4.175298	2.98E-05	0.000785	0.002175
VOL_1_1min	-332.465	112.7972	-2.94746	0.003204	-553.544	-111.387
VOL_1_2min	67.88061	179.9601	0.377198	0.706026	-284.835	420.596
VOL_1_5min	-55.8454	271.0023	-0.20607	0.836736	-587	475.3093
VOL_1_10min	36.52509	286.0571	0.127685	0.898399	-524.137	597.1868
VOL_2_1min	6.095202	1.104971	5.516166	3.46E-08	3.929499	8.260905
VOL_2_2min	-5.96658	1.590235	-3.75201	0.000175	-9.08338	-2.84978
VOL_2_5min	-4.02738	2.149298	-1.87381	0.060956	-8.23993	0.185166
VOL_2_10min	19.84205	2.331745	8.509528	1.75E-17	15.27192	24.41219
VOL_4_1min	2.754861	6194.337	0.000445	0.999645	-12137.9	12143.43
VOL_4_2min	2.870529	10254.95	0.00028	0.999777	-20096.5	20102.19
VOL_4_5min	2.577828	13808.9	0.000187	0.999851	-27062.4	27067.52
VOL_4_10min	3.352821	11427.39	0.000293	0.999766	-22393.9	22400.62
Hour	0.019725	0.002944	6.700296	2.08E-11	0.013955	0.025495
DistNorm50	20.25476	4455.851	0.004546	0.996373	-8713.05	8753.562
DistNorm20	-199.294	20273.16	-0.00983	0.992157	-39934	39535.38
DistNorm10	-194.514	18044.84	-0.01078	0.991399	-35561.7	35172.72
DistNormClean50	149.5656	27.31266	5.476056	4.35E-08	96.03381	203.0975
DistNormClean20	-176.587	31.28896	-5.64374	1.66E-08	-237.912	-115.262
DistNormClean10	93.17991	48.21986	1.932397	0.053311	-1.32927	187.6891
DistNormVol50	0.553286	0.078023	7.091356	1.33E-12	0.400364	0.706207
DistNormVol20	0.089309	0.030394	2.938428	0.003299	0.029739	0.14888
DistNormVol10	-0.01351	0.018983	-0.71187	0.476548	-0.05072	0.023692
DistNormVolClean50	-0.61615	0.09366	-6.57852	4.75E-11	-0.79972	-0.43258
DistNormVolClean20	-0.05694	0.036533	-1.55872	0.119063	-0.12855	0.014659
DistNormVolClean10	-0.08247	0.025564	-3.22598	0.001255	-0.13257	-0.03236
IMB_order_delta	-0.25393	0.188881	-1.3444	0.178818	-0.62413	0.116268
FRA_delta	-0.04651	0.375481	-0.12387	0.901414	-0.78244	0.689416
FRB_delta	-0.97608	0.377774	-2.58377	0.009773	-1.71651	-0.23566
QS_delta	70.65095	65.19404	1.083702	0.278497	-57.127	198.4289
IMB_order_delta_t2	0.366445	0.155094	2.362738	0.01814	0.062468	0.670423
FRA_delta_t2	-0.08441	0.315573	-0.26749	0.789091	-0.70293	0.534099
FRB_delta_t2	-0.36953	0.31831	-1.16091	0.245679	-0.99341	0.254347
QS_delta_t2	45.82324	57.36325	0.798826	0.424391	-66.6067	158.2531

Table 17: Elastic net regularization for variables choice. This table shows the results of the lasso regularization technique with an alpha 50%. We remove from the analysis those predictors that have the value of the coefficient equals to zero (column *Coef.*).

