PUMP AND DUMP DETECTION ON CRYPTO CURRENCIES USING COMPUTER VISION

Jayasundara Abeykoon Waruna Priyankara Wickramasingha

199371P

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

PUMP AND DUMP DETECTION ON CRYPTO CURRENCIES USING COMPUTER VISION

Jayasundara Abeykoon Waruna Priyankara Wickramasingha

199371P

Dissertation submitted in partial fulfilment of the requirements for the Degree Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

2021/05/29

J.A.W.P Wickramasingha Date

The above candidate has carried out research for the Master's dissertation under my supervision.

…….. 2021/05/29

Dr. Uthayasanker Thayasivam Date

Abstract

Inspired by the immense success shown by artificial neural networks in computer vision on images classification, we propose a novel framework to detect one of the rife fraudulent financial manipulations in crypto currency trading world known as pump and dump. The representation of crypto currency financial charts was re-imagined ameliorating the classification by taking advantage of some of the very recent advancements of time series to spatial encoding techniques of Gramian Angular Field (GAF), Markov Transition Field (MTF) and Recurrence plots (RP) that are capable of spatially encoding the temporal financial time series data in the form of images. Encoded images were then used to train several convolutional neural network architectures which have been able to achieve a very high precision, recall and F1 values close to 99% over the unseen data for the above classification task. This is one of the first of such researches in pump and dump detection in crypto currencies using computer vision. This approach has the potential to be extended in detecting predefined shapes of time series charts.

Key words

cryptocurrency, pump and dump, imbalanced time series classification, spatial encoding temporal data, Gramian angular field, Markov transition field, recurrence plot, cnn, machine learning, market surveillance, class imbalance problem, synthetic minority class oversampling technique

Acknowledgement

My utmost gratitude and appreciation goes to Mrs. Kaushalya Kularatnam - Head of Quantitative Surveillance & Technology London Stock Exchange in making this project a success by providing the initial idea of this research and facilitating all the business, technical and required cloud resources for this research from the very beginning.

I would like to give my utmost gratitude to my supervisor Dr. Uthayasanker Thayasivam - Senior Lecturer of department of Computer Science and Engineering of University of Moratuwa, giving the outstanding consultation and supervision throughout the project.

I am really grateful to Dr. Rasika Withanawasam - Senior Software Architect at Millennium Surveillance systems, sharing all the expertise and knowledge in financial surveillance world and providing the golden opportunity to conduct this research with the help of London Stock Exchange's (LSE) market supervision team.

I would like to extend my gratitude to all the wonderful people from London Stock Exchange Quantitative Surveillance team and LSEG Technology who supported verifying the classifications by these models and assisted me expand the knowledge and exposure to capital market and market surveillance domains.

Last but not least, I would like to extend my gratitude to my beloved wife Ranmali and beloved baby Kiara who were behind me all the time giving me all the strength, courage and dedication for the time required for this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
CNN	Convolutional Neural Network
GAF	Gramian Angular Field
GASF	Gramian Angular Summation Field
GADF	Gramian Angular Difference Field
MTF	Markov Transition Field
RP	Recurrence Plot
PND	Pump and Dump
OHLCV	Open High Low Close Volume
MTS	Multivariate Time Series Sensory
SMOTE	Synthetic Minority Oversampling Technique

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Pump and dump (PND) is one of the most common fraudulent price manipulations being reported in financial markets since the early ages of 1700s. With the technological advancements of social media and cryptography, and proliferation of the acceptance of crypto currency as a medium of payment in many companies, crypto currency has been an emerging financial market where a dearth of regulations are imposed when compared to traditional capital markets. Therefore, it has opened many grounds to market manipulators to organize and conduct fraudulent activities like pump and dump over crypto currency markets. Owing to the fact that the shape of price and volume graphs in case of a PND is known to follow a specific pattern, investigation of a PND activity can be easily done by a market surveillance expert simply by eyeballing the price and volume charts of the stocks, followed by analyzing the trade participant details for the point of suspicion using the exchange level data to confirm the fraud. The sole intention of this research was to investigate the possibility to replace the human eyeballing process to a computer vision technique in identifying possible PND cases from crypto currency time series data.

1.1 Problem Definition

Pump and dump (PND) is a coordinated and short-term market manipulation to artificially inflate the price of an owned security and then selling it to a much higher price to other investors. This is a quite common fraud in crypto exchanges, where the regulations are loosely imposed due to its nature of cutting regulators in the transactions. There are three stages involved in any pump and dump activity.

1. Accumulation stage – the offenders accumulates an adequate amount of commodity with buy orders over a period.
2. Pump stage – artificially inflate the price of the crypto currency coin by the means of misinformation, promotions and email spams and artificially impose a synthetic demand for the coin with buy orders with even higher prices than the true value of the coin which in turn causes a net positive

effect on the stock price to increase with a synthetic demand influencing the other traders to buy the coin at higher prices.

3. Dump stage – when the price has gone high enough, manipulators start selling the stocks which they have bought previously at a lower price in the accumulation stage. Owing to the inflated price when the pump is complete, the price starts to drop significantly at the dumping stage until it reaches the coin's real value causing the misinformed buyers the victims who has taken part in buying at inflated prices and end up in loss.

Due to the unregulated nature of the crypto exchanges via central stock depositories (CSD) or banks and whole process is powered by the block chain technology, the manipulations like pump and dump are quite possible in crypto exchanges. Therefore, it creates negative impacts on the genuine investors who become the prey of the manipulators and trigger multiple financial repercussions. Therefore, pump and dump are detrimental to the liquidity and price of the coins.

