econometrics

MDPI

Article

Detecting Pump-and-Dumps with Crypto-Assets: Dealing with Imbalanced Datasets and Insiders' Anticipated Purchases

Dean Fantazzini [1,2,*,†] **and Yufeng Xiao** [1,†]

[1] Moscow School of Economics, Moscow State University, 119992 Moscow, Russia
[2] Higher School of Economics, Faculty of Economic Sciences, 109028 Moscow, Russia
* Correspondence: fantazzini@mse-msu.ru
† These authors contributed equally to this work.

Abstract: Detecting pump-and-dump schemes involving cryptoassets with high-frequency data is challenging due to imbalanced datasets and the early occurrence of unusual trading volumes. To address these issues, we propose constructing synthetic balanced datasets using resampling methods and flagging a pump-and-dump from the moment of public announcement up to 60 min beforehand. We validated our proposals using data from Pumpolymp and the CryptoCurrency eXchange Trading Library to identify 351 pump signals relative to the Binance crypto exchange in 2021 and 2022. We found that the most effective approach was using the original imbalanced dataset with pump-and-dumps flagged 60 min in advance, together with a random forest model with data segmented into 30-s chunks and regressors computed with a moving window of 1 h. Our analysis revealed that a better balance between sensitivity and specificity could be achieved by simply selecting an appropriate probability threshold, such as setting the threshold close to the observed prevalence in the original dataset. Resampling methods were useful in some cases, but threshold-independent measures were not affected. Moreover, detecting pump-and-dumps in real-time involves high-dimensional data, and the use of resampling methods to build synthetic datasets can be time-consuming, making them less practical.

Keywords: pump-and-dump; crypto-assets; minority class; class imbalance; machine learning; random forests

Citation: Fantazzini, Dean, and Yufeng Xiao. 2023. Detecting Pump-and-Dumps with Crypto-Assets: Dealing with Imbalanced Datasets and Insiders' Anticipated Purchases. *Econometrics* 11: 22. https://doi.org/10.3390/econometrics11030022

Academic Editor: Massimiliano Caporin

Received: 11 May 2023
Revised: 20 July 2023
Accepted: 21 August 2023
Published: 30 August 2023

1. Introduction

A cryptocurrency is a form of digital currency that uses strong encryption and a ledger for protection. It allows users to buy goods and services online and is usually traded for speculative purposes. Blockchain technology is the backbone of a cryptocurrency, enabling it to function through a decentralized, distributed system that records transactions on multiple computers; see Antonopoulos (2014), Narayanan et al. (2016) for a technical discussion at the textbook level. At the time of writing this paper, there were over 10,000 publicly traded cryptocurrencies, with a total market value exceeding USD 2.2 trillion as of 1 March 2022, according to CoinMarketCap.com, accessed on 1 December 2022.

The unregulated nature of the cryptocurrency market has led to the emergence of pump-and-dump schemes, a traditional form of securities fraud. Fraudsters use social media platforms such as Telegram, Discord, and Twitter to initially spread false information and manipulate the price of low market-cap crypto assets. They then sell their holdings to make a profit, causing the price to fall and investors to lose money.

Kamps and Kleinberg (2018) and La Morgia et al. (2020, 2023) examined large sets of pumps-and-dumps and proposed several market-based approaches to detect them. Unfortunately, the detection of pump-and-dumps with crypto assets and high-frequency data involves imbalanced datasets where the minority class flagging the pump-and-dumps is very small (usually less than 0.2% of all cases). Moreover, even if the VIP members of the social media groups organizing these market manipulations are supposed to receive

detailed information about them seconds or minutes before the general public (and they pay a fee for such a service), in reality, unusual trading volumes can take place much earlier. These two problems can strongly affect the performance of any classifier used to flag potential pump-and-dump activity.

This study aims to solve these issues by building synthetic balanced datasets and enlarging the time sample flagging a pump-and-dump. The goal is to make it easier for investors to make informed decisions, and for financial regulators to prevent fraudulent activities in the cryptocurrency market.

To reach the paper's objective, we built synthetic balanced datasets generated according to four remedies for class imbalances proposed in the statistical literature. Moreover, we flagged a pump-and-dump from the minute of the public announcement up to 60 min before it. This simple solution was able to both deal with the insiders' anticipated purchases of the targeted crypto-assets and to solve the issue of imbalanced datasets. An empirical analysis with 351 pump-and-dumps that took place in 2021–2022 and a set of machine learning methods to detect these illegal activities confirmed the validity of the proposed solutions.

The paper is organized as follows: Section 2 reviews the literature devoted to cryptocurrency price manipulation and pump-and-dumps, while the methods proposed to detect pump-and-dump schemes are discussed in Section 3. The empirical results are reported in Section 4, while robustness checks are discussed in Section 5. Section 6 briefly concludes.

2. Literature Review

2.1. Pump-and-Dumps

In the last two decades, the financial literature has extensively examined "pump-and-dump" schemes, which are fraudulent practices involving the artificial inflation of stock prices through deceptive statements. These schemes aim to sell overvalued securities for a profit, leaving unsuspecting buyers with worthless investments. Researchers have focused on understanding the mechanics of these schemes, their impact on small retail investors, market efficiency, and the role of regulators in prevention.

Zaki et al. (2012) present a case study and propose architectural solutions to detect and prevent "pump-and-dump" schemes. They use simulations to demonstrate the manipulation's effects on market stability and investor confidence.

Withanawasam et al. (2013) develop a framework for market manipulation, specifically in a limit-order market. They show that "pump-and-dump" manipulation becomes profitable in the presence of technical traders, who exploit information asymmetry and generate profits by raising prices and selling immediately.

Bouraoui (2015) analyzes the impact of stock spam on stock prices and investor behavior. Despite investors' awareness of scams, the study finds that stock prices increase due to information asymmetry and the influence of spammers.

Siering (2019) examine the economics of "pump-and-dump" schemes in internet-based stock promotion. They analyze the schemes' financial and organizational structures, participants' incentives, and their effects on stock prices and investor behavior. The article emphasizes the need for increased regulation to protect investors and maintain market stability.

Ouyang and Cao (2020) investigate pump-and-dump events in the Chinese stock market. They find that smaller, less liquid stocks are more vulnerable to manipulation, and indicators such as market capitalization, free float, and liquidity provide insights into manipulators' allocation strategies.

Additionally, several academic and professional surveys provide general introductions to pump-and-dump schemes in financial markets; see US Security and Exchange Commission (2005), Krinklebine (2010), and Frieder and Zittrain (2008).

2.2. Cryptocurrency Pump-and-Dumps

The surge of cryptocurrencies in an unregulated environment has raised concerns about fraudulent activities, leading to a growing body of literature investigating pump-and-dump schemes in the cryptocurrency market. These schemes involve coordinated efforts to inflate the price of a specific cryptocurrency for personal profit. Social media platforms such as Telegram and Discord have been identified as common platforms for these schemes. Studies have explored the factors influencing the success of pump-and-dump schemes, the effectiveness of machine learning models in detecting and predicting these schemes, and the impact of such schemes on the cryptocurrency market.

Feder et al. (2018) examined pump-and-dump schemes on Telegram and Discord, identifying factors such as low market capitalization and liquidity that make certain cryptocurrencies more susceptible to targeting by pump groups. Similarly, Dhawan and Putniņš (2023) highlighted the correlation between low liquidity and higher abnormal returns in cryptocurrencies, indicating the significance of liquidity in these schemes. Victor and Hagemann (2019) compared pump-and-dump schemes in the cryptocurrency market to penny stock schemes and proposed the use of machine learning for detection.

Xu and Livshits (2019) investigated pump-and-dump events in cryptocurrencies, analyzing the fake trading volume and high profits obtained by organizers. Their machine learning models accurately predicted pump likelihoods, yielding significant returns. Using a similar approach, Shao (2021) employed several supervised learning models, including random forests, decision trees, and k-nearest neighbors, to detect pump-and-dump schemes in Dogecoin, with the random forest model being the most effective.

Hamrick et al. (2021) monitored Telegram channels and observed short-term price increases resulting from pump campaigns, but noted a decline in impact over time. They differentiated between transparent pumps, which provided clear information such as the coin name and target profit, and "obscure" pumps, which only hinted at potential price increases. The results showed that transparent pumps generated higher returns than obscure pumps. Nghiem et al. (2021) proposed a method using market and social media signals to predict target cryptocurrencies for pump-and-dump schemes, achieving superior results compared to existing methods.

Kamps and Kleinberg (2018) developed a machine learning model trained on pump-and-dump event data to effectively detect these schemes. Based on this analysis, La Morgia et al. (2020) conducted an in-depth examination of community-organized pumps and dumps, reporting several case studies and providing a real-time detection classifier for investors that outperformed all the other competing models. La Morgia et al. (2023) extended this work by introducing an AdaBoost classifier, which achieved better precision, recall, and F1-score results.

Overall, these studies contribute to the understanding of pump-and-dump schemes in the cryptocurrency market, provide insights into detection techniques, and emphasize the need for regulation to mitigate financial harm to investors.

2.3. The Class Imbalance Problem

The class imbalance problem is a significant challenge in classification tasks. Neglecting class imbalance can lead to severe implications for model estimation and accuracy evaluation; see the reviews by He and Garcia (2009) and Sun et al. (2009) for more in-depth discussions. Different classification methods exhibit various behaviors in the presence of class imbalance. For example, logistic regression underestimates conditional probabilities of the rare class King and Zeng (2001) and linear discriminant analysis is also affected by imbalanced class distributions Hand and Vinciotti (2003). Moreover, nonparametric methods tend to favor classification rules that perform well only on the frequent class Chawla (2003), Akbani et al. (2004), Cieslak and Chawla (2008).

Research on imbalanced classification focuses on two main approaches: learning level solutions and data level solutions Kotsiantis et al. (2006). Learning level solutions aim to enhance the learning process for the minority class by modifying the classifier or adjusting

misclassification costs Riddle et al. (1994), Cieslak and Chawla (2008), Breiman et al. (1984), Ting (2002), Kukar and Kononenko (1998), Lin et al. (2002). However, cost information is often unavailable.

Data level solutions focus on altering the class distribution to obtain a more balanced sample, but methods such as Random Over-Sampling and Random Under-Sampling have drawbacks such as loss of important data, decrease in sample size, increased risk of overfitting, and additional computational effort McCarthy et al. (2005).

Recent works propose generating artificial examples similar to the observations in the minority class to reduce overfitting and improve generalization Lee (1999), Chawla et al. (2002), Guo and Viktor (2004), Mease et al. (2007), Menardi and Torelli (2014), and Lunardon et al. (2014).

The accuracy evaluation of classifiers is equally important in class imbalance settings Weiss and Provost (2001) and Weiss (2004). Common performance measures such as error rate can be misleading due to their dependence on class distribution. Precision, recall, F-measure, receiver operating characteristic (ROC) curve, precision-recall curves, and cost curves are more suitable for evaluation He and Garcia (2009).

Estimating the accuracy measure is challenging in imbalanced learning when there are few examples of the rare class. Common error estimators include apparent error, holdout method, bootstrapping, and cross-validation techniques. However, in strongly imbalanced frameworks, evaluating results on the same data used in training or on a test sample may lead to high variance estimates and scarcity of rare examples Menardi and Torelli (2014), Schiavo and Hand (2000).

3. Materials and Methods

Detecting pump-and-dump schemes in the cryptocurrency market using high-frequency data is a challenging task. This is because such schemes are often associated with imbalanced datasets, where the instances of pump-and-dump cases are very rare, usually comprising less than 0.2% of all cases. Additionally, it is widely assumed that members of social media groups that organize these market manipulations are given detailed information about the scheme seconds or minutes before it is made public; see Kamps and Kleinberg (2018) and La Morgia et al. (2020, 2023). They pay a fee for such advanced knowledge, but in reality, unusual trading volumes can occur much earlier than that, complicating the process of detecting such schemes. These two problems can significantly impact the performance of any machine learning classifier used for detecting pump-and-dump activities, making it difficult to accurately identify such instances in a timely manner.

We will first provide a brief review of the four methods used to address class imbalance in our empirical analysis. In addition, we will present a simple solution to increase the dimensionality of the minority class, which involves flagging instances of pump-and-dump schemes from the moment of public announcement up to 60 min prior to it. Subsequently, we will discuss classification models that are capable of detecting pump-and-dump schemes.