Features	Coef.	Features	Coef.
IMB	-0.0000020571	UF_100ms_7	-0.0000346457
IMB_order	0.0000011417	UF_100ms_8	-0.0000283937
FRA	-0.0000063929	UF_100ms_9	0.0000230656
FRB	-0.0000061548	UF_100ms_10	-0.0000084160
QS	0.0000000000	UF_600ms_1	-0.0000389521
ES	0.0000000000	UF_600ms_2	0.0000076768
IMB_0	0.0000007831	UF_600ms_3	0.0000759581
IMB_1	0.0000050115	UF_600ms_4	0.0000072271
IMB_2	0.0000032278	UF_600ms_5	-0.0001139139
IMB_3	0.0000030746	UF_600ms_6	-0.0000864398
IMB_4	-0.0000034290	UF_600ms_7	-0.0000397455
IMB_5	-0.0000029741	UF_600ms_8	0.0001119282
UF_1ms_1	0.0002037987	UF_600ms_9	0.0001552371
UF_1ms_2	0.0001268850	UF_600ms_10	0.0002367776
UF_1ms_3	0.0002629904	VOL_1_1min	0.0000000000
UF_1ms_4	0.0002236545	VOL_1_2min	0.0000000000
UF_1ms_5	0.0002090883	VOL_1_5min	0.0000000000
UF_1ms_6	0.0001877748	VOL_1_10min	0.0000000000
UF_1ms_7	0.0001086263	VOL_2_1min	0.0000001040
UF_1ms_8	-0.0000996961	VOL_2_2min	0.0000000555
UF_1ms_9	-0.0001278121	VOL_2_5min	0.0000001025
UF_1ms_10	-0.0002366476	VOL_2_10min	0.0000001424
UF_10ms_1	0.0001470780	VOL_4_1min	0.0000000000
UF_10ms_2	0.0000016729	VOL_4_2min	0.0000000000
UF_10ms_3	0.0000927591	VOL_4_5min	0.0000000000
UF_10ms_4	0.0000683550	VOL_4_10min	0.0000000000
UF_10ms_5	0.0001042563	Hour	-0.0000807383
UF_10ms_6	0.0001729422	DistNorm50	-0.0000000461
UF_10ms_7	0.0001453083	DistNorm20	-0.0000000167
UF_10ms_8	0.0000134912	DistNorm10	-0.0000000069
UF_10ms_9	0.0000455122	DistNormClean50	-0.0000000466
UF_10ms_10	0.0000721976	DistNormClean20	-0.0000000175
UF_50ms_1	0.0000791259	DistNormClean10	-0.0000000077
UF_50ms_2	-0.0000835425	DistNormVol50	0.0000051493
UF_50ms_3	-0.0000380171	DistNormVol20	0.0000037754
UF_50ms_4	-0.0000731352	DistNormVol10	0.0000003867
UF_50ms_5	-0.0001232195	DistNormVolClean50	0.0000038100
UF_50ms_6	-0.0000435570	DistNormVolClean20	0.0000026168
UF_50ms_7	-0.0000991683	DistNormVolClean10	-0.0000005011
UF_50ms_8	-0.0001743680	IMB_order_delta	-0.0000000849
UF_50ms_9	-0.0001484642	FRA_delta	0.0000000824
UF_50ms_10	-0.0001885770	FRB_delta	-0.0000001857
UF_100ms_1	0.0000356753	QS_delta	0.0000000000
UF_100ms_2	-0.0000570796	IMB_order_delta_t2	0.0000001432
UF_100ms_3	-0.0000136687	FRA_delta_t2	0.0000000255
UF_100ms_4	-0.0000411564	FRB_delta_t2	-0.0000001150
UF_100ms_5	-0.0001139238	QS_delta_t2	0.0000000000
UF_100ms_6	-0.0000416980		

