Manipulation Detection in Cryptocurrency Markets: An Anomaly and Change Detection Based Approach

Olaf Kampers*
Utrecht University
Utrecht, Netherlands
olaf.kampers96@gmail.com

Abdulhakim Qahtan
Utrecht University
Utrecht, Netherlands
a.a.a.qahtan@uu.nl

Swati Mathur
ChainSlayer, Utrecht
Netherlands
swati@chainslayer.io

Yannis Velegrakis
Utrecht University and University of Trento
Utrecht, Netherlands
i.velegrakis@uu.nl

ABSTRACT

As a financial asset, cryptocurrencies innovated the financial industry in different ways. However, the lack of regulations and transparency in cryptocurrency markets is hindering the industry from reaching its full potential. There is a need for extensive technical analysis of the cryptocurrency market data to detect possible market manipulation attempts. Anomaly detection techniques can reveal information about abnormal activities in the market and provide insights on manipulation attempts. In this study, a robust unsupervised anomaly detection tool (ADT) is developed for this purpose. Experiments show that ADT outperforms a set of methods in detecting the anomalies in features extracted from the cryptocurrency exchanges data and on a set of benchmark data sets.

ACM Reference Format:
Olaf Kampers*, Abdulhakim Qahtan, Swati Mathur, and Yannis Velegrakis. 2022. Manipulation Detection in Cryptocurrency Markets: An Anomaly and Change Detection Based Approach. In *Proceedings of ACM SAC Conference (SAC'22).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.1145/3477314.3507185

1 INTRODUCTION

Cryptocurrencies are digital assets, which are mostly used as a secure medium of exchange. Customers can trade cryptocurrencies with each other using online exchanges in the absence of centralized authorities [16]. Cryptocurrencies use the blockchain technology to combat fraud by removing the need of intermediaries; however, the actual trading process is extremely susceptible to fraud.

Analyzing the cryptocurrency market data to discover possible market manipulation can give traders and institutional investors insights about which cryptocurrency assets they should buy/sell and when, with the ultimate goal to make a profit or at least minimize potential loss. In this paper, we provide an informative discussion about the data in the cryptocurrency exchanges and how to extract important features. Based on that, we propose an accurate and robust anomaly detection tool (ADT) over data streams. The tool combines two well-known methods to discover abnormal activities in the cryptocurrency market: i) KDE-Track [24], which is based on the intuition that values in sparse regions are highly likely to be outliers; ii) isolation-based anomaly detection [15], where anomalies are assumed to be isolated easily from the rest of the points. That is, in an isolation forest, the average path length from the root of the trees to the leaf with anomaly is shorter than the paths to normal data values. Combining both methods improves the robustness and the accuracy of the proposed method. Moreover, we propose a method to adjust the threshold for detecting the anomalies dynamically, which minizes the user involvement and the required prior knowledge about the data. In order to capture the trends in the data stream, ADT utilizes a batch window scenario.

* Work done while interning at ChainSlayer.

SAC'22, April 25 –April 29, 2022, Brno, Czech Republic
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8713-2/22/04…$15.00
https://doi.org/10.1145/3477314.3507185

2 RELATED WORK

Several studies [17, 18, 21] on anomaly detection in the cryptocurrency field focus on the Bitcoin Network specifically. The Bitcoin network only tracks Bitcoin transactions between different wallets. In Monamo et al. [18], a method that is based on K-Means clustering has been used to detect anomalies. Kamps et al. [12] focus on detecting Pump-and-Dump events in the cryptocurrency market data. The co-occurrence of a price and a volume anomaly with addition to some contextual information (e.g. low market cap, type of trading pair) determine whether a data point could indicate a pump-and-dump event. The Pump-and-Dump approach is also used in Limelight [19].

A lot of techniques have been developed for detecting the outliers (anomalies) in the data. These techniques differ in the intuition and the way in which they report the outliers. Anomaly detection approaches include statistical-based [9], distance-based [13], density-based [5, 11, 20], and isolation-based approaches [15]. The main problem in these approaches is the requirement for prior knowledge about the application and the data to accurately set the parameters. The high computational time for distance and density based outlier detection approaches is another issue.