3.1. Building Synthetic Balanced Datasets

Resolving class imbalance is crucial for accurate prediction models in binary outcomes. Popular resampling techniques include Random Under-Sampling (RUS), Random Over-Sampling (ROS), the Synthetic Minority Over-sampling Technique (SMOTE) by Chawla et al. (2002), and the Random Over-Sampling Examples (ROSE) by Menardi and Torelli (2014).

The RUS method reduces the majority class by randomly sampling from it until it matches the number of minority class samples. The ROS method replicates minority class observations to achieve balance. SMOTE generates synthetic minority class observations by interpolating between existing observations and their nearest neighbors. ROSE combines over-sampling and under-sampling by resampling the majority and minority classes

using bootstrap techniques, followed by generating synthetic samples in the feature space neighborhood Menardi and Torelli (2014).

RUS can be effective when the dataset is large, but may lose information in small datasets or if the minority class contains important information. ROS works well for small datasets but may not handle outliers or noisy data. SMOTE is effective for small datasets with proximity between minority and majority classes, but it may struggle if the minority class is sparse or if there is high overlap between the minority and majority classes in the feature space. Moreover, the choice of the number of k nearest minority class neighbors in SMOTE impacts its effectiveness. ROSE generates synthetic samples representative of the minority class and is useful for extreme imbalanced learning Menardi and Torelli (2014). The detailed description of all these resampling methods is reported in Appendix A.

The previous four methods have been shown to effectively balance the data distribution and improve model performance in several cases, enabling more precise classification of individuals into high and low risk groups. However, more recent research has revealed mixed results and has highlighted potential drawbacks associated with these resampling methods. In certain cases, these methods may even lead to a decline in model performance and wrong model interpretations; see Tantithamthavorn et al. (2018), Wongvorachan et al. (2023), and van den Goorbergh et al. (2022).

Given the potential challenges associated with the methods used to address class imbalance, we propose a simple approach to increasing the size of the minority class in the context of our dataset, which deals with pump-and-dump schemes involving crypto-assets. Prior research has suggested that paying VIP members of the social media groups involved in such market manipulations receive detailed information about them seconds or minutes before the general public; see, for example, Kamps and Kleinberg (2018) and La Morgia et al. (2020, 2023), and references therein. However, upon analyzing our dataset, we discovered that unusual trading volumes can occur much earlier, particularly within 60 min prior to the public announcement (see Figures A3–A46 in Appendix D). These atypical trades are likely executed by the true organizers of the pump-and-dump, who are likely the only ones profiting from these manipulations. As a result, we propose flagging a pump-and-dump from the minute of the public announcement up to 60 min before it, rather than only 1 or 2 min prior to it, as has been done in the past. This straightforward solution has the potential to address both insider anticipated purchases of the targeted crypto-assets and the issue of imbalanced datasets. In addition to the time frame of up to 60 min before the public announcement, we also experimented with longer time frames, such as up to 24 h before the announcement. However, the performance metrics obtained from these longer time frames were worse, and hence were not included in this study.

3.2. Methods for Binary Classification: A (Brief) Review

We employed three different classifiers, namely the logit model, random forests, and AdaBoost, to detect the onset of fraudulent pump-and-dump schemes. These classifiers have previously been used in La Morgia et al. (2020, 2023) and have demonstrated superior performance in detecting such schemes compared to previous state-of-the-art models. We present a brief overview of these models below, while a more comprehensive discussion at the textbook level can be found in Hastie et al. (2017).

The logit model can be expressed as follows,

$$P(Y = 1|\mathbf{X}) = \exp(\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p)/(1 + \exp(\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p))$$

where $P(Y = 1|\mathbf{X})$ is the probability of Y being 1 given the predictor variables $\mathbf{X}$, β_0 is the intercept, $\beta_1, \ldots, \beta_p$ are the parameters associated with each predictor variable, and $X_1, \ldots, X_p$ are the predictor variables. In our case, $Y = 1$ denotes the presence of a pump-and-dump scheme.

Random forests are an ensemble learning method that combines multiple decision trees to make a prediction. In a random forest model, multiple decision trees are constructed using bootstrap samples of the training data and a random subset of the predictor variables.

The final prediction is made by aggregating the predictions of all the individual trees. The randomness in the tree-building process and the use of multiple trees help to reduce the overfitting problem and improve the model's accuracy. The random forest model can be expressed as follows,

$$f(\mathbf{X}) = \frac{1}{M} \sum_{m=1}^{M} h_m(\mathbf{X})$$

where $f(\mathbf{X})$ is the predicted outcome variable for a given input $\mathbf{X}$, M is the number of trees in the forest, $h_m(\mathbf{X})$ is the prediction of the m-th tree in the forest for input $\mathbf{X}$, and $\sum_{m=1}^{M} h_m(\mathbf{X})$ is the aggregated prediction of all the trees.

AdaBoost is another ensemble learning method that combines multiple weak classifiers to create a strong classifier. In AdaBoost, each weak classifier is trained on a subset of the training data, and the weights of misclassified data points are increased for the next round of training. The final prediction is made by aggregating the weighted predictions of all the individual weak classifiers. The iterative process of training weak classifiers and updating weights helps the model to improve its accuracy. The AdaBoost model can be expressed as follows,

$$f(\mathbf{X}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{X}) \right)$$

where $f(\mathbf{X})$ is the predicted outcome variable for a given input $\mathbf{X}$, M is the number of weak classifiers, $h_m(\mathbf{X})$ is the prediction of the m-th weak classifier for input $\mathbf{X}$, α_m is the weight assigned to the m-th weak classifier based on its accuracy, and $\text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{X})\right)$ is the aggregated prediction of all the weak classifiers, where the sign function is used in the AdaBoost algorithm to convert the weighted sum of base classifiers' predictions into a binary prediction.

4. Results

4.1. Data

The main objective of this study is to improve the detection of pump-and-dumps with crypto assets and high-frequency data in case of imbalanced datasets and insiders' anticipated purchases. To achieve this goal, we first collected data about pump-and-dump schemes from Pumpolymp, a website that provides detailed information about all Telegram groups dealing with crypto pump-and-dumps.[1] We collected historical messages from these Telegram groups by setting keywords such as 'pump,' 'dump,' 'coin,' and 'buy.' The available data included group size, targeted coin, pump announcement time, and pump target price. Because pump events often occur in a short period, we needed a sufficient number of Telegram group messages as a preprocessing condition to have a valid pump-and-dump.

Pump-and-dump schemes are typically initiated by fraudsters who organize their activities using the social platform Telegram, where anyone can be invited to join their groups. In this study, we tracked the history of Telegram messages for 27 pump-and-dump groups from 1 June 2021 to 20 September 2022, available on the Pumpolymp website.

After searching the history of messages, we found that there was significant variability in the different pump-and-dumps schemes used in Telegram groups. Some of the most active groups sent 5 to 10 messages per day that included investment advice on a wide variety of coins, as well as detailed information about the advertised pump-and-dumps, such as coin name, target price, stop-loss price, buy price, and the crypto exchange involved. Table 1 shows some examples of the reported pump signals.

Table 1. An example of pump-and-dumps signals.

Coin Symbol	Exchange	Buy Range	Target 1	Target 2	Target 3	Stop Loss
ADX	Binance	800–830	900	1100	1200	750
AGIX	Binance	280–290	320	350	380	250
POLY	Binance	1020–1050	1150	1200	1300	900
CTSI	Binance	720–740	800	900	1100	650
FOR	Binance	93–98	110	115	125	90

If the pumped coin reached the target price, a bot automatically sent a message indicating the profit rate and completion time. We manually retrieved the messages after the pump signals and the first target prices. Since the historical information collected was redundant, we filtered the available data by using previous keywords and messages and by checking the validity of the timing of the pump signal. In this way, we managed to extract 351 pump signals relative to the Binance crypto exchange that took place in 2021 and 2022. The symbols of the 351 coins and the timing of the pump-and-dumps are reported in Table 2.

Table 2. Names of cryptoassets (Binance symbols) and dates and time of the announced pump-and-dumps.

Names	Dates and Time	Names	Dates and Time	Names	Dates and Time	Names	Dates and Time
ADX	2022-09-20 15:56:00	ATM	2022-01-12 17:37:00	SNT	2021-10-23 16:31:00	NAS	2021-08-22 17:00:00
IRIS	2022-09-17 13:24:00	ASR	2022-01-09 16:30:00	ADX	2021-10-23 16:15:00	POND	2021-08-22 13:10:00
STEEM	2022-09-16 11:05:00	POWR	2022-01-08 08:10:00	AGIX	2021-10-22 13:28:00	GVT	2021-08-21 16:00:00
BTS	2022-09-15 12:57:00	VIB	2022-01-02 17:12:00	POA	2021-10-21 16:00:00	FOR	2021-08-21 13:04:00
PROM	2022-09-15 08:52:00	NEBL	2022-01-02 17:00:00	EVX	2021-10-21 11:24:00	AION	2021-08-20 13:41:00
REQ	2022-09-11 14:42:00	ATM	2021-12-31 18:32:00	NXS	2021-10-21 10:59:00	WRX	2021-08-20 11:40:00
ARDR	2022-09-11 12:21:00	PIVX	2021-12-28 14:27:00	RDN	2021-10-21 07:41:00	ARPA	2021-08-20 04:13:00
SUPER	2022-09-11 11:54:00	POWR	2021-12-28 14:13:00	BRD	2021-10-20 16:54:00	NU	2021-08-20 02:08:00
AGIX	2022-09-11 09:19:00	PNT	2021-12-26 18:53:00	AVAX	2021-10-19 13:03:00	FOR	2021-08-17 12:03:00
PIVX	2022-09-11 08:41:00	RAMP	2021-12-26 15:45:00	BEAM	2021-10-17 17:48:00	OAX	2021-08-17 02:54:00
GTO	2022-09-06 10:52:00	TCT	2021-12-25 14:13:00	ANT	2021-10-17 07:27:00	CTXC	2021-08-16 05:46:00
CVX	2022-09-06 06:53:00	OG	2021-12-25 13:55:00	WAXP	2021-10-16 17:35:00	ADX	2021-08-15 17:31:00
WABI	2022-09-01 08:32:00	AGIX	2021-12-24 17:21:00	IRIS	2021-10-16 17:14:00	BEAM	2021-08-15 16:48:00
SUPER	2022-08-29 15:19:00	CTXC	2021-12-24 12:49:00	GXS	2021-10-16 16:20:00	RDN	2021-08-15 04:13:00
AION	2022-08-29 10:29:00	NEBL	2021-12-24 12:39:00	PNT	2021-10-15 15:52:00	MDA	2021-08-10 17:38:00
ELF	2022-08-28 09:20:00	RDN	2021-12-17 13:47:00	IRIS	2021-10-14 20:49:00	ARPA	2021-08-10 04:44:00
ADX	2022-08-26 14:08:00	RAMP	2021-12-16 14:52:00	WABI	2021-10-14 15:00:00	ARPA	2021-08-09 17:57:00
CTXC	2022-08-26 12:18:00	AST	2021-12-16 14:04:00	GXS	2021-10-13 14:27:00	AVAX	2021-08-08 13:44:00