Figure 1.1 below shows the price variation of a coin in the above three stages during a pump and dump. [1] Type of chart in Figure 1.1 is called OHLC (open, high, low, and close) price candlestick chart where each candle represents values of Open, High, Low, Close prices within a certain time duration (ex: 1 minute). Colors in Figure 1.1 are given such that green color when closing price is higher than open price and red color otherwise. High and Low values refer the highest and lowest values observed within the candle window.

Figure 1.1 Schematic abstraction of the three phases of a pump-and-dump operation

Sources: [1]

Figure 1.2 below shows how to convert a continuous time series chart to discrete OHLC candlestick [2] by encoding the Open, High, Low, Close values for the duration.

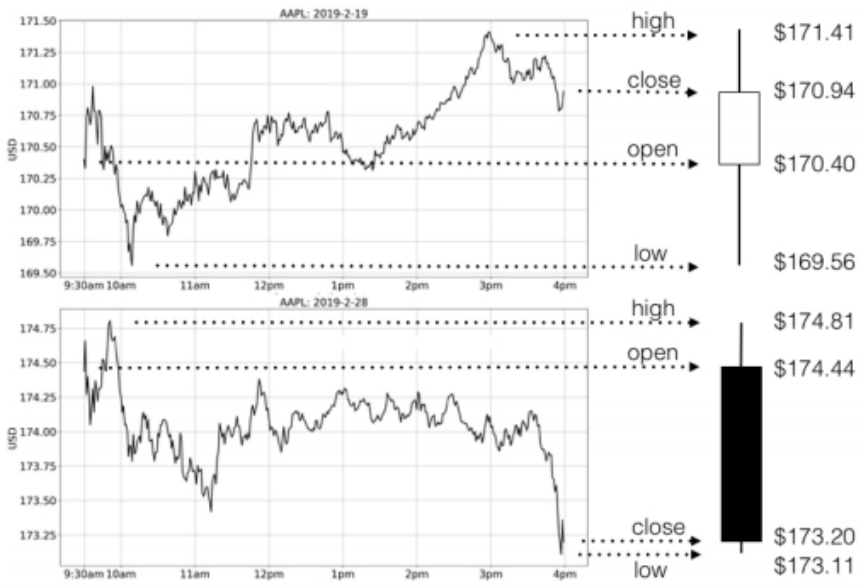


Figure 1.2 Converting continuous time series to OHLC candlestick

Sources: [2]

Pump and dump schemes are often active on crypto currencies called “penny” or “microcap” that represent a smaller market capital and do not meet the requirements to be listed on reputed large exchanges [1]. These microcap exchanges are not holding the level of standards and regulations and the listed commodities also do not hold the reputation with a dearth public information available and hence are easier to create misinformation by the manipulators. Often, these fraudulent tasks are organized via public online chat groups such as Discord and Telegram with a number of members as high as 200,000. Those groups usually target less popular crypto currencies with low market cap and low circulation [1]. In those chat groups, group leaders schedule pump and dump events on selected coins and exchanges resulting genuine investors who do not have insider information of the chat group to start buying those coins from those exchange amidst of fake news, false stories, fake partnership or fake celebrity endorsement news etc considering the synthetic price hikes as genuine trends and finally end up in loss.

1.2 What are Crypto Currencies?

Apart from conventional capital market trading, crypto currencies have increasingly gained the attention of the public as an investment platform and an internet based medium of exchange to conduct financial transactions by replacing the use of fiat currencies such as USD, GBP etc. The main feature of crypto currency is its backbone technology known as Blockchain, making it a decentralized platform not being controlled by any central authority such as governments or banks. Crypto currencies are transferred directly between the two counterparties with the help of cryptography by using private and public keys and spend a minimal processing fee by skipping the intermediary.

In traditional financial systems customers trust a third party such as a bank to update their account balance with the ledger on behalf of them. In case of crypto currencies this task is distributed across a network in which everyone gets a copy of the ledger and verify the content to succeed the transaction [1].

Table 1.1 Top 10 coins by percentage of market cap

Coin	% of total market cap
BTC	42.0
ETH	19.4
XRP	7.8
BCH	5.6
EOS	3.5
LTC	2.1
XLM	1.6
ADA	1.6
TRX	1.2
USDT	0.95

Sources: [1]

Crypto currencies are a byproduct of the invention of Bitcoin (BTC) by Satoshi Nakamoto, opening a world of decentralized peer-to-peer electronic cash system [3] which currently possesses a market capitalization around 300 billion USD. Due to its unregulated nature by principles, crypto currencies are an attractive target of many

scams such as pump and dump. Table 1.1 above shows the top 10 crypto currencies with their market capital [1].

1.3 Motivation

Majority of the time series classification literature has focused on processing individual 1D features of the multivariate or univariate time series data and then use mathematical models such as adaptive thresholds [1], Gaussian models or use LSTM models [4] for classification. Table 1.2 below summarizes some of the commonly used techniques in pump and dump detection by previous researchers.

On the contrary, in this research time series data were represented as images such that it introduced a different feature set that was not available in original 1D time series data and enabled models to search only a smaller parameter space to engineer the features [5]. Therefore, with the recent proven success of computer vision techniques involving artificial neural networks, it captured our attention to conduct a research on the ability of artificial convolutional neural networks (CNN) to classify time series data by mining the pump and dump shapes from crypto financial time series data.