7.2. Appendix 2. Correlation matrix between predictors

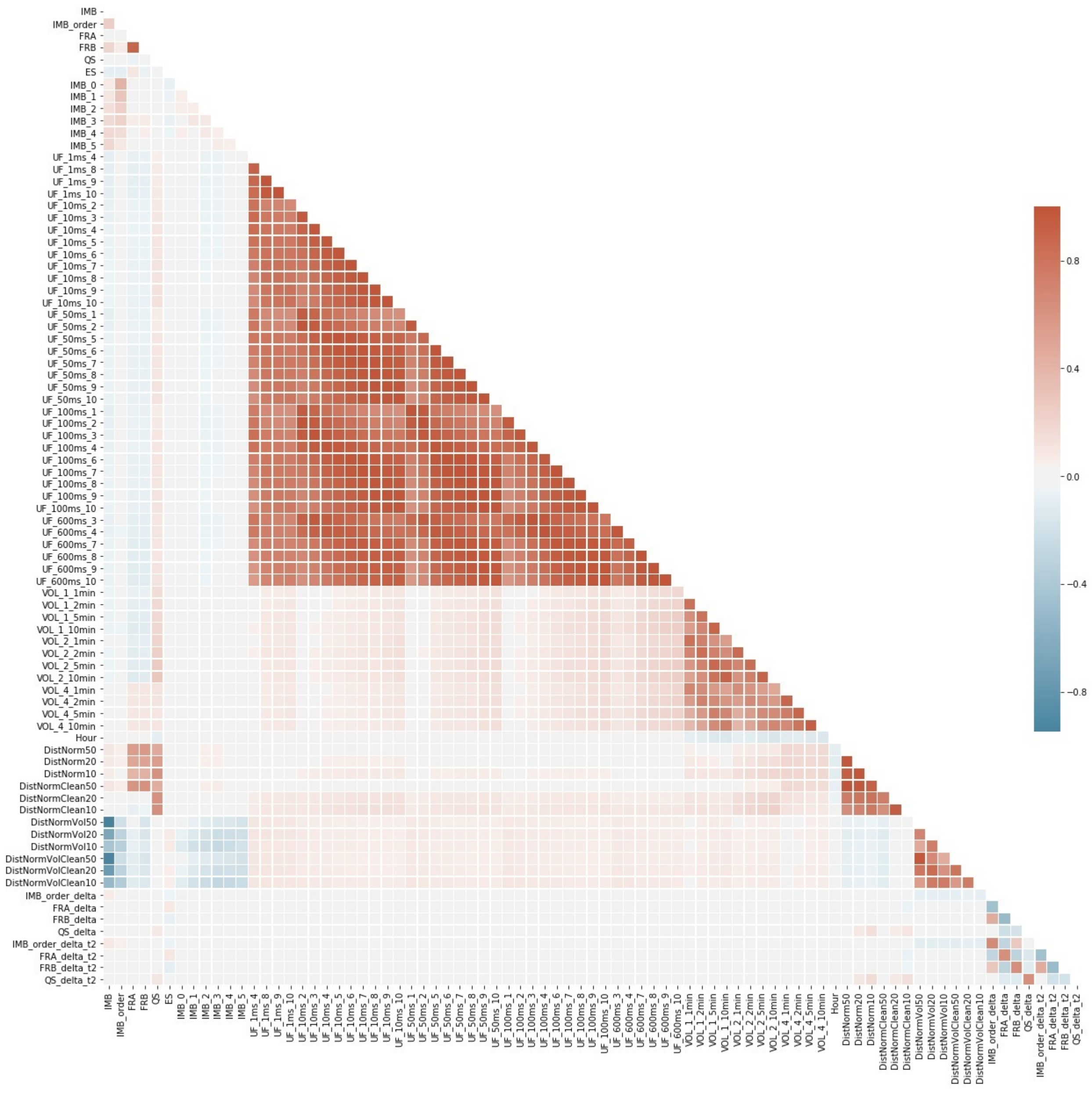

Figure 2: Correlation matrix between predictors

7.3. Appendix 3. List of features

We remove from the further analysis unnecessary features using the elastic net (Enet) variable selection method with coefficients equal to 0. Unnecessary features are the following: QS, ES, VOL_1_1min, VOL_1_2min, VOL_1_5min, VOL_1_10min, VOL_3_1min, VOL_3_2min, VOL_3_5min, VOL_3_10min, QS_delta, QS_delta_t2. Variable 'Hour' leads to over-fitting of the model due to the absence of False orders in the first and last trading hours. 'VOL_2_1min' variable has many unidentified values. Therefore, we remove both features 'Hour' and 'VOL_2_1min'.