Concept drift, which refers to unpredictable changes in the distribution of the data [10], is also related to our work as dynamic changes also occur in the cryptocurrency market data. These changes may happen abruptly, by changing from one concept to another, or incrementally consisting of many intermediate concepts in between. Concept drift is unavoidable for complex processes in the real world applications [25]. Detecting the drift in the concept can help in adapting the application to the new concept by rebuilding/retraining the data mining models.

3 CRYPTOCURRENCY ASSETS AND EXCHANGES

In this Section, we give a brief description about cryptocurrencies and their markets to understand how market manipulation can be done. During the last decade, a set of cryptocurrency assets have emerged where people can trade these assets with each other on different online trading platforms called *exchanges* [16]. In most cases, users need to send their assets from their own wallets to these exchanges before they can trade them. Exchanges use an *order book* to keep track of the supply and demand of an asset.

Order Books: the *limit order book* [26] is a list of trade orders placed by the traders. A trader can place a *limit sell/buy order* if they want to sell/buy a certain amount (volume) of an asset at a given price. Typically, a matching engine is used to match orders together and (partially) execute them. Impatient traders who do not want to wait until the price of a stock reaches a certain value can sell/buy immediately by placing a *Market Sell/Buy Order*. The matching engine will then choose the stock with either the lowest ask price or highest bid price, depending on the order. The distance between the highest bid and lowest ask price is called the *spread* where the size of the spread is a measure of *market liquidity* [6].

Since traders can place orders at their desired prices, a large number of orders might not be satisfied within a reasonable time interval. This would result in a very large order book. For example, if a trader placed a sell order for Bitcoin in December 2017 at a price of \$21000, then the order would wait for 3 years until December 2020 to get satisfied unless it gets deleted by the user. Such orders have small contribution for short term price movements. However, traders with massive supplies of coins can easily manipulate the market. We believe that analyzing the transactions in the order books could lead to discovering possible market manipulation attempts.

Data Description: most cryptocurrency exchanges provide Application Programming Interfaces (APIs) that allow users to retrieve trading and order book data about different assets. However, there are a number of restrictions that should be kept in mind: 1) It is difficult to track individual orders. Exchanges do not provide information about who placed a given order. Moreover, individual orders are aggregated at each price level. 2) Only a limited number of order book entries can be requested. This limit can be different from an exchange to another.

The cryptocurrency trading data from different exchanges is collected and stored on AWS cloud storage. The states of the order books are pulled every 12 seconds where the changes in the order book data are expressed using *order book events*. There are three types of events for both buy and sell transactions, which are *created event, deleted event* and *updated event*. These events basically show how the order book changes over time. To summarize the data for further analysis, the data is processed to extract a set of features that includes: 1) maximum, minimum and average volume; 2) maximum, minimum and average price; and 3) total volume and the total number of event objects. Event objects are created whenever the current state of the order book is different from the previous one. The goal of the 8 features is to have a concise summary of the changes happening in the order book.

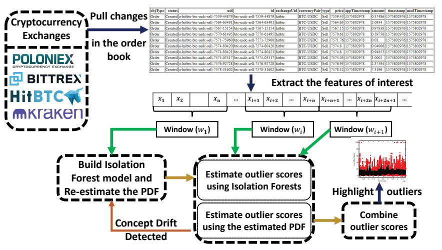
Market Manipulation: market manipulation encompasses different strategies [1, 3] that include *action-based manipulation* (e.g., charging less monthly taxes on electrical cars attract people to buy them), *information-based manipulation*, and *trade-based manipulation*. Our interest is in the trade-based market manipulation techniques that are affecting the trading data. Trade-based techniques include *spoofing* [7] (submitting a set of orders into the market without the intention of the orders to be executed) and *wash trading* [8] (the same group of collusive clients are active on both the sell and buy sides of the order book).

Problem Statement: data anomalies refer to data instances or patterns that do not comply with the 'normal' behavior of the data [13]. Anomalies can be categorized into [2] *Point anomaly*, *Contextual anomaly*, and *Collective anomalies*. In this paper, our main focus is to detect contextual anomalies. We believe that trade-based market manipulation attempts would appear as outliers compared to the normal values in the same context. The problem that we are studying can be formalized as follows: Given a time series $X = \{x_{i+1}, ..., x_{i+n}\}$, compute a score $s(x_j), \quad j = i+1, ..., n$ for every element in X, such that $s(x_j)$ is small if element x_j is highly likely to be an outlier and large if x_j is highly likely to be a normal value. The values in X can be sorted according to their outlierness score s and the top k values are returned as outliers. Another approach is to use a user provided threshold τ to determine the labels of the values such x_i is considered an outlier if $s(x_i) < \tau$ and normal value if $s(x_i) \geq \tau$. Providing the threshold τ requires knowledge about the data and the scoring mechanism, which is not easy for typical users. In this work, we provide a module that determines the threshold according to the data behavior.