Table 2. *Cont.*

Names	Dates and Time	Names	Dates and Time	Names	Dates and Time	Names	Dates and Time
DOCK	2022-08-24 07:37:00	SNT	2021-12-16 12:47:00	POND	2021-10-13 14:16:00	APPC	2021-08-08 01:00:00
AION	2022-08-23 10:43:00	NXS	2021-12-16 02:32:00	BRD	2021-10-12 17:19:00	ANKR	2021-08-07 15:16:00
NEXO	2022-08-20 04:56:00	AST	2021-12-09 06:00:00	SKL	2021-10-12 16:58:00	CTXC	2021-08-07 14:59:00
OAX	2022-08-18 14:03:00	TCT	2021-12-09 02:20:00	VIB	2021-10-12 11:36:00	BRD	2021-08-06 11:34:00
ELF	2022-08-14 11:50:00	QLC	2021-12-08 18:54:00	VIB	2021-10-12 10:27:00	LSK	2021-08-06 03:31:00
VIB	2022-08-14 07:00:00	SNM	2021-12-07 16:00:00	WNXM	2021-10-10 17:00:00	EVX	2021-08-05 04:02:00
IDEX	2022-08-14 06:32:00	GRS	2021-12-06 16:00:00	VIB	2021-10-10 05:31:00	BTS	2021-08-02 23:13:00
WABI	2022-08-14 03:31:00	ADX	2021-12-06 06:45:00	VIB	2021-10-09 17:20:00	NEBL	2021-07-30 09:24:00
PHA	2022-08-11 15:51:00	NEBL	2021-12-05 15:00:00	NXS	2021-10-09 14:38:00	ARPA	2021-07-27 13:47:00
BEAM	2022-08-07 17:40:00	NXS	2021-12-04 12:59:00	POA	2021-10-09 13:13:00	DREP	2021-07-25 17:00:00
SUPER	2022-08-05 08:46:00	ELF	2021-12-02 19:30:00	ADX	2021-10-09 06:27:00	EVX	2021-07-25 15:29:00
BRD	2022-07-30 11:38:00	QSP	2021-11-30 12:31:00	NAV	2021-10-08 16:17:00	NXS	2021-07-25 15:22:00
FOR	2022-07-29 14:55:00	VIB	2021-11-29 07:49:00	SKY	2021-10-08 15:42:00	QLC	2021-07-25 12:03:00
SUPER	2022-07-29 10:39:00	ALPHA	2021-11-29 05:21:00	VIB	2021-10-08 12:40:00	FOR	2021-07-25 04:24:00
AGIX	2022-07-29 08:32:00	VIB	2021-11-28 18:53:00	QSP	2021-10-08 12:10:00	GRS	2021-07-24 16:58:00
VIB	2022-07-27 11:54:00	PHB	2021-11-28 17:00:00	ANT	2021-10-07 05:39:00	GVT	2021-07-24 16:40:00
RNDR	2022-07-24 07:29:00	GRS	2021-11-28 16:01:00	BTCST	2021-10-07 05:27:00	POA	2021-07-23 13:03:00
MDA	2022-07-23 13:16:00	FXS	2021-11-28 15:48:00	QLC	2021-10-05 18:15:00	FOR	2021-07-18 18:34:00
FXS	2022-07-22 11:31:00	VIB	2021-11-28 15:26:00	SKY	2021-10-05 15:00:00	REQ	2021-07-17 16:57:00
DOCK	2022-07-21 10:48:00	FOR	2021-11-28 07:09:00	GTO	2021-10-05 12:54:00	CTSI	2021-07-16 14:50:00
LOKA	2022-07-18 13:00:00	GXS	2021-11-27 16:50:00	EVX	2021-10-03 16:51:00	TRU	2021-07-16 14:35:00
OCEAN	2022-07-17 17:10:00	NXS	2021-11-27 16:37:00	GXS	2021-10-02 17:32:00	POA	2021-07-11 17:00:00
FIO	2022-07-17 15:56:00	TKO	2021-11-27 10:49:00	QLC	2021-10-02 11:03:00	OAX	2021-07-11 12:00:00
VIB	2022-07-16 16:28:00	ELF	2021-11-26 04:19:00	APPC	2021-09-30 14:07:00	VIA	2021-07-10 14:42:00
DOCK	2022-07-16 15:09:00	NAV	2021-11-26 01:21:00	MTH	2021-09-28 14:18:00	FIS	2021-07-07 19:06:00
BEAM	2022-07-16 12:31:00	MDA	2021-11-23 15:00:00	SUPER	2021-09-25 11:07:00	ARPA	2021-07-07 13:35:00
CTXC	2022-07-15 10:08:00	REQ	2021-11-23 11:54:00	FIO	2021-09-23 14:20:00	TWT	2021-07-06 14:30:00
MDT	2022-07-14 18:23:00	ARDR	2021-11-21 13:27:00	NXS	2021-09-22 16:00:00	MDA	2021-07-05 16:14:00
PHA	2022-07-14 11:45:00	PIVX	2021-11-21 11:08:00	EVX	2021-09-22 14:24:00	NEBL	2021-07-05 15:17:00

Table 2. *Cont.*

Names	Dates and Time	Names	Dates and Time	Names	Dates and Time	Names	Dates and Time
LOKA	2022-07-14 05:37:00	NXS	2021-11-20 13:36:00	PNT	2021-09-22 13:14:00	IDEX	2021-07-03 16:12:00
DOCK	2022-07-13 17:06:00	RAMP	2021-11-19 11:03:00	NEBL	2021-09-22 13:07:00	LINK	2021-07-01 21:10:00
HIGH	2022-07-13 09:56:00	APPC	2021-11-16 16:22:00	NAV	2021-09-21 13:50:00	QSP	2021-07-01 19:07:00
PIVX	2022-07-12 15:36:00	NXS	2021-11-16 16:00:00	FXS	2021-09-19 17:00:00	DLT	2021-07-01 17:52:00
MLN	2022-07-12 12:33:00	NAS	2021-11-14 15:00:00	BRD	2021-09-19 14:51:00	OAX	2021-06-29 14:27:00
ADX	2022-07-11 14:09:00	EVX	2021-11-12 16:00:00	AION	2021-09-18 12:47:00	ARPA	2021-06-28 17:30:00
VIB	2022-07-11 13:45:00	ATM	2021-11-12 15:43:00	CTXC	2021-09-15 17:00:00	DOGE	2021-06-27 19:14:00
BEAM	2022-07-09 13:41:00	ASR	2021-11-12 15:36:00	PIVX	2021-09-15 15:29:00	MTH	2021-06-27 17:00:00
OAX	2022-07-01 09:25:00	NAS	2021-11-12 14:37:00	CHZ	2021-09-14 12:30:00	ADX	2021-06-26 16:08:00
ATM	2022-03-22 10:45:00	EVX	2021-11-11 12:29:00	RDN	2021-09-13 18:10:00	ELF	2021-06-24 15:55:00
ASR	2022-03-22 10:08:00	OAX	2021-11-09 16:56:00	GTO	2021-09-12 12:40:00	WABI	2021-06-20 17:00:00
AST	2022-03-19 16:04:00	SUPER	2021-11-09 03:49:00	DLT	2021-09-11 16:00:00	TRU	2021-06-20 14:36:00
ATA	2022-03-16 18:52:00	EPS	2021-11-08 14:12:00	ROSE	2021-09-11 12:22:00	NAV	2021-06-20 11:00:00
MITH	2022-03-14 14:12:00	VIB	2021-11-07 18:58:00	PHB	2021-09-11 11:42:00	VIA	2021-06-19 17:48:00
SNM	2022-03-14 10:00:00	VIB	2021-11-07 17:21:00	ALGO	2021-09-11 07:24:00	CDT	2021-06-18 14:22:00
POND	2022-03-13 14:57:00	MTH	2021-11-07 17:00:00	QLC	2021-09-10 17:41:00	NAS	2021-06-16 16:29:00
FIRO	2022-03-13 09:57:00	NXS	2021-11-06 11:02:00	AVAX	2021-09-10 12:16:00	ARK	2021-06-16 15:09:00
FOR	2022-03-12 06:41:00	NULS	2021-11-05 19:11:00	OAX	2021-09-10 12:13:00	SXP	2021-06-16 10:32:00
MDX	2022-02-15 12:41:00	QSP	2021-11-05 09:46:00	SOL	2021-09-09 17:48:00	CTXC	2021-06-15 16:41:00
OG	2022-02-15 10:26:00	BRD	2021-11-04 13:36:00	FUN	2021-09-09 14:00:00	ONE	2021-06-15 16:36:00
KLAY	2022-02-14 20:10:00	REQ	2021-11-04 13:16:00	NEAR	2021-09-08 08:26:00	REP	2021-06-15 16:00:00
FOR	2022-02-13 15:10:00	BEL	2021-11-04 09:41:00	ADX	2021-09-07 14:28:00	RLC	2021-06-15 15:54:00
SKL	2022-02-13 10:21:00	FIRO	2021-11-03 16:25:00	ALGO	2021-09-06 16:53:00	OST	2021-06-15 15:53:00
NEBL	2022-02-11 14:30:00	MTH	2021-11-03 06:29:00	SFP	2021-09-06 11:06:00	MATIC	2021-06-14 12:31:00
GRS	2022-02-09 14:08:00	OAX	2021-11-02 16:16:00	DOGE	2021-09-05 23:32:00	DLT	2021-06-13 17:01:00
ELF	2022-02-07 14:39:00	CHR	2021-11-01 14:57:00	NXS	2021-09-05 17:39:00	FIO	2021-06-13 17:01:00
BTG	2022-02-07 12:24:00	VIB	2021-10-31 17:06:00	VIB	2021-09-05 17:00:00	LIT	2021-06-13 16:59:00
QLC	2022-02-05 16:50:00	BRD	2021-10-31 15:00:00	MDA	2021-09-05 11:03:00	QSP	2021-06-13 16:00:00
DEGO	2022-02-04 12:30:00	DUSK	2021-10-31 09:29:00	PNT	2021-09-04 18:00:00	GVT	2021-06-12 14:45:00

Table 2. *Cont.*

Names	Dates and Time	Names	Dates and Time	Names	Dates and Time	Names	Dates and Time
AGIX	2022-02-03 14:40:00	NEBL	2021-10-30 17:30:00	ALPHA	2021-09-02 17:03:00	POA	2021-06-11 14:54:00
BICO	2022-02-03 12:54:00	RDN	2021-10-30 16:06:00	CHZ	2021-09-01 08:38:00	QLC	2021-06-09 16:13:00
OG	2022-02-03 09:49:00	NXS	2021-10-30 05:24:00	SKL	2021-08-30 18:40:00	WABI	2021-06-07 13:28:00
MDX	2022-02-02 18:04:00	VIB	2021-10-27 21:13:00	BRD	2021-08-29 17:00:00	CND	2021-06-06 17:00:00
ASR	2022-01-27 12:19:00	ARDR	2021-10-26 17:06:00	ICP	2021-08-29 13:38:00	MTH	2021-06-06 17:00:00
OM	2022-01-26 11:31:00	GRS	2021-10-26 15:15:00	BNB	2021-08-29 12:54:00	POA	2021-06-06 10:49:00
SNM	2022-01-26 11:12:00	SC	2021-10-26 10:46:00	KEEP	2021-08-27 19:27:00	ELF	2021-06-06 08:48:00
OG	2022-01-26 11:03:00	VIB	2021-10-24 19:15:00	GXS	2021-08-27 16:00:00	MTL	2021-06-04 17:59:00
QLC	2022-01-26 10:41:00	EVX	2021-10-24 17:00:00	QSP	2021-08-27 15:50:00	ENJ	2021-06-03 12:55:00
PIVX	2022-01-24 18:59:00	IDEX	2021-10-24 14:47:00	BLZ	2021-08-24 06:38:00	REEF	2021-06-03 05:04:00
PIVX	2022-01-18 16:00:00	FOR	2021-10-24 14:00:00	TCT	2021-08-24 00:51:00	EOS	2021-06-01 15:21:00
ELF	2022-01-17 17:46:00	VIB	2021-10-24 09:05:00	SRM	2021-08-22 20:31:00		

We obtained historical transaction data of the identified pumped coins through the CryptoCurrency eXchange Trading Library (CCXT).[2] The CCXT library is used to connect and trade with cryptocurrency exchanges and payment processing services worldwide, providing a large set of transaction data. The available data includes the timestamp, datetime, price, amount, volume expressed in BTC, and order type (buy or sell). Even though several types of orders are possible, such as market orders, limit orders, limit sell orders, and limit buy orders, we only collected basic order information (buy or sell) due to the limitations of the CCXT Library. A small example of the obtained raw data is reported in Table 3. In line with previous studies La Morgia et al. (2020, 2023), we analyzed a limited dataset consisting of only three days of data surrounding the fraudulent event, including the day of the fraud, the day before, and the day after. This approach assumes the absence of any other fraud in the dataset during this period.

Table 3. An example of coin (raw) data.