The final selection of predictors for further analysis is the following: IMB, IMB_order, FRA, FRB, IMB_0, IMB_1, IMB_2, IMB_3, IMB_4, IMB_5, UF_1ms_4, UF_1ms_8, UF_1ms_9, UF_1ms_10, UF_10ms_2, UF_10ms_3, UF_10ms_4, UF_10ms_5, UF_10ms_6, UF_10ms_7, UF_10ms_8, UF_10ms_9, UF_10ms_10, UF_50ms_1, UF_50ms_2, UF_50ms_5, UF_50ms_6, UF_50ms_7, UF_50ms_8, UF_50ms_9, UF_50ms_10, UF_100ms_1, UF_100ms_2, UF_100ms_3, UF_100ms_4, UF_100ms_6, UF_100ms_7, UF_100ms_8, UF_100ms_9, UF_100ms_10, UF_600ms_3, UF_600ms_4, UF_600ms_7, UF_600ms_8, UF_600ms_9, UF_600ms_10, VOL_2_2min, VOL_2_5min, VOL_2_10min, DistNorm50, DistNorm20, DistNorm10, DistNormClean50, DistNormClean20, DistNormClean10, DistVol50, DistVol20, DistVol10, DistVolClean50, DistVolClean20, DistVolClean10, IMB_delta, FRA_delta, FRB_delta, IMB_delta_t2, FRA_delta_t2, FRB_delta_t2.

7.4. Appendix 4. Diebold-Mariano test

Table 18: Diebold-Mariano test. The table shows the test result that compares the models' predictive qualities in pairs. We compare Random Forest, Decision Tree and XGBoost with other ML models. The null hypothesis is that models are equal. If the p-value is less than 0.05, then the models are not equal. We mark the results with the '-' sign if the first model is better than the second model; the '+' sign shows the opposite.

First model	Second model	Diebold-Mariano statistic	P-value
Random Forest	Logistic regression	-54.09	0.00
Random Forest	SVM	-48.59	0.00
Random Forest	KNN	-24.51	0.00
Random Forest	Naive Bayes	-62.61	0.00
Random Forest	SGD	-59.31	0.00
Random Forest	Decision Tree	-22.97	0.00
Random Forest	Gradient Boosting	-26.32	0.00
Random Forest	XGBoost	-26.81	0.00
Random Forest	LSTM with 10 layers	-50.05	0.00
Random Forest	GRU	-52.20	0.00
Random Forest	Percepton	-54.35	0.00
Decision Tree	Logistic regression	-37.09	0.00
Decision Tree	SVM	-26.91	0.00
Decision Tree	KNN	-1.83	0.07
Decision Tree	Naive Bayes	-37.16	0.00
Decision Tree	SGD	-36.24	0.00
Decision Tree	Gradient Boosting	-1.66	0.10
Decision Tree	XGBoost	-2.12	0.03
Decision Tree	LSTM with 10 layers	-33.51	0.00
Decision Tree	GRU	-37.10	0.00
Decision Tree	Percepton	-34.42	0.00
XGBoost	Logistic regression	-33.59	0.00
XGBoost	SVM	-25.31	0.00
XGBoost	KNN	0.25	0.80
XGBoost	Naive Bayes	-39.49	0.00
XGBoost	SGD	-36.87	0.00
XGBoost	Gradient Boosting	1.63	0.10
XGBoost	LSTM with 10 layers	-29.46	0.00
XGBoost	GRU	-32.16	0.00
XGBoost	Percepton	-33.03	0.00

7.5. Appendix 5. Model Confidence Set test

Table 19: Model Confidence Set test. The table shows the implementation of the Model Confidence Set (MCS) procedure for model comparison at the 95% confidence level (alfa is 0.05) and MAE (mean absolute error) loss function. Panel A shows that comparing all the models we use in the research, Random Forest is the best model. Panel B shows the results of the model's comparison test, excluding Random Forest, to find the second-best model.

Panel A

Model	Elimination result						
Logistic regression	eliminated						
SGD	eliminated						
GRU	eliminated						
Naive Bayes	eliminated						
LSTM with 10 layers	eliminated						
SVM	eliminated						
XGBoost	eliminated						
Gradient Boosting	eliminated						
KNN	eliminated						
Random Forest	eliminated						
Superior Set Model created	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
Random Forest	1	-28.06988	1	1	-28.06988	1	0.2332108
p-value : [1] 0							

Panel B

Model	Elimination result						
Logistic regression	eliminated						
Naive Bayes	eliminated						
GRU	eliminated						
SGD	eliminated						
LSTM with 10 layers	eliminated						
SVM	eliminated						
Superior Set Model created	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
KNN	3	0.6730877	0.769	3	1.886552	0.212	0.3200074
Decision Tree	1	-2.3702309	1.000	1	-1.876958	1.000	0.3126542
Gradient Boosting	2	0.6042045	0.813	2	1.876958	0.223	0.3192163
XGBoost	4	1.7018618	0.159	4	2.368780	0.081	0.3210779
p-value : [1] 0.159							