4 ANOMALY DETECTION TOOL (ADT)

This section presents the anomaly detection tool (ADT) that is designed for detecting the anomalies in the cryptocurrency exchange data. The framework is developed for detecting outliers in univariate time series, but it can be easily adapted to detect anomalies in multivariate data. Figure 1 gives a high-level description of our anomaly detection tool. First, the data is collected from the different cryptocurrency exchanges using the APIs and a set of statistical quantities are computed and stored in the form of univariate time series. The tool then process these time series in the form of disjoint windows (batches).

After receiving the first batch from the stream, an outlier detection model that combines a statistical method that is based on estimating the probability density function (PDF) [24] of the data and an isolation-based approach using Isolation Forests [15] is created and used to compute the outlierness score of each sample in the batch. The created model is used to detect the outliers (anomalies) until a concept drift is detected where a new model is created using the current batch. In our framework, concept drifts are detected by monitoring the anomaly rate reported by the anomaly detection tool. The anomaly rate should not be too high, since outliers (by definition) should be rare events. Once the anomaly rate becomes significantly higher compared to the anomaly rate of the original sample, it is assumed that a change has occurred in the distribution of the data. The current model is then deemed inaccurate, so a new model has to be built using the data in the current window.


Cryptocurrency Exchanges
Pull changes
in the order book
Extract the features of interest
x_1 x_2 x_n ... x_{i+1} x_{i+2} ... x_{i+n} x_{i+n+1} ... x_{i+2n} x_{i+2n+1} ...
Window (w_1)
Window (w_i)
Window (w_{i+1})
Build Isolation Forest model and Re-estimate the PDF
Estimate outlier scores using Isolation Forests
Estimate outlier scores using the estimated PDF
Concept Drift Detected
Highlight outliers
Combine outlier scores

objType	status	uid	id	exchangeUid	currencyPair	type	price	appTimestamp	amount	timestamp	modTimestamp
Order	Created	o-hitbtc-btc-usdc-sell-7559-44879	btc-usdc-sell-7559-44879	hitbtc	BTC-USDC	Sell	7559.45	1575802978	0.37486	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7564-63491	btc-usdc-sell-7564-63491	hitbtc	BTC-USDC	Sell	7564.63	1575802978	2.0934	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7567-15154	btc-usdc-sell-7567-15154	hitbtc	BTC-USDC	Sell	7567.15	1575802978	0.97838	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7570-61495	btc-usdc-sell-7570-61495	hitbtc	BTC-USDC	Sell	7570.61	1575802978	0.59756	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7571-75960	btc-usdc-sell-7571-75960	hitbtc	BTC-USDC	Sell	7571.76	1575802978	0.03	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7574-80420	btc-usdc-sell-7574-80420	hitbtc	BTC-USDC	Sell	7574.8	1575802978	0.04006	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7574-80421	btc-usdc-sell-7574-80421	hitbtc	BTC-USDC	Sell	7574.8	1575802978	0.94853	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7575-03317	btc-usdc-sell-7575-03317	hitbtc	BTC-USDC	Sell	7575.03	1575802978	3.0002	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7576-92728	btc-usdc-sell-7576-92728	hitbtc	BTC-USDC	Sell	7576.93	1575802978	2.37594	1575802976	1575802978
Order	Created	o-hitbtc-btc-usdc-sell-7578-31602	btc-usdc-sell-7578-31602	hitbtc	BTC-USDC	Sell	7578.32	1575802978	7.3196	1575802976	1575802978