Table 1.2 Common approached in PND detection

#	Technique	Shortcomings
1	Threshold based alerting [1]	Threshold must be predefined.
2	NLP text mining applications monitoring public chat groups. [6]	Need access to all the chat groups. Secret chat groups cannot be used. Cannot read cyphered images when used to announce PND.
3	Clustering and outlier detection [7]	Unsupervised, automatic thresholds, still not given best results.
4	Mathematical models using exchange level data. [8]	Need access to exchange level data which is strictly governed.

Historic trading details for a crypto currency can be taken by few of the crypto currency exchanges in the form of a time series containing data such as open price,

high price, low price, close price and trading volume (known as OHLCV data) for a specific time period (candle width) with the corresponding timestamp. Table 1.3 below shows one of such OHLCV records for a specific crypto currency for a specific time duration [1].

Table 1.3 Example data point of OHLCV data stream

Timestamp	Price				Trading volume
	Open	High	Low	Close	
2018-04-20 01:00:00	0.11804	0.11882	0.11758	0.11881	181.16102255

Sources: [1]

In order to convert financial time series data into an image, few options were identified. One was to directly chart data points against time and take the image of that graph as shown in Figure 1.3, otherwise to convert/encode time series data into a spatial/texture format (i.e an image) with the help of some encoding techniques.

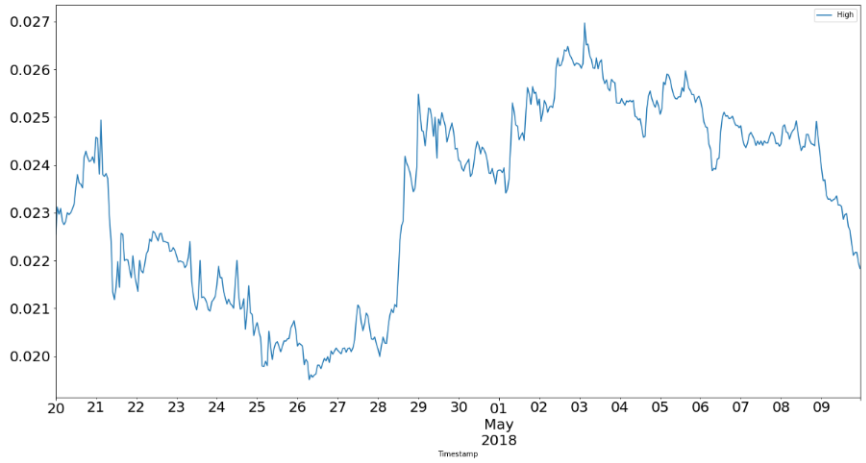


Figure 1.3 Price chart of a crypto currency against time

Sources: [1]

Therefore, below temporal to spatial encoding techniques were used in this research for this spatial conversion task.

1- Gramian Angular field (GAF) [5]
2- Markov transition field (MTF) [5]

3- Recurrence plots (RP) [5]

The perks of using above techniques in timeseries classification were their ability to achieve significant results beating the state of the art techniques with a high precision, recall, and f1 value around 99% on the unseen data while employing a simple CNN architecture consuming spatially encoded Price and Volume features out from 1 minute Open, High, Low, Close, Volume candles (known as OHLCV).

This was one of the first of such researches conducted to make use of above spatial encoding techniques to address the problem of PND detection. Hence time series classification problem was converted into a texture image classification task where even simple CNN architectures could achieve significant results. This framework of spatial encoding and CNN modelling was computationally less complex as well. It had also alleviated the need of complex feature engineering and yet all the temporal interdependencies were present in the encoded 2D image and CNN models would self-discover those feature types.

1.4 Pump and Dump alerting logic

Below set of observations are present in every pump and dump (PND) scenario therefore those were considered as major factors to sense a possible PND activity.

1. Trade price is advanced by a participant using a defined number of small buy trades or more.
2. Price of these buy trades are at least a percentage above the last trade price of the instrument.
3. The cumulative sum of small buy trade quantities is at least a user defined percentage of the total traded volume during the monitoring period.
4. A large sell trade is executed for the same instrument by the same participant with a price at or above the trade price of the last buy trade executed by the participant for the instrument.
5. The large sell trade quantity is at least a user defined percentage from the cumulative sum of small buy trade quantities.

6. The sell turnover from the large sell trade is greater than the cumulative buy turnover of small buy trades.
7. If above pattern is observed in the given order within a user defined period of time, then it can be taken as an alert for a possible complete pump and dump.

However, the market data published by crypto exchanges do not include the trade participant information and therefore it is not possible to identify whether the buy and sell trades were submitted by the same trading member. Because of this, a simplified version of above alerting logic is suggested over a rolling window of 10 minutes as below.

1. Price was increased at least by 15% during a 10 minutes rolling window.
2. Turnover at the end of the rolling window was significant at least by 75% for the window.
3. 10min window to the left had a pumping shape.
4. 10min window to the right had a dumping shape.

CHAPTER 2

LITERATURE REVIEW

2 LITERATURE REVIEW

Pump and dump detection is a time series classification problem and there was a dearth of reliable published literature on the pump and dump detection problem where the code and data were being public. Rather, most of the existing literature found on the keyword ‘pump and dump’ were focused on contexts such early pump and dump prediction and yielding financial benefits [9] [10], and literature on how pump and dump activities are carried out, and effects of pump and dump activities over the financial markets [11]. Pump and dump detection problem was only taken up by few of the researches such as [1] [12] [13] [8] and most of their data and models were not publicly available and some have used exchange level data for their research [8]. And there have been only a few of such attempts to spatially encode time series data as images to do the classification in some of the time series anomaly detection contexts such as, sensory data, financial data etc. List below are some of the common approaches other researchers have taken up this context of pump and dump detection.