7.6. Appendix 5. Optimization of the parameters for ML models

To optimize the chosen models, we tune the parameters while training models on 75% of the data and testing on 25% of the data. For Random Forest, increasing 'max_depth' to 41 and 'n_estimators' to 600 improves the accuracy. Further increasing 'max_depth' does not improve the results, and an increase of 'n_estimators' to 1000 improves the accuracy by 0.5%; however, it dramatically raises the complexity of the models, causing a decrease in model performance. For XGBoost, we choose 'n_estimators' of 500, as the increase to 1000 adds only 1.5% to the model's accuracy, raising the complexity and decreasing the performance speed and decreasing 'n_estimators' to 100 results in a significant drop of the accuracy by 5%. Decreasing 'max_depth' below 10 results in a drop of the accuracy. For Decision Tree, 'max_depth' over 5 does not yield the results; also, increasing 'min_samples_leaf' over 3 does not improve results either.

So, we choose the following parameters for selected machine learning models.

Random Forest parameters:

$clf = RandomForestClassifier\ (random\_state = 0$
$n\_estimators = 600,\ min\_samples\_split = 5,$
$min\_samples\_leaf = 1,\ max\_features =' sqrt',$
$max\_depth = 41,\ bootstrap = False)$

XGBoost parameters:

$clf = xgb.XGBClassifier\ (learning\_rate = 0.01,$
$n\_estimators = 500,$
$max\_depth = 10,$
$min\_child\_weight = 1,$
$gamma = 0.1,$
$subsample = 0.8,$
$colsample\_bytree = 0.8,$
$objective =' binary : logistic',$
$nthread = 4,$
$scale_pos_weight = 1.5,$random_state $= 0)$

Decision Tree parameters:

$clf = DecisionTreeClassifier$
$(random\_state = 0, max\_depth = 5, min\_samples\_leaf = 3)$

7.7. Appendix 6. Expanding and rolling validation

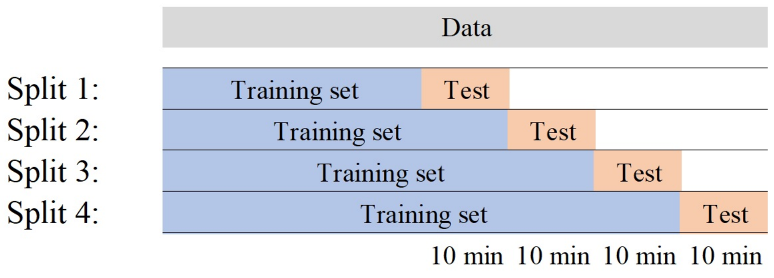

Figure 3: Illustration of the expanding validation approach

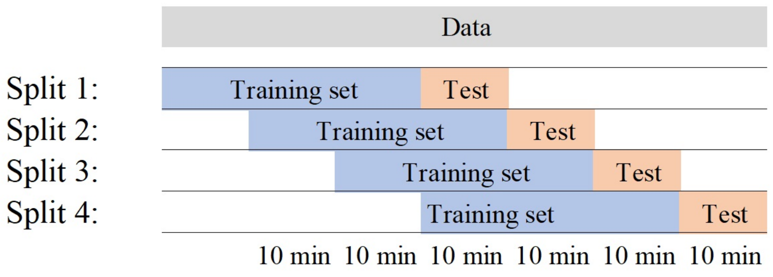

Figure 4: Illustration of the rolling validation approach

References

L. Guiso, P. Sapienza, L. Zingales, Trusting the stock market, the Journal of Finance 63 (2008) 2557–2600.

S. Imisiker, B. Tas, Which firms are more prone to stock market manipulation?, Emerging Markets Review 16 (2013) 119–130.

M. Punniyamoorthy, J. J. Thoppan, Ann-ga based model for stock market surveillance, Journal of Financial Crime (2013).

Dodd-Frank, Dodd-frank wall street reform and consumer protection act, 2010.

D. Cumming, S. Johan, D. Li, Exchange trading rules and stock market liquidity, Journal of Financial Economics 99 (2011) 651–671.