Symbol	*Timestamp*	*Datetime*	*Side*	*Price*	*Amount*	*btc_VOLUME*
ADX/BTC	1663602963725	2022-09-19T15:56:03.725Z	sell	8.23×10^{-6}	694	0.00571162
ADX/BTC	1663602965541	2022-09-19T15:56:05.541Z	sell	8.23×10^{-6}	192	0.00158016
ADX/BTC	1663602970102	2022-09-19T15:56:10.102Z	sell	8.23×10^{-6}	705	0.00580215
ADX/BTC	1663602978208	2022-09-19T15:56:18.208Z	buy	8.24×10^{-6}	415	0.0034196
ADX/BTC	1663603016649	2022-09-19T15:56:56.649Z	buy	8.25×10^{-6}	2910	0.0240075
ADX/BTC	1663603028493	2022-09-19T15:57:08.493Z	buy	8.26×10^{-6}	2928	0.02418528
ADX/BTC	1663603029672	2022-09-19T15:57:09.672Z	sell	8.26×10^{-6}	457	0.00377482
ADX/BTC	1663603037503	2022-09-19T15:57:17.503Z	buy	8.27×10^{-6}	1276	0.01055252
...	...	...	...	...	...	...

Understanding how to detect pump-and-dump schemes requires a basic knowledge of cryptocurrency trading. Pending orders for a cryptocurrency are listed in the order book, which is a double sorted list of sell (ask) and buy (bid) orders. The orders are sorted

by price, with the lowest ask price listed first, and the highest bid price listed first. To execute a buy order quickly, traders can use a buy market order, which fills all pending asks in the order book until the requested amount of currency is traded. However, the difference between the first and last ask prices can be high, particularly in markets with low liquidity, leading to an unpredictably high total order cost. A more prudent investor would use limit buy orders, which specify the maximum price at which to buy a security. Although buy market orders are not frequently used in everyday transactions, members of pump-and-dump groups use them when fast execution is necessary. Unfortunately, the CCXT Library (and the Binance APIs) do not provide information about the type of order placed by the buyer (e.g., market, limit, or stop loss), requiring us to infer this information in a different way. La Morgia et al. (2020, 2023) used the fact that market orders are completed instantly, and they aggregated all trades filled at the exact millisecond as a single market order. They referred to these orders as *rush orders*. La Morgia et al. (2020, 2023) argued that this approach provided a good indication of the abrupt rise in market orders, and we followed the same methodology in our work.

As highlighted in Kamps and Kleinberg (2018) and La Morgia et al. (2020, 2023), the trading volume, including both buy and sell orders, can be a good indicator of potential pump-and-dump schemes. Coins selected by manipulators usually exhibit a weak trend, with low volume and flat K-lines (a.k.a Japanese candlesticks) being their hallmarks. Abnormal volume may indicate that the market situation is about to change and signal that manipulators are ready to pump the targeted coin.

We developed our classification models based on the approach of La Morgia et al. (2020, 2023), which built upon the methodology proposed in Siris and Papagalou (2004) for detecting Denial of Service attacks using an adaptive threshold. Specifically, we segmented the data into chunks of s seconds and defined a moving window of size w hours. We conducted various experiments with different feature sets and settings regarding the window and chunk sizes. We chose a reasonably short chunk size to build a classifier capable of detecting pump-and-dump schemes as soon as possible from the moment they start. We considered chunk sizes of 30, 60, and 90 s and window sizes of 10, 30, and 60 min.

In our baseline scenario, we first flagged a pump-and-dump scheme (that is $Y_t = 1$) from the moment of the public announcement up to 60 s before it (or 120 s, in case of chunk sizes of 90 s). This was done to acknowledge that the VIP members of the social media groups who orchestrate these manipulations are presumed to receive insider information seconds or minutes ahead of the general public. As such, our analysis accounts for the possibility of information asymmetry and its potential impact on the identified market manipulation. However, as previously discussed, our investigation has revealed that there can be instances of unusual trading volumes occurring much earlier, specifically within the 60-minute period preceding the public announcement, as shown in Figures A3–A46 in Appendix D. These figures indicate that almost 50% of the examined pump-and-dumps had significant abnormal trading volumes within one hour before the announcement (we used the well-known 3-sigma rule to quickly identify potential outliers, but more robust methods can be utilized; see Pukelsheim (1994), Rousseeuw and Leroy (2005), and Charu (2019). These anomalous trades are most likely executed by the actual organizers of the pump-and-dump scheme, who are presumably the sole beneficiaries of these manipulations. Therefore, we broadened the identification of a pump-and-dump scheme from the time of the public announcement to the 60-minute period prior to it, instead of only considering the 1 or 2 min immediately before the announcement. This straightforward solution has the potential to address both insider trading of the targeted crypto-assets and the issue of imbalanced datasets.

The regressors employed in our analysis included:

- *StdRushOrders* and *AvgRushOrders*: the moving standard deviation and the average of the volume of rush orders in each chunk of the moving window.
- *StdTrades*: the moving standard deviation of the number of trades.

- *StdVolumes* and *AvgVolumes*: the moving standard deviation and the average of the volume of trades in each chunk of the moving window.
- *StdPrice* and *AvgPrice*: the moving standard deviation and average of the closing price.
- *HourSin*, *HourCos*, *MinuteCos*, *MinuteSin*: the hour and minute of the first transaction in each chunk. We used the sine and cosine functions to express their cyclical nature[3].

We also tested the moving average of the maximum price in each chunk, as proposed in La Morgia et al. (2020, 2023). However, it was almost perfectly collinear with the average of the closing price and was not included in our analysis.

The number of data points corresponding to different combinations of chunk sizes and window sizes, and the frequency of detection of pump-and-dump schemes (i.e., the number of times where $Y_t = 1$) in both levels and in %, when the pump-and-dump is flagged from the moment of the public announcement up to 60/120 s and 60 min before it, respectively, are reported in Table 4 . The ADF and KPSS unit root test statistics for the average number of rush orders, average trade volumes, and average closing price, as well as the *p*-values of the Jarque–Bera normality tests, are presented in Table 5.

Table 4. The number of data points corresponding to different combinations of chunk sizes and window sizes, and the frequency of detection of pump-and-dump schemes (P&D) in both levels and in %, when the pump-and-dump is flagged from the moment of the public announcement up to 60/120 s and 60 min before it, respectively.

Chunk Size/... .../Window Size	N. of Data	N. of Times $Y_t = 1$ (%) P&D Flagged 1 or 2 Min before Announcement	N. of Times $Y_t = 1$ (%) P&D Flagged 60 Min before Announcement
30 s./10 m.	441,130	518 (0.12%)	11,534 (2.61%)
30 s./30 m.	457,804	535 (0.12%)	11,900 (2.60%)
30 s./60 m.	460,033	537 (0.12%)	11,938 (2.60%)
60 s./10 m.	320,178	503 (0.15%)	8378 (2.62%)
60 s./30 m.	337,519	532 (0.16%)	8777 (2.60%)
60 s./60 m.	339,782	537 (0.16%)	8818 (2.60%)
90 s./10 m.	258,095	455 (0.18%)	6659 (2.58%)
90 s./30 m.	274,723	484 (0.18%)	7035 (2.56%)
90 s./60 m.	277,034	488 (0.18%)	7076 (2.55%)

Table 5. ADF and KPSS tests, and *p*-values of the Jarque–Bera normality tests for different combinations of chunk sizes and window sizes. ** The null hypothesis is rejected at the 1% probability level; * The null hypothesis is rejected at the 5% probability level ($H_{0,ADF}$: unit root, $H_{0,KPSS}$: stationarity).

Chunk Size/... .../Window Size	AvgRushOrders ADF/KPSS	AvgPrice ADF/KPSS	AvgVolumes ADF/KPSS	AvgRushOrders JB (*p*-Value)	AvgPrice JB (*p*-Value)	AvgVolumes JB (*p*-Value)
30 s./10 m.	−60.28 **/0.66 *	−6.27 **/1.40 **	−40.28 **/0.76 **	0.00	0.00	0.00
30 s./30 m.	−45.86 **/0.57 *	−6.38 **/1.37 **	−36.87 **/0.69 **	0.00	0.00	0.00
30 s./60 m.	−40.25 **/0.51 *	−6.40 **/1.37 **	−32.49 **/0.68 **	0.00	0.00	0.00
60 s./10 m.	−56.26 **/0.41	−7.48 **/0.88 **	−35.60 **/0.45	0.00	0.00	0.00
60 s./30 m.	−39.42 **/0.35	−7.67 **/0.88 **	−30.69 **/0.42	0.00	0.00	0.00
60 s./60 m.	−36.91 **/0.32	−7.70 **/0.88 **	−29.58 **/0.41	0.00	0.00	0.00
90 s./10 m.	−51.46 **/0.31	−8.21 **/0.70 *	−33.82 **/0.35	0.00	0.00	0.00
90 s./30 m.	−37.26 **/0.27	−8.45 **/0.65 *	−28.71 **/0.32	0.00	0.00	0.00
90 s./60 m.	−32.93 **/0.24	−8.49 **/0.68 *	−25.69 **/0.31	0.00	0.00	0.00

All the regressors used in our analysis, except *MinuteCos* and *MinuteSin*, which are too dense to be visible in a plot, for a chunk size of 60 s and a window size of 60 min are illustrated in Figure A1 in Appendix B. The plots for the other combinations of chunk sizes and window sizes are available from the authors upon request.

Table 4 indicates that the occurrence of $Y_t = 1$, signifying a pump-and-dump event, is very rare (less than 0.2% of all instances) if the pump-and-dump is flagged 1 or 2 min before the public announcement. The situation can be improved if we identify pump-and-dump schemes during the time between the public announcement and the 60-min period preceding it, which represents approximately 2.6% of all instances. As a result, detecting a pump-and-dump poses a considerable challenge. Moreover, while the regressors are generally stationary, their distribution strongly deviates from normality; see Table 5.[4] In this regard, to evaluate the robustness of our analysis, we will also examine in Section 5 the potential utility of a generalized Box–Cox transformation that accounts for zero values, which may potentially improve the accuracy of classification.

4.2. Empirical Analysis

As previously discussed, we used various types of regressors in conjunction with three distinct classifiers (Logistic Regression, Random Forest, and the AdaBoost classifier) to identify the onset of a pump-and-dump scheme. Due to the limited number of pump-and-dump instances in our dataset, we did not partition the data into standard training and testing sets. Instead, we employed a 5-fold cross-validation method to obtain a more reliable performance evaluation (see also La Morgia et al. (2020, 2023) for similar approaches). For the random forest classifier, we employed a forest of 200 trees, with a maximum depth of 5 for each tree. For the AdaBoost classifier, we used 10 weak classifiers.

For the sake of space and interest, given the very large dataset at our disposal, we focused exclusively on the 5-fold cross-validation, whereas a small summary of the in-sample analysis can be found in the Appendix C.

Due to the large class imbalance in our dataset, we employed the resampling methods discussed in Section 3.1 to address this issue and we artificially balanced the dataset during model training. Resampling techniques used to create synthetic balanced datasets are also commonly referred to as "subsampling" techniques. The detailed implementation in R with the *caret* package can be found at https://topepo.github.io/caret/subsampling-for-class-imbalances.html, accessed on 1 December 2022. For our 5-fold cross-validation, we considered ten possible data configurations as follows:

1 Model training using the original dataset with pump-and-dumps flagged 1 or 2 min before the public announcement;
2–5 Model training using synthetic balanced data created with Random Under-Sampling (RUS), Random Over-Sampling (ROS), Synthetic Minority Over-sampling Technique (SMOTE), and the Random Over-Sampling Examples (ROSE) method, respectively, with pump-and-dumps flagged 1 or 2 min before the public announcement.
6 Model training using the original dataset with pump-and-dumps flagged 60 min before the public announcement;
7–10 Model training using synthetic balanced data created with Random Under-Sampling (RUS), Random Over-Sampling (ROS), Synthetic Minority Over-sampling Technique (SMOTE), and the Random Over-Sampling Examples (ROSE) method, respectively, with pump-and-dumps flagged 60 min before the public announcement.

The above-mentioned procedures were employed to train our three classifiers and evaluate their performances in detecting pump-and-dump schemes with crypto-assets. The following performance metrics were then used to evaluate the three competing classifiers:

- *Area under the Receiver Operating Characteristic (ROC) curve (AUC)*: it is a metric that measures the ability of a binary classification model to distinguish between positive and negative classes. It is calculated as the area under the receiver operating characteristic (ROC) curve, which plots the true positive rate against the false positive rate at various classification thresholds. AUC is commonly used to evaluate the overall performance of a classification model, and it ranges from 0 to 1, with a higher score indicating better performance; see Sammut and Webb (2011), pp. 869–75, and references therein for more details.