2.1 NLP techniques to classify financial frauds

It was observed the symbiotic nature of most of the financial manipulations to omnipresent with a widespread of misinformation such as spam emails, fake press releases on acquisitions, merges and celebrity partnerships, social media campaigns, announcements by manipulator groups in social media platforms like Telegram and Discord. Therefore, there have been previous work to correlate and classify these scam activities like pump and dumps by implementing text mining algorithms to predict and post detect anomalies. Distribution of false information such as above, play a key role in urging the investors to involve in the scam and it is quite easy to make a significant influence on those illiquid small market capital coins. With the proliferation of internet-based communication platforms, social media and decentralized nature of crypto market have provided fertile grounds for scammers to organize and march the manipulations.

Mohamed Zaki at el [10] in 2012 suggested a market monitoring framework (MMF) with an automated linguistic based text mining technique that used a dictionary

based and pattern-based information extraction process to mine spam emails and misleading press releases on trading data such that they could proactively predict an ongoing market manipulation activity like a PND. Domain specific taxonomies from financial ontologies, thesaurus, synonyms and semantic grouping of concepts were used in their dictionary-based technique. Morphological analysis was used to extract tokens from spam emails and named entity recognition following linguistic rules had been used to extract information like stock symbols. However, these techniques became challenging when the manipulators use deceiving ways to communicate the information using special formatting in the messages.

ex: specify symbol ADABTC as ADABT$C or A|D|A|B|T|C etc.

Mehrnoosh Mirtaheri at el [6] in 2019 has experimented on how to relate data on twitter feeds with cash-tags to predict an unfolded pump and dump activity to be announced in Telegram channels and to predict whether such attempts announced in Telegram would succeed by reaching its Price targets. They implemented crawlers on Twitter and Telegram to scrape the textual data and classify each message as to pump or not-pump by representing each post as a TF-IDF vector and used n-grams to create a vocabulary. A linear SVM with SGD optimizer then achieved 87% accuracy and 89% precision on predicting a possible pump and dump announcement in Telegram by using Twitter data. A binary random forest classifier was implemented using a set of features from Twitter feeds and market data were used to extract “buy” and “target” prices from all the textual messages and coin Price data were used to evaluate whether announced pump and dump activity had been successful.

Figure 2.1 Use of obfuscation to make it hard for bots to parse

Sources: [14]

However, above NLP techniques were based on textual data, those had faced challenges when the messages used cyphered images as shown above in Figure 2.1 for coin announcements and communication. In some cases, obfuscation was used in coin announcements messages in Telegram and Discord channels mainly to avoid automated bots to take part in action and do the trades because only a human need to read such messages and take further actions. Therefore, NLP techniques faced challenges when the required data were cyphered and when the communication happens over secret groups which the tools cannot access.

2.2 Converting time series classification into image classification domain

Zhiguang Wang et al [5] in 2015 suggested a novel approach to encode time series data as images for classification by enabling computer vision techniques to classify time series data via the new framework called Gramian angular fields and Markov transition fields. They had shown its success by using a tiled convolutional neural network architecture on 12 standard datasets so that it could extract high level features by searching a smaller parameter space, while yielding competitive results to other state of the art methods in the classification. Their work had opened an novel approach to realize time series classification problems as image classification problems and work on this research was also based the encoding techniques suggested by [5].

Ruinan Zhang et al [15] in 2018 implemented an fraud detection schema for online streaming data such as online shopping and financial scenarios and proposed two neural models such that, one model was built by encoding the sequential data and trained using a RNN constructed with LSTM cells while the other model was built using a CNN trained upon the Markov transition fields extracted from the sequential data and shown that MTF encoding could boost the model performance.

Nima Hatam et al [16] in 2018 used recurrence plots (RP) to transform time-series data into 2D texture images and shown that by using multiple time series data sets from UCR achieve that, by spatially encoding the data as 2D images, a new types of features were introduced which were discovered by the neural models and took the advantage of a deep CNN to classify those images. They claimed to have achieved

competitive results against the SOTA techniques over those data sets in the experiments.

Chao-Lung Yang et al [17] in 2019 proposed encoding multivariate time series sensory (MTS) data into 2D colored images by using Gramian angular field and Markov transition field techniques to train a ConvNet and VGGNet models such that it could perform binary classification on the data as to normal or abnormal sensory data. They had encoded each 1D time series using GASD, GADF and MTF techniques to result in 3 encodings channels like RGB channels for an image and concatenated those layers each in RGB channel spaces to form a bigger image which was then used for model training and classification. They had claimed even simple ConvNet models had given competitive results when compared with VGGNet results and shown the advantages of the classification when simple CNN architectures were used with above spatial encoding techniques.

Naftali Cohen et al [2] from JP Morgan in 2019 showed that systematic financial trading can be done with more precision and maximized gain simply by converting and classifying time series financial data as images to train several machine learning models to detect patterns as trends, cycles and correlations that were used as the indicators for a prospective trade by an experienced trader. In their research also time series data had been encoded in GAF, MTF, and Recurrence Plots and achieved significant results indicating even very complex patterns to be discovered by multi scale algebraic operations could had been identified simply by transforming the task into an image classification problem.