Figure 1: Anomaly Detection Framework

Using PDF for Outlier Detection (PDF-Based OD): in our ADT tool, we implemented a modified version of the outlier detector proposed in [24]. Using probability density functions (PDFs) in outlier detection has been shown to outperform other popular outlier detection methods in terms of time efficiency and detection effectiveness [14, 24]. The KDE-Track [24] has been proposed to reduce the quadratic time complexity of the Kernel Density Estimation (KDE). KDE-Track [24] uses the traditional KDE to estimate the PDF of a set of samples $M = m_1, m_2, ..., m_k$, where $m_i - m_{i-1} = c, \forall 1 \leq i \leq k$. These points are called resampling points. KDE-Track utilizes *linear interpolation* to estimate the PDF for values that are not among the set of resampling points. Using linear interpolation to approximate the density value $\hat{y} = \hat{f}(x)$ of a given value x that lies within an interval I_i, with boundaries m_i and m_{i+1} and densities $\hat{y}_i$ and $\hat{y}_{i+1}$ is estimated as follows:

$$\tilde{f}(x) = \hat{y}_i(1 - \frac{x - m_i}{m_{i+1} - m_i}) + \hat{y}_{i+1}(\frac{x - m_i}{m_{i+1} - m_i}). \qquad (1)$$

The upper bound for the extra estimation error incurred by linear interpolation has been computed in [24] to be: $|\tilde{f}(x) - \hat{f}(x)| \leq \frac{\{m_{i+1}-m_i\}^2}{8}\widehat{f''}(x)+O_p(\{m_{i+1}-m_i\}^3)$, where $\hat{f}(x)$ is the density estimated by KDE and d is the distance between two resampling points. We use the estimated error to differentiate between weak/strong outliers as will be explained later.

Isolation Forest: Isolation Forests (IFO) are based on the intuition that outliers (anomalies) exist in sparse areas so they can be isolated from the rest of the values using a small number of splits on the data space. IFO-based anomaly detection models generate a set of t binary trees (called iTrees) using a subsample (with size ψ) of the data set. The value of ψ should be smaller than the data set size to avoid model over-fitting, where $\psi = 256$ has been shown empirically to be good enough. In an iTree, instances are randomly recursively partitioned. When the tree is fully grown, each leaf of the tree represents an instance in the subsample. It is expected that outliers are, on average, easier to isolate than normal data points (i.e. the path length in the isolation tree should be relatively small). Using large number of trees makes the model more robust but increases the computational cost. Using 100 trees has been shown to perform well in many applications [15]. In order to calculate the outlierness score, we estimate the average path length to the leaves, given a sample of ψ instances, to be the number of unsuccessful searches in an equivalent binary search tree. It is estimated as:

$$c(\psi) = \begin{cases} 2H(\psi - 1) - 2(\psi - 1)/n & \text{for } \psi > 2 \\ 1 & \text{for } \psi = 2 \\ 0 & \text{otherwise} \end{cases},$$

where $H(i)$ is the harmonic number [22]. This expected average path length c is used to normalize the average path length of specific instances in the Isolation Forest. The anomaly score $s(x, \psi)$ of an instance x is defined as follows: $s(x, \psi) = 2^{-E(h(x))/c(\psi)}$, where $s(x, \psi) \rightarrow 1.0 \implies E(h(x)) \rightarrow 0$ and the sample is outlier.

Batch Window: to take the context of the data values in consideration while detecting the anomalies, the model should adapt to the changes in the data distribution. Older data might not be relevant anymore [23], so ADT utilizes a batch window to keep the most recent data values. The observations in the batch window are used to detect the contextual anomalies and discover possible concept drifts in the data. Upon detecting a concept drift, the ADT model is rebuilt using the most recent batch of values. The size of the batch window is application dependent where in ADT, we use a default setting for the batch window to hold the data collected in one day.