- *H-measure*: the AUC has some well-known drawbacks, such as potentially providing misleading results when ROC curves cross. However, a more serious deficiency of the AUC (recognized only recently by Hand (2009) is that it is fundamentally incoherent in terms of misclassification costs, as it employs different misclassification cost distributions for different classifiers. This is problematic because the severity of misclassification for different points is a property of the problem rather than the classifiers that have been selected. The H-measure, proposed by Hand (2009) and Hand and Anagnostopoulos (2022), is a measure of classification performance that addresses the incoherence of AUC by introducing costs for different types of misclassification. However, it is common for the precise values of costs to be unknown at the time of classifier evaluation. To address this, they proposed taking the expectation over a distribution of likely cost values. While researchers should choose this distribution based on their knowledge of the problem, Hand (2009) and Hand and Anagnostopoulos (2022) also recommended a standard default distribution, such as beta distribution, for conventional measures, and they also generalized this approach to cover cases when class sizes are unknown. Moreover, in many problems, class sizes are extremely unbalanced (as in our case with pump-and-dumps), and it is rare to want to treat the two classes symmetrically. To address this issue, the H-measure requires a Severity Ratio (SR) that represents how much more severe misclassifying a class 0 instance is than misclassifying a class 1 instance. The severity ratio is formally defined as $SR = c_0/c_1$, where $c_0 > 0$ is the cost of misclassifying a class 0 datapoint as class 1. It is sometimes more convenient to consider the normalized cost $c = c_0/(c_0 + c_1)$, so that $SR = c/(1-c)$, where c is in the range [0,1]. By default, the severity ratio is set to the reciprocal of the relative class frequency, i.e., $SR = \hat{\pi}_1/\hat{\pi}_0$, so that misclassifying the rare class is considered a graver mistake. For more detailed motivation of this default value, see Hand and Anagnostopoulos (2014).
- *Accuracy*: it is a measure of the correct classification rate, which is the ratio of the number of correct predictions to the total number of predictions made. It is a widely used performance metric for binary classification, but it can be misleading in cases of imbalanced datasets.
- *Sensitivity*: it measures the proportion of true positive predictions out of all the actual positive cases in the dataset. In other words, it measures how well the model identifies the positive cases correctly (in our case, the number of pump-and-dumps).
- *Specificity*: it measures the proportion of true negative predictions out of all the actual negative cases in the dataset. In other words, it measures how well the model identifies the negative cases correctly.

The first two metrics are threshold-independent and do not depend on the specific choice of the threshold used to determine the final assigned class, whereas the last three metrics do depend on the choice of the threshold. In the latter case, we considered the following possible values:

- $p = 50\%$: this is the classical level used in many applications in various fields;
- $p = 0.18\%$: this level is close to the empirical frequency of pump-and-dumps in the dataset where they are flagged 1 or 2 min before the public announcement (that is, close to the so-called *observed prevalence*[5], $\hat{\pi}_{1,1\,or\,2min.}$);
- $p = 2.6\%$: this level is close to the empirical frequency of pump-and-dumps in the dataset where they are flagged 60 min before the public announcement ($\hat{\pi}_{1,60min.}$).

We chose thresholds close to the observed prevalence because this is often the optimal choice for datasets with very low prevalence; see Freeman and Moisen (2008) and references therein for more details. We would like to emphasize that there are several methods available to select the optimal threshold, and the choice of the threshold ultimately depends on the cost function employed by the model user. Two large reviews with detailed applications in R software can be found in López-Ratón et al. (2014) and Thiele and Hirschfeld (2021).

The performance metrics of the 5-fold cross-validation for the ten possible data configurations, three classifiers, and for different chunk sizes and window sizes are reported in Tables 6–11.

Table 6. Performance metrics of 5-fold cross-validation. Original data with pump-and-dumps flagged 1 or 2 min before the public announcement.

(1) ORIGINAL DATA (P&Ds Flagged 1 or 2 min before Announcement)								
Chunk/Window	*Threshold Independent*		*Threshold Dependent (p = 50%)*			*Threshold Dependent (p = 0.18%)*		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.52	0.04	1.00	0.01	1.00	1.00	0.05	1.00
Logit	0.73	0.16	1.00	0.01	1.00	0.76	0.53	0.76
Random Forest	0.82	0.55	1.00	0.07	1.00	0.97	0.65	0.97
30 s./30 m. Ada.	0.52	0.03	1.00	0.01	1.00	1.00	0.04	1.00
Logit	0.73	0.16	1.00	0.00	1.00	0.75	0.53	0.75
Random Forest	0.81	0.53	1.00	0.08	1.00	0.97	0.63	0.97
30 s./60 m. Ada.	0.51	0.02	1.00	0.00	1.00	1.00	0.03	1.00
Logit	0.73	0.16	1.00	0.00	1.00	0.75	0.54	0.75
Random Forest	0.84	0.60	1.00	0.07	1.00	0.97	0.69	0.97
60 s./10 m. Ada.	0.53	0.05	1.00	0.04	1.00	1.00	0.06	1.00
Logit	0.72	0.16	1.00	0.02	1.00	0.64	0.72	0.64
Random Forest	0.75	0.39	1.00	0.09	1.00	0.95	0.53	0.95
60 s./30 m. Ada.	0.53	0.04	1.00	0.03	1.00	1.00	0.05	1.00
Logit	0.72	0.16	1.00	0.01	1.00	0.64	0.73	0.64
Random Forest	0.74	0.38	1.00	0.11	1.00	0.95	0.51	0.95
60 s./60 m. Ada.	0.51	0.02	1.00	0.00	1.00	1.00	0.02	1.00
Logit	0.72	0.15	1.00	0.01	1.00	0.64	0.73	0.64
Random Forest	0.77	0.43	1.00	0.07	1.00	0.95	0.57	0.95
90 s./10 m. Ada.	0.53	0.05	1.00	0.05	1.00	1.00	0.06	1.00
Logit	0.73	0.18	1.00	0.03	1.00	0.61	0.77	0.61
Random Forest	0.71	0.30	1.00	0.10	1.00	0.94	0.46	0.94
90 s./30 m. Ada.	0.53	0.05	1.00	0.04	1.00	1.00	0.06	1.00
Logit	0.73	0.17	1.00	0.01	1.00	0.61	0.77	0.61
Random Forest	0.72	0.34	1.00	0.09	1.00	0.94	0.49	0.94
90 s./60 m. Ada.	0.52	0.04	1.00	0.02	1.00	1.00	0.05	1.00
Logit	0.73	0.17	1.00	0.01	1.00	0.62	0.77	0.62
Random Forest	0.75	0.39	1.00	0.07	1.00	0.95	0.53	0.95

Table 7. Performance metrics of 5-fold cross-validation. Balanced data created with Random Under-Sampling (RUS) and Random Over-Sampling (ROS), with pump-and-dumps flagged 1 or 2 min before the public announcement.

RANDOM UNDER-SAMPLING (P&Ds Flagged 1 or 2 min before Announcement)								
Chunk/Window	*Threshold Independent*		*Threshold Dependent (p = 50%)*			*Threshold Dependent (p = 0.18%)*		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.74	0.16	0.61	0.75	0.61	0.31	0.94	0.31
Logit	0.74	0.17	0.60	0.76	0.60	0.00	1.00	0.00
Random Forest	0.87	0.45	0.76	0.80	0.76	0.00	1.00	0.00
30 s./30 m. Ada.	0.74	0.15	0.56	0.81	0.56	0.29	0.95	0.29
Logit	0.72	0.15	0.60	0.77	0.60	0.00	1.00	0.00
Random Forest	0.87	0.45	0.75	0.81	0.75	0.00	1.00	0.00
30 s./60 m. Ada.	0.74	0.15	0.54	0.84	0.54	0.26	0.96	0.26
Logit	0.73	0.15	0.60	0.78	0.60	0.00	1.00	0.00
Random Forest	0.89	0.52	0.74	0.85	0.74	0.00	1.00	0.00
60 s./10 m. Ada.	0.76	0.18	0.65	0.75	0.65	0.28	0.96	0.28
Logit	0.73	0.17	0.60	0.76	0.60	0.00	1.00	0.00
Random Forest	0.84	0.36	0.73	0.77	0.73	0.00	1.00	0.00
60 s./30 m. Ada.	0.73	0.14	0.59	0.77	0.59	0.27	0.95	0.26
Logit	0.73	0.15	0.59	0.77	0.59	0.00	1.00	0.00
Random Forest	0.82	0.33	0.71	0.78	0.71	0.00	1.00	0.00
60 s./60 m. Ada.	0.73	0.14	0.54	0.82	0.54	0.25	0.95	0.25
Logit	0.72	0.14	0.60	0.76	0.60	0.00	1.00	0.00
Random Forest	0.83	0.33	0.71	0.77	0.71	0.00	1.00	0.00
90 s./10 m. Ada.	0.77	0.21	0.70	0.69	0.70	0.25	0.97	0.25
Logit	0.74	0.18	0.62	0.75	0.62	0.00	1.00	0.00
Random Forest	0.82	0.34	0.73	0.76	0.73	0.00	1.00	0.00
90 s./30 m. Ada.	0.75	0.17	0.63	0.76	0.63	0.30	0.95	0.30
Logit	0.73	0.16	0.60	0.76	0.60	0.00	1.00	0.00
Random Forest	0.81	0.31	0.72	0.74	0.72	0.00	1.00	0.00
90 s./60 m. Ada.	0.74	0.16	0.61	0.76	0.61	0.24	0.97	0.24
Logit	0.73	0.16	0.61	0.77	0.61	0.00	1.00	0.00
Random Forest	0.81	0.31	0.71	0.73	0.71	0.00	1.00	0.00

Table 7. *Cont.*

RANDOM OVER-SAMPLING (P&Ds Flagged 1 or 2 min before Announcement)								
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 0.18%)***		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.68	0.10	0.56	0.77	0.56	0.53	0.80	0.53
Logit	0.74	0.17	0.60	0.76	0.60	0.00	1.00	0.00
Random Forest	0.83	0.52	1.00	0.18	1.00	0.92	0.69	0.92
30 s./30 m. Ada.	0.65	0.07	0.63	0.66	0.63	0.60	0.70	0.60
Logit	0.73	0.15	0.60	0.78	0.60	0.00	1.00	0.00
Random Forest	0.82	0.53	1.00	0.19	1.00	0.93	0.68	0.93
30 s./60 m. Ada.	0.67	0.09	0.53	0.80	0.53	0.52	0.80	0.52
Logit	0.73	0.15	0.60	0.78	0.60	0.00	1.00	0.00
Random Forest	0.83	0.56	1.00	0.18	1.00	0.92	0.69	0.92
60 s./10 m. Ada.	0.71	0.15	0.62	0.80	0.62	0.62	0.81	0.62
Logit	0.73	0.17	0.60	0.75	0.60	0.00	1.00	0.00
Random Forest	0.76	0.37	1.00	0.13	1.00	0.89	0.59	0.89
60 s./30 m. Ada.	0.67	0.09	0.58	0.75	0.58	0.57	0.77	0.57
Logit	0.72	0.15	0.60	0.76	0.60	0.00	1.00	0.00
Random Forest	0.75	0.38	1.00	0.11	1.00	0.91	0.56	0.91
60 s./60 m. Ada.	0.66	0.08	0.57	0.74	0.57	0.54	0.77	0.54
Logit	0.72	0.15	0.60	0.77	0.60	0.00	1.00	0.00
Random Forest	0.77	0.43	1.00	0.09	1.00	0.93	0.58	0.93
90 s./10 m. Ada.	0.71	0.14	0.67	0.71	0.67	0.59	0.80	0.59
Logit	0.74	0.19	0.61	0.77	0.61	0.00	1.00	0.00
Random Forest	0.72	0.29	1.00	0.13	1.00	0.88	0.52	0.88
90 s./30 m. Ada.	0.70	0.13	0.66	0.71	0.66	0.64	0.74	0.64
Logit	0.74	0.17	0.60	0.79	0.60	0.00	1.00	0.00
Random Forest	0.72	0.33	1.00	0.11	1.00	0.92	0.50	0.92
90 s./60 m. Ada.	0.69	0.11	0.59	0.77	0.59	0.57	0.78	0.57
Logit	0.73	0.16	0.60	0.79	0.60	0.00	1.00	0.00
Random Forest	0.74	0.37	1.00	0.10	1.00	0.92	0.54	0.93

Table 6 reports the results for the original dataset with pump-and-dumps flagged 1 or 2 min before the public announcement, and it clearly demonstrates the impact of using classifiers with a heavily imbalanced dataset and a traditional threshold of $p = 50\%$ on accuracy and sensitivity. The results show that accuracy is almost perfect, but sensitivity is close to zero, indicating an inability to detect *any* pump-and-dumps. However, by setting the threshold close to the observed prevalence ($p = 0.2\%$), the results significantly improved, with sensitivity exceeding 50% for both the random forest and logit models. AdaBoost, on the other hand, continued to struggle. If we use threshold-independent measures, the random forest model performs the best, while AdaBoost is the worst. Regarding chunk sizes and window sizes, the best results seem to be obtained with a chunk size of 30 s and a window size of 60 min, consistent with the findings reported in La Morgia et al. (2020, 2023).