Jun-Hao et al [18] in 2020 showed how they detected 8 types of popular OHLCV candlestick trading patterns such as morning star, evening start, bearish engulfing etc, over a GAF-CNN model with 90.7% average accuracy outperforming the LSTM models. The detected pattern can be used to forecast many indices in financial markets so that traders can decide to take part in the trading. Their experiments showed the impact of using max pooling layers in the CNN models affecting the results when encoded with GAF technique. Their work provided a sound baseline to the CNN architectures used in this research.

2.3 LSTM techniques for time series classification

Pankaj Malhotra at el [4] in 2015 showed how a stacked long short term memory network (LSTM) could be used for anomaly detection in time series by alleviating the need to pre specify a time window size by making use of LSTM's long term memory correlations in a sequence such that when the models were trained with normal behavior, it could identify an abnormal behavior. They have used a predictor at each time step and used the probability distribution of the related error to predict whether the observed values to be taken as a normal or abnormal behavior. They have experimented with four data sets and have achieved promising results where both short-term and long-term temporal dependencies had been discovered by the LSTM models. However, LSTM models trade off with the computational complexity and cost to train whereas this research, only simple CNN architectures were employed for this specialized problem of pump and dump detection.

2.4 Clustering and automatic thresholding techniques

Stan Salvador at el [19] in 2004 demonstrated a way to perform time series anomaly detection via generated states and rules using a clustering algorithm called Gecko and a method called L method which is similar to hierarchical and agglomerative clustering algorithms for dynamically finding a reasonable number of clusters by integrating RIPPER and finite state transition logic that can be easily read and modified by a human to generate a anomaly detection system. Hadi Mansourifar at el [12] in 2020 showed how their novel approach of a hybrid of distance-based and density-based metrics could perform pump and dump detection on crypt currencies by converting context anomaly problem into point anomaly detection problem. The distance-based method used an automatic thresholding by histogram processing and density-based anomaly detection used a novel approach called density score and they claimed to have achieved good results when above two methods were used in hybrid. For evaluating the results, they have used the labels given by the benchmarking model [1] used in this research.

2.5 Threshold-based approaches

Josh Kamps at el [1] in 2018 showed how to detect crypto currency pump and dump activities by identifying anomalies in price and volume data acquired from 1h-OHLCV data which exceeded a rolling window average for a specific window size by predefined thresholds. Their work has taken the ground truth by monitoring two pump and dump chat groups called Moonlight (3000 members) and Crypto trading (56000 members) and implemented 3 types of threshold parameters as initial parameters, strict parameters and balanced parameters. Their work has been referred by many following researches in pump and dump detection even by taking their labels as the ground truth. It was the only research that I could find online such that the code and the dataset was made open to public. Due to this reason, the work in this research has baselined Josh Kamps' to compare the results. Victor Friedhelm et al [20] in 2019 had showed how to classify pump and dump events using an extreme gradient boosting (XGBoost) binary classifier trained on a set of engineered features from price and volume from low market capital coins paired with BTC at Binance. They had labelled the high-resolution data taken from Binance using Twitter account of the coins and messages from telegram chatgroups using APIs and applied a thresholded logic to filter some of the PND detections. But none of their code, data or the way to calculate features were not published.

CHAPTER 3

METHODOLOGY

3 METHODOLOGY

The work on this research was based on 1-minute OHLCV data downloaded from Binance exchange using its API over BTC paired coins for a duration of 2 years from 1st of Jan 2018 to 1st of Jan 2020. Due to the unavailability of a labelled data, an initial numerical labelling process was applied over the data and only after the expert reviews on the labels, a synthetic balanced data set was created for model training using SMOTE technique. Then each data point was spatially encoded using the techniques explained below and CNN models were trained and used for prediction on the unseen data.

3.1 Downloading and encoding data

Historical crypto trading data for a duration of 2 years from Jan 1st 2018 to Jan 1st 2020 were downloaded for BTC paired coins from the Binance exchange in 1 minute OHLCV format using Binance python API and stored in aws s3 bucket as csv files. Binance is the largest crypto currency exchange of the world in terms of the trading volume. According to a similar research by [1] B. K. Josh Kamps, "To the moon: defning and detecting cryptocurrency pump-and-dumps" *Crime Science,* 2018 BTC paired symbols were considered to have a higher chance of being manipulated and therefor it was considered as one of the reinforcing factors for a pump and dump.

Python's Pyts [21] module was used to generate all above spatial encodings out of a time series data. "pip install pyts" would install the library in a Linux environment. Appendix D has shown high level pseudo code used to generate N * N spatially encoded matrix out of N time points. Figure 3.1 shows a sample chart and Table 3.1 shows how it appeared when spatially encoded using above given techniques.