M. J. Aitken, F. H. de B Harris, S. Ji, A worldwide examination of exchange market quality: Greater integrity increases market efficiency, Journal of Business Ethics 132 (2015) 147–170.

H. Öğüt, M. M. Doğanay, R. Aktaş, Detecting stock-price manipulation in an emerging market: The case of turkey, Expert Systems with Applications 36 (2009) 11944–11949.

K. Golmohammadi, O. R. Zaiane, Time series contextual anomaly detection for detecting market manipulation in stock market, in: 2015 IEEE international conference on data science and advanced analytics (DSAA), IEEE, 2015, pp. 1–10.

Y. Cao, Y. Li, S. Coleman, A. Belatreche, T. M. McGinnity, Detecting price manipulation in the financial market, in: 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), IEEE, 2014, pp. 77–84.

Y. Cao, Y. Li, S. Coleman, A. Belatreche, T. M. McGinnity, Adaptive hidden markov model with anomaly states for price manipulation detection, IEEE transactions on neural networks and learning systems 26 (2015) 318–330.

J.-N. Tuccella, P. Nadler, O. Şerban, Protecting retail investors from order book spoofing using a gru-based detection model, arXiv:2110.03687 (2021).

X. Tao, A. Day, L. Ling, S. Drapeau, On detecting spoofing strategies in high-frequency trading, Quantitative Finance 22 (2022) 1405–1425.

B. Williams, A. Skrzypacz, Spoofing in equilibrium, Available at SSRN (2020).

Á. Cartea, R. Payne, J. Penalva, M. Tapia, Ultra-fast activity and intraday market quality, Journal of Banking & Finance 99 (2019) 157–181.

C. M. Lee, Market integration and price execution for nyse-listed securities, The Journal of Finance 48 (1993) 1009–1038.

M. E. Blume, M. A. Goldstein, Displayed and effective spreads by market, Rodney L. White Center for Financial Research Working Paper (1992).

M. Aitken, A. Frino, The determinants of market bid ask spreads on the australian stock exchange: Cross-sectional analysis, Accounting & Finance 36 (1996) 51–63.

J. F. Egginton, B. F. Van Ness, R. A. Van Ness, Quote stuffing, Financial Management 45 (2016) 583–608.

J. Hasbrouck, G. Saar, Low-latency trading, Journal of Financial Markets 16 (2013) 646–679.

A. Cartea, S. Jaimungal, Y. Wang, Spoofing and price manipulation in order-driven markets, Applied Mathematical Finance 27 (2020) 1–2, 67–98.

B. Biais, P. Hillion, C. Spatt, An empirical analysis of the limit order book and the order flow in the paris bourse, the Journal of Finance 50 (1995) 1655–1689.

A. Ranaldo, Order aggressiveness in limit order book markets, Journal of Financial Markets 7 (2004) 53–74.

Y. Cao, Y. Li, S. Coleman, A. Belatreche, T. McGinnity, Adaptive hidden markov model with anomaly states for price manipulation detection, IEEE Transactions on Neural Networks and Learning Systems 26 (2015) 318–330.

R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society: Series B (Methodological) 58 (1996) 267–288.

T. Hastie, R. Tibshirani, J. H. Friedman, J. H. Friedman, The elements of statistical learning: data mining, inference, and prediction, volume 2, Springer, 2009.

H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the royal statistical society: series B (statistical methodology) 67 (2005) 301–320.

S. Khodabandehlou, S. A. H. Golpayegani, Market manipulation detection: A systematic literature review, Expert Systems with Applications 210 (2022) 118330.

Z.-H. Zhou, Ensemble methods: foundations and algorithms, CRC press, 2012.

T. K. Ho, Random decision forests, in: Proceedings of 3rd international conference on document analysis and recognition, volume 1, IEEE, 1995, pp. 278–282.

L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

T. Hastie, R. Tibshirani, J. H. Friedman, J. H. Friedman, The elements of statistical learning: data mining, inference, and prediction, Springer, 2017.

T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, et al., Xgboost: extreme gradient boosting, R package version 0.4-2 1 (2015) 1–4.

G. Claeskens, J. R. Magnus, A. L. Vasnev, W. Wang, The forecast combination puzzle: A simple theoretical explanation, International Journal of Forecasting 32 (2016) 754–762.