When considering the original dataset with pump-and-dumps flagged 60 min before the public announcement (Table 9), the performance of both random forest and logit models improved significantly, with the random forest showing an AUC, H-measure, accuracy, sensitivity, and specificity remarkably close to 1 for almost all chunk sizes and window sizes. These metrics are the best across all 10 data configurations considered in this study. AdaBoost remained the worst model, although its metrics improved as well.

If we consider resampling methods to build synthetic balanced datasets with pump-and-dumps flagged 1 or 2 min before the announcement (Tables 7 and 8), we can observe improvements in threshold-dependent metrics with $p = 50\%$, while there are no significant differences in threshold-dependent metrics with $p = 0.18\%$ and threshold-independent metrics compared to the baseline case. However, particularly in the case of random forests together with Random Under-Sampling and ROSE methods, the results can be much worse. On the other hand, the AdaBoost classifier seems to benefit the most from resampling methods.

Table 8. Performance metrics of 5-fold cross-validation. Balanced data created with the Synthetic Minority Over-sampling Technique (SMOTE) and the Random Over-Sampling Examples (ROSE) method, with pump-and-dumps flagged 1 or 2 min before the public announcement.

	Synthetic Minority Over-sampling Technique (SMOTE)-(P&Ds Flagged 1 or 2 min before Announcement)							
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 0.18%)***		
.../Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.69	0.12	0.60	0.78	0.60	0.60	0.78	0.60
Logit	0.74	0.17	0.61	0.75	0.61	0.00	1.00	0.00
Random Forest	0.83	0.50	1.00	0.24	1.00	0.83	0.74	0.83
30 s./30 m. Ada.	0.67	0.09	0.61	0.73	0.60	0.60	0.73	0.60
Logit	0.73	0.15	0.61	0.77	0.61	0.00	1.00	0.00
Random Forest	0.83	0.54	1.00	0.27	1.00	0.87	0.72	0.87
30 s./60 m. Ada.	0.68	0.11	0.61	0.75	0.61	0.56	0.78	0.55
Logit	0.73	0.15	0.61	0.77	0.61	0.00	1.00	0.00
Random Forest	0.85	0.58	1.00	0.24	1.00	0.89	0.75	0.89
60 s./10 m. Ada.	0.67	0.11	0.74	0.57	0.74	0.72	0.61	0.72
Logit	0.73	0.17	0.61	0.75	0.61	0.00	1.00	0.00
Random Forest	0.78	0.36	1.00	0.14	1.00	0.78	0.68	0.78
60 s./30 m. Ada.	0.66	0.10	0.76	0.56	0.76	0.76	0.56	0.76
Logit	0.72	0.15	0.61	0.76	0.61	0.00	1.00	0.00
Random Forest	0.79	0.40	1.00	0.14	1.00	0.82	0.69	0.82
60 s./60 m. Ada.	0.66	0.09	0.66	0.65	0.66	0.66	0.65	0.66
Logit	0.72	0.15	0.60	0.77	0.60	0.00	1.00	0.00
Random Forest	0.79	0.42	1.00	0.12	1.00	0.85	0.67	0.85
90 s./10 m. Ada.	0.69	0.13	0.71	0.64	0.71	0.69	0.67	0.69
Logit	0.74	0.18	0.62	0.76	0.62	0.00	1.00	0.00
Random Forest	0.74	0.30	1.00	0.13	1.00	0.79	0.62	0.79
90 s./30 m. Ada.	0.67	0.12	0.74	0.59	0.74	0.72	0.62	0.72
Logit	0.74	0.17	0.61	0.79	0.61	0.00	1.00	0.00
Random Forest	0.76	0.34	1.00	0.14	1.00	0.82	0.63	0.82
90 s./60 m. Ada.	0.67	0.11	0.71	0.62	0.71	0.71	0.62	0.71
Logit	0.74	0.17	0.61	0.79	0.61	0.00	1.00	0.00
Random Forest	0.76	0.37	1.00	0.11	1.00	0.86	0.60	0.86
	Random Over-Sampling Examples (ROSE)-(P&Ds Flagged 1 or 2 min before Announcement)							
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 0.18%)***		
.../Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.54	0.06	0.98	0.10	0.99	0.98	0.10	0.98
Logit	0.74	0.17	0.60	0.77	0.59	0.00	1.00	0.00
Random Forest	0.55	0.07	1.00	0.08	1.00	0.96	0.14	0.96
30 s./30 m. Ada.	0.53	0.04	0.98	0.08	0.98	0.98	0.09	0.98
Logit	0.73	0.15	0.59	0.78	0.59	0.00	1.00	0.00
Random Forest	0.55	0.06	1.00	0.07	1.00	0.92	0.17	0.92
30 s./60 m. Ada.	0.53	0.02	0.97	0.08	0.97	0.97	0.08	0.97
Logit	0.73	0.15	0.59	0.78	0.59	0.00	1.00	0.00
Random Forest	0.57	0.06	1.00	0.07	1.00	0.83	0.30	0.83
60 s./10 m. Ada.	0.54	0.06	0.99	0.09	0.99	0.99	0.10	0.99
Logit	0.73	0.16	0.59	0.77	0.59	0.00	1.00	0.00
Random Forest	0.55	0.07	1.00	0.08	1.00	0.97	0.12	0.97
60 s./30 m. Ada.	0.54	0.05	0.99	0.09	0.99	0.98	0.09	0.99
Logit	0.72	0.14	0.59	0.77	0.59	0.00	1.00	0.00
Random Forest	0.54	0.06	1.00	0.07	1.00	0.95	0.12	0.95
60 s./60 m. Ada.	0.53	0.03	0.98	0.07	0.98	0.98	0.07	0.98
Logit	0.72	0.14	0.59	0.78	0.59	0.00	1.00	0.00
Random Forest	0.52	0.05	1.00	0.07	1.00	0.92	0.12	0.92
90 s./10 m. Ada.	0.55	0.07	0.99	0.10	0.99	0.99	0.11	0.99
Logit	0.74	0.18	0.59	0.80	0.59	0.00	1.00	0.00
Random Forest	0.55	0.08	1.00	0.09	1.00	0.97	0.12	0.98
90 s./30 m. Ada.	0.54	0.05	0.99	0.08	0.99	0.98	0.09	0.99
Logit	0.73	0.16	0.59	0.79	0.59	0.00	1.00	0.00
Random Forest	0.54	0.06	1.00	0.07	1.00	0.96	0.11	0.96
90 s./60 m. Ada.	0.53	0.04	0.98	0.08	0.99	0.98	0.08	0.98
Logit	0.73	0.16	0.59	0.80	0.59	0.00	1.00	0.00
Random Forest	0.53	0.05	1.00	0.07	1.00	0.94	0.12	0.94

If we consider resampling methods to build synthetic balanced datasets with pump-and-dumps flagged 60 min before the announcement (Tables 10 and 11), the results are generally better than the case with pump-and-dumps flagged 1 or 2 min before the announcement, with the exception of the ROSE method, which continues to show rather poor results. However, there are again no qualitative differences with the baseline case without resampling methods, which is worse only if threshold-dependent metrics with $p = 50\%$ are considered.

Table 9. Performance metrics of 5-fold cross-validation. Original data with pump-and-dumps flagged 60 min before the public announcement.

(6) ORIGINAL DATA (P&Ds Flagged 60 min before Announcement)								
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 2.6%)***		
…/Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.51	0.01	0.97	0.01	1.00	0.97	0.01	1.00
Logit	0.71	0.13	0.97	0.00	1.00	0.58	0.77	0.57
Random Forest	0.98	0.83	0.99	0.44	1.00	0.87	0.96	0.87
30 s./30 m. Ada.	0.51	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.71	0.14	0.97	0.00	1.00	0.58	0.78	0.58
Random Forest	0.99	0.91	0.99	0.63	1.00	0.91	0.98	0.91
30 s./60 m. Ada.	0.50	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.72	0.14	0.97	0.00	1.00	0.59	0.79	0.58
Random Forest	0.99	0.94	0.99	0.73	1.00	0.93	0.99	0.93
60 s./10 m. Ada.	0.50	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.71	0.13	0.97	0.00	1.00	0.58	0.78	0.58
Random Forest	0.96	0.75	0.98	0.27	1.00	0.83	0.94	0.83
60 s./30 m. Ada.	0.50	0.01	0.97	0.01	1.00	0.97	0.01	1.00
Logit	0.72	0.14	0.97	0.00	1.00	0.58	0.78	0.58
Random Forest	0.99	0.87	0.99	0.52	1.00	0.88	0.97	0.88
60 s./60 m. Ada.	0.50	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.72	0.14	0.97	0.00	1.00	0.59	0.80	0.58
Random Forest	0.99	0.91	0.99	0.62	1.00	0.90	0.98	0.90
90 s./10 m. Ada.	0.50	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.71	0.14	0.97	0.00	1.00	0.59	0.77	0.59
Random Forest	0.94	0.70	0.98	0.20	1.00	0.81	0.92	0.81
90 s./30 m. Ada.	0.50	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.72	0.14	0.97	0.00	1.00	0.60	0.78	0.59
Random Forest	0.97	0.82	0.98	0.39	1.00	0.86	0.95	0.85
90 s./60 m. Ada.	0.50	0.01	0.97	0.00	1.00	0.97	0.01	1.00
Logit	0.72	0.15	0.97	0.00	1.00	0.60	0.79	0.59
Random Forest	0.99	0.89	0.99	0.54	1.00	0.89	0.98	0.89

The main result that emerges from this analysis is that the most effective approach to detecting pumps-and-dumps is by using the random forest model with a small chunk size (maximum of 30 s) and a window size of at least one hour, while setting the threshold close to the observed prevalence in the dataset for pump-and-dumps flagged 60 min before the announcement. Although resampling methods may be useful in some cases, particularly when using threshold-dependent metrics with $p = 50\%$, threshold-independent measures were not affected, and a better balance between sensitivity and specificity can be obtained by simply choosing a proper probability threshold. Moreover, detecting pump-and-dumps in real-time involves high-dimensional, high-frequency data, and the use of resampling methods to build synthetic datasets could be time-consuming, making such methods less practical. Additionally, implementing these methods for real-time detection would require a significant investment in hardware, which may not be feasible for crypto-exchanges and financial regulators due to the associated high financial costs.

Finally, we remark that our empirical analysis indirectly confirms the large-scale Monte Carlo simulations and case study in the medical field recently reported by van den Goorbergh et al. (2022), who showed that using Random Under-Sampling, Random Over-Sampling, and SMOTE methods resulted in poorly calibrated models that did not provide higher areas under the ROC curve compared to models developed without correction for class imbalance. Furthermore, they also showed that a better balance between sensitivity and specificity can be achieved by simply adjusting the probability threshold.

Table 10. Performance metrics of 5-fold cross-validation. Balanced data created with Random Under-Sampling (RUS) and Random Over-Sampling (ROS), with pump-and-dumps flagged 60 min before the public announcement.