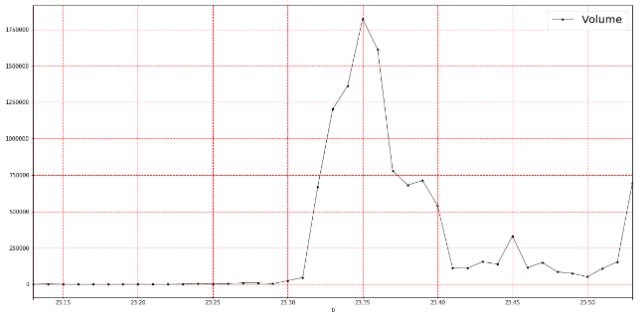
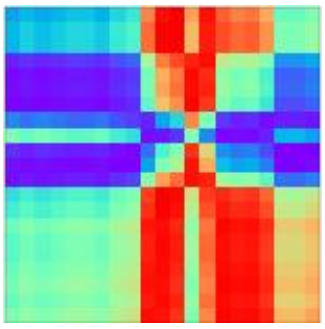
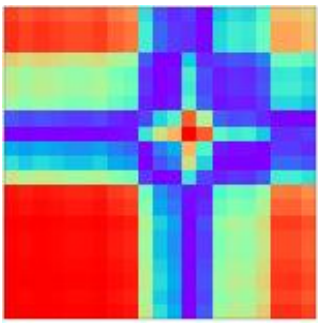
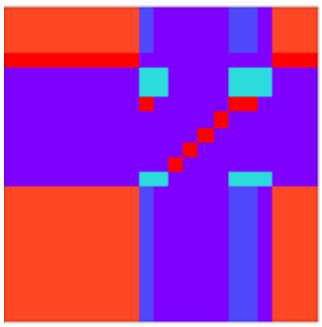
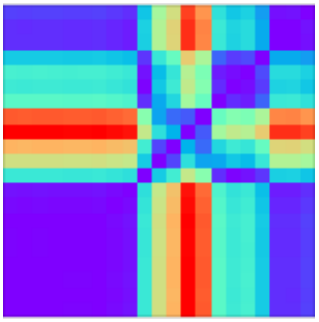


Figure 3.1 Sample chart to be encoded in different techniques

All the computational and storage resources are taken from aws cloud infrastructure. Coding was done at aws Sagemaker notebooks by choosing an appropriate instance type depending on the complexity of the load. For an example, when downloading data ml.m5.2xlarge instance type was used and when labelling data and hyperparameter tuning of the models, ml.m5.12xlarge instance type was used. Storage of all the originally downloaded data, enriched data, labelled data, model data are saved in aws s3 buckets so that unlimited, durable, persistence storage is available.

Table 3.1 Same time window when encoded in different spatial techniques

 GADF	 GASF
 MTF	 RP

3.2 Binance python API

In order to start using Binance python API, it is first required to create a free account in Binance and generate an API key and secret key. Then using ‘pip install python-binance’ it would install python-binance module on the Linux environment to be used for downloading the data. In the context of this research data with a higher resolution is required which means the width of the candle window in OHLCV data need to be as small as possible. There were different standard sizes binance API support for download data and some of them are as follows.

1. KLINE_INTERVAL_1MINUTE= '1m'
2. KLINE_INTERVAL_3MINUTE= '3m'
3. KLINE_INTERVAL_5MINUTE= '5m'
4. KLINE_INTERVAL_15MINUTE= '15m'
5. KLINE_INTERVAL_30MINUTE= '30m'
6. KLINE_INTERVAL_1HOUR= '1h'
7. KLINE_INTERVAL_2HOUR= '2h'
8. KLINE_INTERVAL_4HOUR= '4h'
9. KLINE_INTERVAL_6HOUR= '6h'
10. KLINE_INTERVAL_8HOUR= '8h'
11. KLINE_INTERVAL_12HOUR= '12h'
12. KLINE_INTERVAL_1MONTH= '1M'

Out of above candle sizes, the smallest window was 1-minute candle which was chosen as the window size to obtain data for this work. Binance Client module was used to connect to the data server and to download data into the given location where it was required to specify a ‘from’ and a ‘to date’ range to extract the data from. Following a successful connection, data could be downloaded from Binance exchange data servers. After checking the system status and exchange information such as rate limits and listed symbols, only BTC paired symbols were selected to download. More information on the Binance API is available at [22]

In order to conduct pre-experiments, initially data for 1months duration was downloaded into pandas’ data frames and stored in s3 as csv files. Different non-overlapping batches of data such as 1 month, 4 months, 6 months, 2 years were downloaded for experimenting. With the increase of the span of the data, it was much time consuming to complete the download. As an example, it had taken almost 3 days

to download 2 years' worth of data from Jan 1st, 2018 to Jan 1st, 2020. The name of the csv file that was given below format to manifest its dates and coin pair.

<symbol_pair_name>_from_<from date>_To_<to date>.csv

ex:- AMBBTC_from_2018-01-01_000000_To_2020-01-01_000000.csv

One challenge in downloading data for a wide span of time was the network jitters and intermittent disconnections from the Binance server and notebook client. Therefore, each time after encountering such hindrance, download was re-attempted from the beginning for a particular coin by checking the existence of the corresponding csv file at s3 destination location. Appendix A has shown the logic used to download data from Binance irrespective of the network disconnections.

3.3 Data cleaning and enriching new features

Originally downloaded OHLCV data from Binance contained below features inside a pandas data frame.

1.1.1. Open
1.1.2. High
1.1.3. Low
1.1.4. Close
1.1.5. Volume
1.1.6. Close Time
1.1.7. Quote Asset Volume
1.1.8. Number of Trades
1.1.9. Taker buy base asset volume
1.1.10. Taker buy quote asset volume
1.1.11. Ignore

Out of above features, only Open, High, Low, Close, Volume features were used for calculations and the labelling. When the download was complete, rest of the features were dropped from each data frame and a set of new features as below were calculated based on the selected features as well.