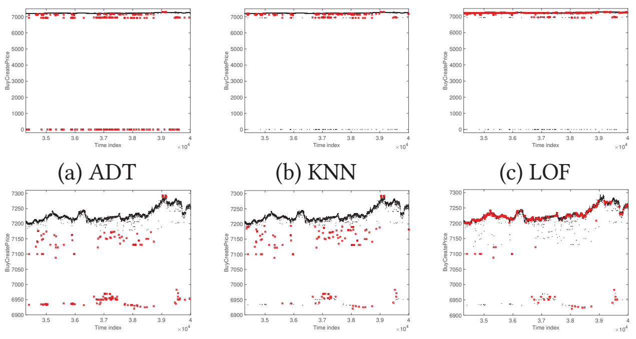
Outlier Ensembles: ADT combines the results of the Isolation Forest and PDF-based outlier detection to determine the outliers. First, we estimate the threshold value for each component. For PFD-based anomaly detection, the average density $\bar{\tilde{f}}$ at the resampling points (the points that discretized the support of the PDF) is computed and the threshold τ_{PDF} is estimated to be $\tau_{PDF} = 0.05 \times \bar{\tilde{f}}$. However, to take the PDF estimation error in consideration, we use a small interval around the determined threshold of length $2 \times \epsilon$, where $\epsilon = \frac{\{m_{i+1}-m_i\}^2}{8}\widehat{f''}$ to indicate a weak outlierness signal. For isolation forest, we first normalize the outlierness score using $z - score$ normalization such that the score values have mean $\mu = 0$ and standard deviation $\sigma = 1$ and consider $\tau_{IFO} = 3 \times \sigma$. We consider an interval around τ_{IFO} of length $2 \times \delta$ with $\delta = 0.5 \times \sigma$. We apply the same technique of strong/weak (normal/outlier) based on the normalized outlierness score. To combine the score of the two methods, a data value is considered to be an outlier if it has **at least** one strong outlier signal and the other signal is a strong or weak outlier signal.

5 EVALUATION

To evaluate our approach, we compare its performance with the performance of Local Outlier Factor [5] and K-Nearest Neighbors [4], which are widely used for outlier detection. We use the implementation of KNN and LOF from the scikit-learn library. ADT is also implemented in Python. The evaluation is done using the extracted features from the cryptocurrency exchanges data and anomaly detection benchmark data sets provided by *Yahoo!* [27]. The *Yahoo!* data set has labeled anomalies which allows us to compute F_β-score with $\beta = 2$ as a performance measure.

Table 1: Performance of anomaly detection techniques

	ADT	iFO	KDE	LOF	KNN
id	F2	F2	F2	F2	F2
2	**0.897**	0.844	0.844	**0.897**	0.733
6	0.875	**0.930**	0.854	**0.930**	0.875
11	**0.914**	0.869	0.869	**0.914**	0.730
13	0.847	0.714	**0.932**	0.833	0.789
21	0.968	0.938	0.938	0.968	**1.0**
39	**0.776**	0.625	0.714	0.689	0.714
42	**0.987**	0.707	0.616	0.352	**0.987**
53	**0.977**	0.952	0.944	0.808	**0.977**
60	**0.904**	0.897	0.879	0.337	0.733



(a) ADT (b) KNN (c) LOF

(d) ADT (e) KNN (f) LOF

Figure 2: Comparing ADT with KNN and LOF when detecting anomalies of the feature *maxBuyPrice*.

Cryptocurrency Data: the cryptocurrency data is not labeled so the anomalies are unknown. Because of that, comparing the evaluated methods using the F-Score is not feasible. To evaluate the different methods, we generated figures that highlight the anomalies detected by the different methods and asked the domain experts to check the detected anomalies. Figure 2 shows the detected anomalies by ADT, KNN and LOF on the *maxBuyCreatePrice*. LOF has shown to perform poorly even though we used multiple parameter settings (around 80 combinations) and selected the most reasonable results. Subfigures (a,b and c) show the whole data set for one day, while subfigures (d, e and f) focus on the data values within the interval [6900, 7310]. As it can be observed from Subfigures (a, d), ADT was able to detect most of the outliers especially those with *price = 0*, which are undetectable by KNN and LOF. Moreover, the computational cost and the user involvement required by KNN and LOF discouraged the domain experts from recommending these methods. Unfortunately, we cannot include the Figures for IFO and PDF in this submission due to space limitations[1]. Suspicious data patterns for this feature can be observed in subfigures (d, e and f) where a set of orders over the 3 months that has a price less than the market price by a constant margin around *price = 6950* (the figure shows one day data only). Domain experts confirmed that the orders are either placed by trading bots or they are fake orders trying to inflate the trading price.

***Yahoo!* benchmarks:** we evaluated ADT against the baseline methods on benchmarks with labeled anomalies. The results are presented in Table 1. ADT has the best F2-Score in more than 66% of the data sets[2].

[1]The code and the data sets cannot be shared as they are protected by an NDA.
[2]The column "id" in the table is the dataset id provided by *Yahoo!*.