RANDOM UNDER-SAMPLING (P&Ds Flagged 60 min before Announcement)								
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 2.6%)***		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.70	0.13	0.73	0.61	0.73	0.57	0.78	0.56
Logit	0.70	0.13	0.58	0.77	0.58	0.03	1.00	0.00
Random Forest	0.96	0.70	0.85	0.92	0.85	0.11	1.00	0.09
30 s./30 m. Ada.	0.72	0.14	0.70	0.64	0.70	0.53	0.82	0.52
Logit	0.71	0.13	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.98	0.81	0.90	0.95	0.90	0.16	1.00	0.14
30 s./60 m. Ada.	0.72	0.15	0.70	0.65	0.70	0.54	0.83	0.54
Logit	0.71	0.14	0.60	0.77	0.60	0.03	1.00	0.00
Random Forest	0.99	0.86	0.93	0.96	0.93	0.21	1.00	0.19
60 s./10 m. Ada.	0.71	0.14	0.68	0.67	0.68	0.55	0.82	0.54
Logit	0.71	0.13	0.58	0.77	0.58	0.03	1.00	0.00
Random Forest	0.93	0.60	0.80	0.90	0.80	0.08	1.00	0.06
60 s./30 m. Ada.	0.73	0.15	0.59	0.79	0.58	0.46	0.89	0.45
Logit	0.71	0.14	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.97	0.75	0.87	0.94	0.87	0.11	1.00	0.08
60 s./60 m. Ada.	0.73	0.15	0.65	0.71	0.65	0.47	0.89	0.46
Logit	0.72	0.14	0.60	0.77	0.60	0.03	1.00	0.00
Random Forest	0.98	0.81	0.90	0.95	0.90	0.14	1.00	0.11
90 s./10 m. Ada.	0.73	0.15	0.58	0.79	0.57	0.49	0.88	0.48
Logit	0.71	0.13	0.59	0.78	0.58	0.03	1.00	0.00
Random Forest	0.91	0.53	0.77	0.88	0.77	0.07	1.00	0.05
90 s./30 m. Ada.	0.72	0.14	0.61	0.76	0.61	0.46	0.90	0.45
Logit	0.72	0.14	0.59	0.78	0.59	0.03	1.00	0.00
Random Forest	0.95	0.67	0.83	0.92	0.83	0.09	1.00	0.06
90 s./60 m. Ada.	0.72	0.14	0.61	0.76	0.61	0.50	0.88	0.49
Logit	0.72	0.14	0.61	0.78	0.60	0.03	1.00	0.00
Random Forest	0.97	0.76	0.87	0.94	0.87	0.11	1.00	0.09
RANDOM OVER-SAMPLING (P&Ds Flagged 60 min before Announcement)								
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 2.6%)***		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.67	0.12	0.75	0.58	0.75	0.75	0.59	0.75
Logit	0.71	0.13	0.58	0.77	0.58	0.03	1.00	0.00
Random Forest	0.98	0.79	0.99	0.65	1.00	0.61	0.99	0.60
30 s./30 m. Ada.	0.67	0.12	0.75	0.58	0.76	0.74	0.60	0.74
Logit	0.71	0.13	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.99	0.89	0.99	0.77	1.00	0.70	0.99	0.70
30 s./60 m. Ada.	0.69	0.14	0.74	0.60	0.74	0.70	0.64	0.70
Logit	0.72	0.14	0.60	0.77	0.60	0.03	1.00	0.00
Random Forest	0.99	0.93	1.00	0.83	1.00	0.78	1.00	0.78
60 s./10 m. Ada.	0.70	0.12	0.64	0.71	0.64	0.61	0.75	0.61
Logit	0.71	0.13	0.58	0.77	0.58	0.03	1.00	0.00
Random Forest	0.96	0.73	0.99	0.50	1.00	0.58	0.98	0.57
60 s./30 m. Ada.	0.70	0.12	0.54	0.84	0.53	0.52	0.85	0.51
Logit	0.71	0.14	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.98	0.87	0.99	0.69	1.00	0.72	0.99	0.72
60 s./60 m. Ada.	0.69	0.12	0.58	0.78	0.58	0.58	0.79	0.57
Logit	0.72	0.14	0.60	0.78	0.60	0.03	1.00	0.00
Random Forest	0.99	0.90	0.99	0.76	1.00	0.74	0.99	0.73
90 s./10 m. Ada.	0.69	0.11	0.58	0.79	0.57	0.57	0.80	0.56
Logit	0.71	0.13	0.59	0.78	0.58	0.03	1.00	0.00
Random Forest	0.95	0.67	0.98	0.39	1.00	0.56	0.98	0.55
90 s./30 m. Ada.	0.69	0.11	0.60	0.76	0.60	0.58	0.80	0.57
Logit	0.72	0.14	0.60	0.77	0.59	0.03	1.00	0.00
Random Forest	0.97	0.81	0.99	0.59	1.00	0.65	0.98	0.64
90 s./60 m. Ada.	0.70	0.12	0.56	0.82	0.56	0.55	0.84	0.54
Logit	0.72	0.14	0.61	0.78	0.60	0.03	1.00	0.00
Random Forest	0.98	0.88	0.99	0.71	1.00	0.73	0.99	0.72

Table 11. Performance metrics of 5-fold cross-validation. Balanced data created with the Synthetic Minority Over-sampling Technique (SMOTE) and the Random Over-Sampling Examples (ROSE) method, with pump-and-dumps flagged 60 min before the public announcement.

Synthetic Minority Over-sampling Technique (SMOTE) (P&Ds Flagged 60 min before Announcement)								
Chunk/Window	*Threshold Independent*		*Threshold Dependent (p = 50%)*			*Threshold Dependent (p = 2.6%)*		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.70	0.12	0.63	0.70	0.63	0.55	0.79	0.54
Logit	0.71	0.13	0.58	0.77	0.58	0.03	1.00	0.00
Random Forest	0.97	0.75	0.98	0.72	0.99	0.53	0.99	0.52
30 s./30 m. Ada.	0.68	0.12	0.75	0.58	0.76	0.71	0.62	0.71
Logit	0.71	0.13	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.99	0.86	0.99	0.82	1.00	0.62	0.99	0.61
30 s./60 m. Ada.	0.68	0.12	0.74	0.60	0.74	0.70	0.64	0.70
Logit	0.71	0.14	0.61	0.78	0.60	0.03	1.00	0.00
Random Forest	0.99	0.91	0.99	0.88	1.00	0.68	1.00	0.67
60 s./10 m. Ada.	0.67	0.12	0.76	0.57	0.76	0.76	0.57	0.76
Logit	0.71	0.13	0.58	0.77	0.58	0.03	1.00	0.00
Random Forest	0.94	0.64	0.97	0.58	0.98	0.47	0.98	0.46
60 s./30 m. Ada.	0.69	0.11	0.58	0.79	0.57	0.57	0.80	0.57
Logit	0.71	0.14	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.98	0.81	0.99	0.73	1.00	0.58	0.99	0.56
60 s./60 m. Ada.	0.69	0.13	0.68	0.68	0.68	0.62	0.73	0.62
Logit	0.72	0.14	0.61	0.78	0.60	0.03	1.00	0.00
Random Forest	0.99	0.86	0.99	0.81	1.00	0.61	1.00	0.60
90 s./10 m. Ada.	0.67	0.12	0.76	0.57	0.76	0.74	0.59	0.74
Logit	0.71	0.13	0.59	0.77	0.58	0.03	1.00	0.00
Random Forest	0.92	0.58	0.97	0.48	0.98	0.45	0.98	0.43
90 s./30 m. Ada.	0.68	0.12	0.66	0.68	0.66	0.63	0.72	0.63
Logit	0.72	0.14	0.60	0.77	0.59	0.03	1.00	0.00
Random Forest	0.96	0.74	0.98	0.65	0.99	0.51	0.99	0.50
90 s./60 m. Ada.	0.70	0.12	0.64	0.71	0.64	0.59	0.77	0.59
Logit	0.72	0.14	0.61	0.78	0.60	0.03	1.00	0.00
Random Forest	0.98	0.81	0.99	0.75	0.99	0.57	0.99	0.56
Random Over-Sampling Examples (ROSE) (P&Ds Flagged 60 min before Announcement)								
Chunk/Window	*Threshold Independent*		*Threshold Dependent (p = 50%)*			*Threshold Dependent (p = 2.6%)*		
. . . /Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./10 m. Ada.	0.50	0.00	0.92	0.05	0.94	0.91	0.06	0.94
Logit	0.70	0.12	0.58	0.77	0.57	0.03	1.00	0.00
Random Forest	0.66	0.08	0.96	0.06	0.98	0.34	0.87	0.33
30 s./30 m. Ada.	0.50	0.00	0.94	0.05	0.96	0.93	0.05	0.96
Logit	0.71	0.13	0.59	0.77	0.58	0.03	1.00	0.00
Random Forest	0.73	0.14	0.86	0.30	0.88	0.13	0.99	0.10
30 s./60 m. Ada.	0.50	0.00	0.93	0.05	0.95	0.92	0.05	0.94
Logit	0.71	0.13	0.59	0.76	0.59	0.03	1.00	0.00
Random Forest	0.76	0.17	0.61	0.80	0.61	0.06	1.00	0.04
60 s./10 m. Ada.	0.50	0.00	0.94	0.04	0.97	0.93	0.04	0.95
Logit	0.71	0.13	0.58	0.78	0.58	0.03	1.00	0.00
Random Forest	0.55	0.02	0.97	0.03	0.99	0.76	0.33	0.77
60 s./30 m. Ada.	0.50	0.00	0.94	0.04	0.97	0.93	0.04	0.95
Logit	0.71	0.13	0.59	0.77	0.58	0.03	1.00	0.00
Random Forest	0.66	0.08	0.96	0.05	0.99	0.37	0.87	0.36
60 s./60 m. Ada.	0.50	0.00	0.93	0.04	0.95	0.93	0.04	0.95
Logit	0.71	0.14	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.73	0.14	0.91	0.19	0.93	0.16	0.98	0.13
90 s./10 m. Ada.	0.50	0.00	0.95	0.02	0.98	0.94	0.03	0.97
Logit	0.71	0.13	0.58	0.78	0.58	0.03	1.00	0.00
Random Forest	0.52	0.01	0.97	0.02	0.99	0.83	0.19	0.85
90 s./30 m. Ada.	0.50	0.00	0.94	0.04	0.96	0.93	0.04	0.96
Logit	0.71	0.14	0.59	0.78	0.59	0.03	1.00	0.00
Random Forest	0.62	0.05	0.97	0.03	0.99	0.54	0.68	0.54
90 s./60 m. Ada.	0.50	0.00	0.93	0.04	0.96	0.93	0.04	0.96
Logit	0.72	0.14	0.59	0.77	0.59	0.03	1.00	0.00
Random Forest	0.70	0.10	0.95	0.08	0.98	0.25	0.95	0.23

5. A Robustness Check: Transforming the Data Using a Generalized Box–Cox Tranformation

Even though all the regressors are stationary, some of them exhibit a large variability. Therefore, we investigated how our previous results would change by transforming the data to stabilize their variance. To achieve this, we transformed all the regressors (except for the four periodic functions of time) using the generalized version of the Box–Cox transformation proposed by Hawkins and Weisberg (2017). This method allows the inclusion of zeros and nonpositive values in the transformation. Their approach uses the original

Box–Cox family of scaled power transformations, given by $(w^{\lambda} - 1)/\lambda$ for $\lambda \neq 0$ and $\log(w)$ for $\lambda = 0$, on the following transformed data:

$$w = 0.5(y + \sqrt{y^2 + \gamma^2})$$

where y is the original data and λ and γ are either user-selected parameters or are estimated; see Hawkins and Weisberg (2017) for the full details on this transformation[6].

The performance metric differences of the 5-fold cross-validation between the transformed datasets and the original datasets, with pump-and-dumps flagged 1 or 2 min before the public announcement, are reported in Table 12.

Table 12. Performance metric differences of the 5-fold cross-validation between the transformed dataset and the original dataset with pump-and-dumps flagged 1 or 2 min before the public announcement.