I. Price = (Open + High + Low + Close) /4; represents the average price for the given 1-minute candle

II. Turnover = Price * Volume; represents total turnover made by all the trades during the 1-minute candle

III. isOpenToClosePriceAbove10P = using a moving window of 10 minutes to the left respective to each data point, each 1-minute candle was given this Boolean feature whether Price at the end of the window (the time of the data point) was at or above 10% of the Price at the start of the widow. Figure 3.2 below shows a simple example case of a 10% Price increase during a 10-minute window.

Figure 3.2 Price 10% pumping during 10 minutes example

IV. isOpenToClosePriceAbove15P = like (c) above, this feature was also calculated over a moving window of 10 minutes to the left respective to each data point, representing a Boolean feature whether Price was at or above 15% of the Price at the start of the window. Figure 3.3 below shows a simple example of a 15% Price increase over a 10-minutes window.

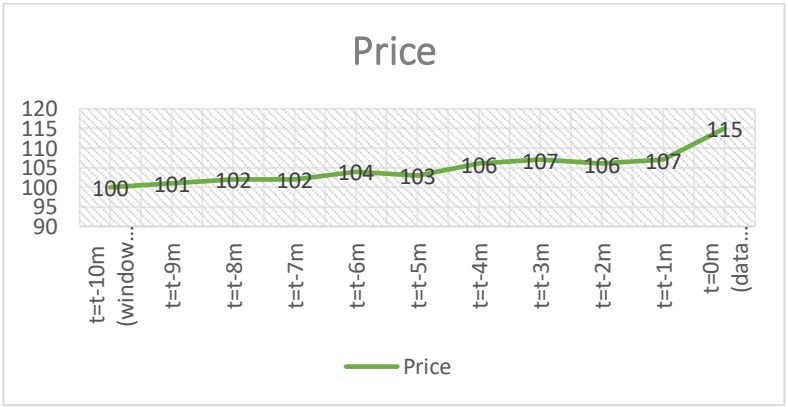


Figure 3.3 Price 15% pumping during 10 minutes example

V. isOpenToClosePriceAbove20P = like (c) and (d) above, this feature was also calculated over a moving window of 10 minutes to the left respective to each data point, representing a Boolean feature whether Price was at or above 20% of the Price the start of the window. Figure 3.4 below shows a simple example of a 20% Price increase over a 10-minutes window.

Figure 3.4 Price 20% pumping during 10 minutes example

VI. isWindowClosingTurnoverSignificant = represents whether the Turnover value was significant at or above by a given percentage (75% in this case) than the cumulative sum of Turnover values within a period of 5 minutes window to the left to that point. Figure 3.5 below show a simple example case 75% Turnover significance at the end of the window.

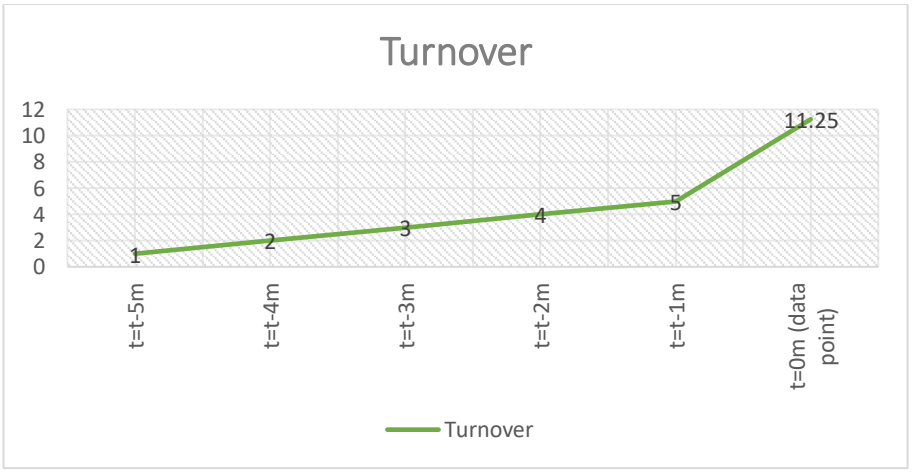


Figure 3.5 Turnover 75% significance during 5 minutes

VII. isLeftWindowPumping = represents a Boolean feature whether a rolling window of 10 minutes to the left contains a consistent price increase such that

Price at beginning (100) < Price at middle (104) < Price at the end (110) of the window. Figure 3.6 below shows and example case where t=0 candle is given isLeftWindowPumping = 1.

Figure 3.6 Left window representing an overall price increase

VIII. isRightWindowDumping = represents a Boolean feature whether a rolling window of 10 minutes to the right of each data candle was such that Price at start (100) > price at window center (20) and Price at start (100) > Price at window end (40). The reason for not taking a consistent decrease was owing to the cases where in case of a PND, Price could suddenly drop and then slightly increase afterwards during a 10 minutes window. Figure 3.7 below shows an example case of such a price decrease during a 10 minutes window.

Figure 3.7 Window to the right represents overall price decrease

3.4 Noise in the downloaded data

It was observed that for some coins, downloaded data contains noise in some of the features. This was evident during the labelling process with the number of detected anomalies being abnormally high and full view graphs created for a given coin as shown in Figure 3.14 are observed to contain noise. It is a highly unlikely to have N number of PND anomalies within N number of days and all the such high number of detections are only due to the noise in data. Cases like Figure 3.8 seems to have such noise these are removed manually and during labelling whenever observed in order to maintain a clean dataset.