6 CONCLUSION

In this paper, we described the process of collecting and analyzing the order book data from different cryptocurrency exchanges. We developed a tool for detecting anomalies in the data that can help in spotting possible market manipulation. Our tool combines two well-known techniques to provide a more robust and accurate solution. The experimental results on well-known benchmark data and the cryptocurrency data show that our tool is able to achieve the best overall performance. For future work, the anomaly detection tool can be further extended to detect anomalies in multi-dimensional data streams. Furthermore, overlapping sliding windows can be used to provide online and real time analysis of the data.

REFERENCES

[1] R.K. Aggarwal and G. Wu. 2006. Stock Market Manipulations. *Journal of Business* 79, 4 (2006), 1915–1954.

[2] M. Ahmed, A.N. Mahmood, and J. Hu. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* 60 (2016), 19–31.

[3] F. Allen and D. Gale. 1992. Stock-Price Manipulation. *The Review of Financial Studies* 5, 3 (1992), 503–529.

[4] F. Angiulli and C. Pizzuti. 2002. Fast Outlier Detection in High Dimensional Spaces. In *ECML/PKDD*.

[5] M.M. Breunig, H. Kriegel, R.T. Ng, and J. Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *SIGMOD*.

[6] M. Brunnermeier and L. Pedersen. 2009. Market Liquidity and Funding Liquidity. *The Review of Financial Studies* 22, 6 (2009), 2201–2238.

[7] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T.M. McGinnity. 2014. Detecting price manipulation in the financial market. In *IEEE Conference on Computational Intelligence for Financial Engineering and Economics*.

[8] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T.M. McGinnity. 2015. Detecting Wash Trade in Financial Market Using Digraphs and Dynamic Programming. *IEEE Transactions on Neural Networks and Learning Systems* 27, 11 (2015), 2351–2363.

[9] D. H. Freedman, R. Pisani, and R. Purves. 1978. *Statistics.* W. W. Norton & Company.

[10] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. 2014. A Survey on Concept Drift Adaptation. *Comput. Surveys* 46, 4 (2014).

[11] J. Huang, Q. Zhu, L. Yang, and J. Feng. 2016. A non-parameter outlier detection algorithm based on Natural Neighbor. *Knowledge-Based Systems* 92 (2016), 71–77.

[12] J. Kamps and B. Kleinberg. 2018. To the moon: defining and detecting cryptocurrency pump-and-dumps. *Crime Science* 7 (2018).

[13] E.M. Knorr and R.T. Ng. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*.

[14] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. 2007. Outlier Detection with Kernel Density Functions *(MLDM)*.

[15] F.T. Liu and K.M. Ting. 2012. Isolation-based Anomaly Detection. *TKDD* (2012).

[16] I. Makarov and A. Schoar. 2020. Trading and arbitrage in cryptocurrency markets. *Journal of Financial Economics* 135, 2 (2020).

[17] P. Monamo, V. Marivate, and B. Twala. 2016. A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers. In *ICMLA*.

[18] P. Monamo, V. Marivate, and B. Twala. 2016. Unsupervised learning for robust Bitcoin fraud detection. *Information Security for South Africa* (2016).

[19] A.I. Nilsen. 2019. Limelight: Real-Time Detection of Pump-and-Dump Events on Cryptocurrency Exchanges using Deep Learning. *University of Norway* Master's Thesis (2019).

[20] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, and C. Faoutsos. 2003. LOCI: Fast Outlier Detection using the Local Correlation Integral. In *ICDE*.

[21] T.T. Pham and S. Lee. 2016. Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods. *Stanford University* Research (2016).

[22] B. R. Preiss. 1999. *Data Structures and Algorithms with Object-Oriented Design Patterns in Java* (1999).

[23] A. Qahtan, B. Alharbi, S. Wang, and X. Zhang. 2015. A PCA-Based Change Detection Framework for Multidimensional Data Streams. In *KDD*.

[24] A. Qahtan, X. Zhang, and S. Wang. 2012. Efficient Estimation of Dynamic Density Functions with an Application to Outlier Detection. *CIKM* (2012).

[25] T. Sethi and M. Kantardzic. 2017. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications* 82 (2017), 77–99.

[26] P. Weber and B. Rosenow. 2005. Order book approach to price impact. *Quantitative Finance* 5, 4 (2005), 357–364.

[27] Yahoo! Webscope, [Dataset]. [n.d.]. S5 - A Labeled Anomaly Detection Dataset (v1.0). http://webscope.sandbox.yahoo.com