(1) ORIGINAL DATA (P&Ds Flagged 1 or 2 min before Announcement)								
Chunk/Window	***Threshold Independent***		***Threshold Dependent (p = 50%)***			***Threshold Dependent (p = 0.18%)***		
…/Model	**AUC**	**H-Measure**	**Accuracy**	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
30 s./ 10 m. Ada.	**0.00**	**0.01**	0.00	0.00	0.00	0.00	0.01	0.00
Logit	−0.02	−0.03	0.00	0.01	0.00	−0.03	−0.02	-0.03
Random Forest	0.01	0.02	0.00	0.01	0.00	0.00	0.03	0.00
30 s./30 m. Ada.	−0.01	−0.01	0.00	0.00	0.00	0.00	−0.01	0.00
Logit	0.00	0.00	0.00	0.00	0.00	−0.02	−0.03	−0.02
Random Forest	0.00	−0.01	0.00	0.00	0.00	0.00	−0.01	0.00
30 s./60 m. Ada.	−0.01	−0.01	0.00	0.00	0.00	0.00	−0.01	0.00
Logit	0.00	0.00	0.00	0.00	0.00	0.00	−0.02	0.00
Random Forest	0.01	0.02	0.00	0.01	0.00	0.00	0.02	0.00
60 s./10 m. Ada.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Logit	−0.03	−0.04	0.00	0.02	0.00	−0.07	0.06	−0.07
Random Forest	0.01	0.02	0.00	0.00	0.00	0.00	0.02	0.00
60 s./30 m. Ada.	0.00	0.00	0.00	−0.02	0.00	0.00	0.00	0.00
Logit	−0.01	−0.01	0.00	0.01	0.00	−0.03	0.05	−0.03
Random Forest	−0.01	−0.02	0.00	0.01	0.00	0.00	−0.01	0.00
60 s./60 m. Ada.	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00
Logit	0.00	0.00	0.00	0.01	0.00	−0.01	0.03	−0.01
Random Forest	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00
90 s./10 m. Ada.	0.00	−0.01	0.00	0.00	0.00	0.00	−0.01	0.00
Logit	−0.03	−0.05	0.00	0.03	0.00	−0.07	0.06	−0.07
Random Forest	0.00	−0.01	0.00	0.01	0.00	0.00	−0.01	0.00
90 s./30 m. Ada.	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Logit	−0.02	−0.02	0.00	0.01	0.00	−0.05	0.04	−0.05
Random Forest	0.02	0.03	0.00	0.00	0.00	0.00	0.04	0.00
90 s./60 m. Ada.	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
Logit	−0.01	−0.01	0.00	0.01	0.00	−0.02	0.03	−0.02
Random Forest	0.03	0.04	0.00	0.00	0.00	0.00	0.05	0.00

The differences are minimal and almost zero in all cases, indicating that using transformed regressors does not improve the classification process. This is a positive result, as computing the generalized Box–Cox transformation with high-dimensional and high-frequency data can be computationally demanding. The performance metric differences in the case of resampling methods used to build synthetic balanced data deliver qualitatively similar results. Therefore, for the sake of interest and space, they are not reported here.

6. Discussion and Conclusions

Detecting pump-and-dump schemes in the cryptocurrency market using high-frequency data is a daunting task due to the rarity of pump-and-dump cases in these datasets, which typically account for less than 0.2% of all cases. Additionally, despite VIP members of social media groups organizing these market manipulations, supposedly receiving detailed information about them seconds or minutes before the general public, unusual trading volumes can occur much earlier, which complicates the detection process. These challenges significantly affect the performance of any machine learning classifier employed to identify pump-and-dump activities, resulting in the difficulty of accurately detecting such instances in a timely manner.

To address the first problem, we proposed building synthetic balanced datasets generated according to four resampling methods for class imbalances proposed in the statistical literature. To address the second problem, we suggested flagging a pump-and-dump from the minute of the public announcement up to 60 min before it. This approach can deal with the insiders' anticipated purchases of targeted crypto-assets and the issue of imbalanced datasets.

To verify the validity of our proposals, we collected data about pump-and-dump schemes from Pumpolymp, which is a website that provides detailed information about all Telegram groups dealing with crypto pump-and-dumps, and managed to extract 351 pump signals relative to the Binance crypto exchange that took place in 2021 and 2022. The historical transaction level data of the identified pumped coins were obtained using the CryptoCurrency eXchange Trading Library (CCXT). We used various types of regressors in conjunction with three distinct classifiers (logistic regression, random forest, and the AdaBoost classifier) to identify the onset of a pump-and-dump scheme.

Our empirical analysis showed that the most effective approach to detecting pump-and-dumps was using the original imbalanced dataset with pump-and-dumps flagged 60 min in advance, together with a random forest model with data segmented into 30-s chunks and regressors computed with a moving window of 1 h. We also found that a better balance between sensitivity and specificity could be achieved by merely adjusting the probability threshold, such as setting the threshold close to the observed prevalence in the original dataset. Although resampling methods were useful in some cases, particularly when using threshold-dependent metrics with a probability level equal to $p = 50\%$, threshold-independent measures were not affected, and in some cases they even provided worse AUC and H-measures compared to models developed without correction for class imbalance. Thus, our empirical evidence confirmed the large-scale Monte Carlo simulations performed by van den Goorbergh et al. (2022) who showed that using Random Under-Sampling, Random Over-Sampling, and SMOTE methods resulted in poorly calibrated models.

We highlight that detecting pump-and-dumps in real-time involves high-dimensional data, and the use of resampling methods to build synthetic datasets could be time-consuming, making such methods less practical. Implementing these methods for real-time detection would require a significant investment in hardware, which may not be feasible for crypto-exchanges and financial regulators due to the associated high financial costs.

Finally, we performed a set of robustness checks to verify that our results also held when transforming the data to stabilize their variance using a generalized version of the Box–Cox transformation to allow the inclusion of zeros and non-positive values. The differences were minimal, indicating that using transformed regressors did not improve the classification process.

We remark that we selected thresholds close to the observed prevalence because this is often the optimal choice for datasets with very low prevalence. However, the optimal threshold ultimately depends on the cost function employed by the model user. We leave a detailed analysis of the methods to find the optimal threshold to detect pump-and-dumps as an avenue for future research.

In the context of pump-and-dump scheme detection, it is important to acknowledge the potential for incorrect identifications and the impact this can have on the application of resampling methods to build synthetic balanced datasets. Our study assumes that all pump-and-dump schemes have been accurately identified through a separate empirical strategy. However, there is a possibility, if not likelihood, of noise or misclassifications in this higher-level identification process. The presence of incorrect identifications introduces a source of bias into the training sample used for model development. If the identification process exhibits a significant Type 1 error (false positive rate), the utilization of synthetic samples could potentially amplify this bias. This can be particularly problematic in cases of class imbalance, where the minority class (pump-and-dump schemes) may not be properly identified, resulting in an underrepresentation of the true positive

cases. It is crucial to address this issue in future research. Conducting experiments on simulated data could provide valuable insights into the impact of misclassifications on the effectiveness of resampling methods. By intentionally introducing noise or biases in the identification process, researchers can examine how the use of synthetic samples exacerbates or mitigates these issues. Additionally, exploring alternative strategies for accurate identification of pump-and-dump schemes and assessing their influence on resampling techniques can contribute to the development of more robust and reliable models in the detection of this financial fraud. The robust methods and the large simulation studies in Bunkhumpornpat et al. (2009) and Barua et al. (2012) can provide valuable starting points.

Author Contributions: Conceptualization, D.F. and Y.X.; methodology, D.F. and Y.X.; software, D.F. and Y.X.; validation, D.F. and Y.X.; formal analysis, D.F.; investigation, D.F. and Y.X.; resources, D.F. and Y.X.; data curation, D.F. and Y.X.; writing—original draft preparation, D.F. and Y.X.; writing—review and editing, D.F.; visualization, D.F. and Y.X.; supervision, D.F.; project administration, D.F.; funding acquisition, D.F. All authors have read and agreed to the published version of the manuscript.

Funding: The first-named author gratefully acknowledges financial support from the grant of the Russian Science Foundation, n. 20-68-47030.

Data Availability Statement: The data are available from the authors upon request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The RUS method reduces the number of instances in the majority class by randomly sampling from it until it matches the number of samples in the minority class. Suppose we have a dataset of n observations with a binary outcome variable Y that takes values 0 or 1, where $Y = 1$ denotes the event of interest. Let p be the proportion of observations with $Y = 1$, so that $p = n_1/n$, where n_1 is the number of observations with $Y = 1$. Typically, p is much smaller than 0.5, indicating class imbalance. The RUS method involves randomly selecting a subset of observations with $Y = 0$ to match the number of observations with $Y = 1$. Let n_0 be the number of observations with $Y = 0$, so that $n_0 = n - n_1$. Then, we randomly select n_1 observations from the n_0 observations with $Y = 0$ to create a balanced dataset of $2n_1$ observations. RUS can be an effective method for addressing class imbalance, particularly when the dataset is large enough so that randomly selecting a subset of observations with $Y = 0$ does not result in loss of information. However, it may not work well if the dataset is small or if the minority class contains important information that is lost during under-sampling.

The ROS method involves randomly replicating observations with $Y = 1$ to create a balanced dataset. Specifically, we randomly select $n_0 - n_1$ observations with replacement from the original n_1 observations with $Y = 1$ until we have a total of n_0 observations with $Y = 1$. This results in a dataset where the number of observations with $Y = 0$ is equal to the number of observations with $Y = 1$. ROS can be an effective method for addressing class imbalance, particularly when the dataset is small or when the minority class contains important information that should not be lost during under-sampling. However, it may not work well if the minority class contains outliers or noisy data.

SMOTE is a data augmentation method that creates synthetic minority class observations by interpolating between existing minority class observations. The basic idea is to identify each minority class observation and then create new synthetic observations by interpolating between it and its k nearest minority class neighbors. The SMOTE algorithm is performed through the following steps:

1. Identify the minority class observations with $Y = 1$.
2. Randomly select a minority class observation x_i from the dataset.
3. Find the k nearest minority class neighbors of x_i in the feature space, where k is a user-defined parameter.

4. Create synthetic observations: for each nearest neighbor x_j, generate a synthetic observation $x_{i,new}$ by randomly interpolating between x_i and x_j in the feature space:

$$x_{i,new} = x_i + rand(0,1) \cdot (x_j - x_i)$$

   where $rand(0,1)$ is a random number between 0 and 1.
5. Repeat steps 2–4 until the desired number of synthetic observations have been generated.

SMOTE can be an effective method for addressing class imbalance, particularly when the dataset is small and when the minority class is not close to the majority class in the feature space. However, it may not work well if the minority class is sparse or if there is high overlap between the minority and majority classes in the feature space. Additionally, the choice of k can impact the effectiveness of the algorithm, as larger values of k can result in more diverse synthetic observations, but can also increase the risk of introducing noise or outliers.

The ROSE technique aims to address class imbalance by generating artificial samples from the feature space neighbourhood around the minority class. This technique combines over-sampling and under-sampling, and comprises four steps:

1. Resample the majority class data using a bootstrap resampling technique to reduce the number of majority class samples to a ratio of 50% via under-sampling.
2. Resample the minority class data using a bootstrap resampling technique to increase the number of minority class samples to a ratio of 50% via over-sampling.
3. Combined the data from steps 1 and 2 to create a new training sample.
4. Generate new synthetic data for both the majority and minority classes in their respective neighborhoods. The neighborhood shape is defined by the kernel density function with a Gaussian kernel $K_{\mathbf{H}_j}$ and a smoothing matrix $\mathbf{H}_j$, $j = 0, 1$, with a d dimension (where d is the number of independent variables), where $\mathbf{H}_j = diag(h_1^{(j)}, \ldots, h_d^{(j)})$ and $h_d^{(j)}$ is defined as follows:

$$h_q^{(j)} = \left(\frac{4}{(d+2)n} \right)^{1/(d+4)} \cdot \hat{\sigma}_q^{(j)}, \quad q = 1, \ldots, d.$$

   Here, $\hat{\sigma}_q^{(j)}$ is the standard deviation of the q-th dimension of the observations belonging to a given class.

These four steps are repeated for each training sample to produce a new synthetic training sample of approximately equal size as the original dataset, with the number of instances for both classes being equally represented. The ROSE technique represents an effective method for generating synthetic samples, as it leverages the available information to synthesize new samples that are more representative of the minority class. Moreover, using the synthetic generation of new examples in ROSE strengthens the learning process and estimates the distribution of the chosen measure of accuracy. The artificial training set can be used to estimate the classification model, while the originally observed data are reserved for testing the classifier. Alternatively, cross-validation or smoothed bootstrap methods can be used. Creating new artificial examples from an estimate of the conditional densities of the two classes allows for overcoming the limits of both the apparent error and the holdout method, which are not advisable in extreme imbalanced learning due to the scarcity of rare class data.