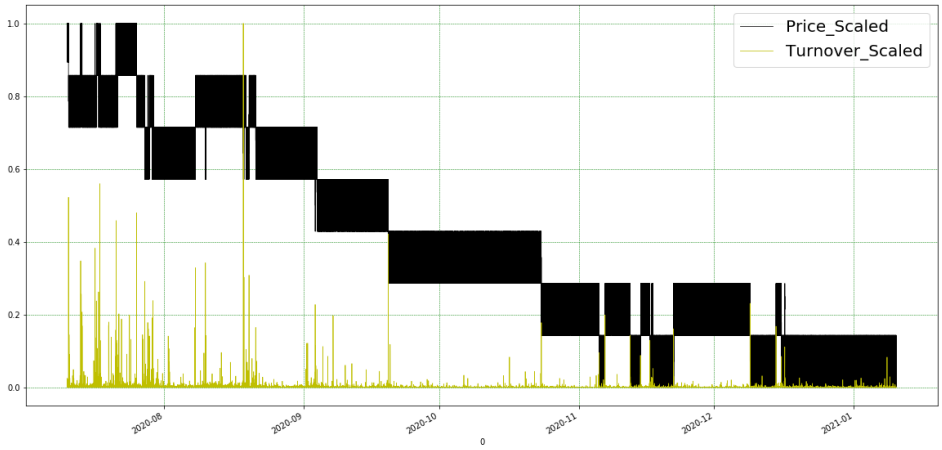


Figure 3.8 Noise in HOTBTC full view Price and Turnover

3.5 Labelling the data

This research followed the approach of a supervised learning for which pre-labelled data were required for model training. However, unavailability of pre-labelled data was the major hindrance to a research tasks specially in the domain of capital markets because of the confidentiality of data and hence were not publicly available (Level 2 data). In the case of this research also, the trade participant details were not available via Binance APIs, whereas only OHLCV data were present (Level 1 data), and it was really a challenge to derive the ground truth for a realistic pump and dump scenario.

Therefore, the alternative approach followed in this research was to come up with an initial set of labels by numerically detecting pump and dump shapes over different features such as Price and Volume and then got experts to review them. The inspiration for this approach was based on the way an experienced market surveillance personal would identify a possible PND by eyeballing various financial charts to detect an anomaly. Similarly, pre-trained image classifiers such as CNNs were intended to perform above task similar to an experienced market supervisor to detect the PND pattern from the encoded feature charts.

As the name would suggests itself, there are two stages for a pump and dump which are pumping stage and dumping stage as shown in Figure 3.9 below.

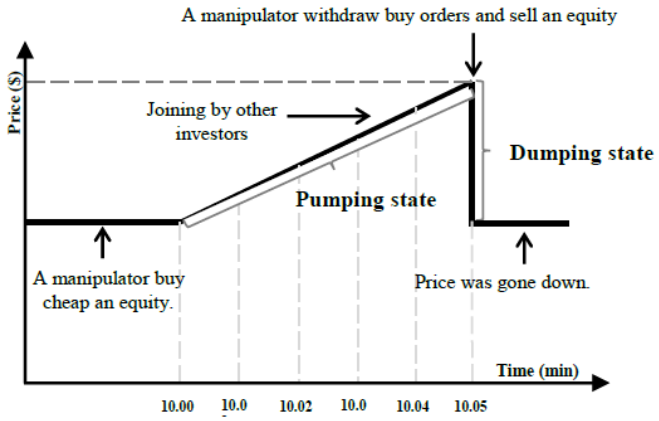


Figure 3.9 Basic stages of a pump and dump

Sources: [8]

Although in a real scenario, above graphs would look similar to as shown in Figure 3.10 [20] which is a real scenario where it can see the Price movement of coin POLY after being announced (dashed line) in a Telegram channel for a pump and dump. The price has reached its maximum after 25 seconds resulting it to be trading a volume of $1 million just after one minute.

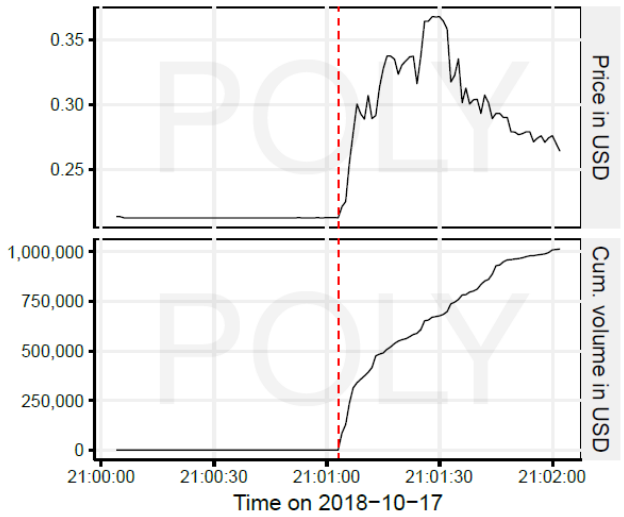


Figure 3.10 Real pump and dump instance of POLY

Sources: [13]

In order to get started with model training and to provide with labelled samples, a numerical labelling logic was applied on above downloaded time series data by using the features given on **section 4.3** as below.

$$label\ as\ PND = yes\ only\ if,$$

$$(isOpenToClosePriceAbove15P = 1)\ \&$$

$$(isWindowClosingTurnoverSignificant = 1)\ \&$$

$$(isLeftWindowPumping = 1)\ \&$$

$$(isRightWindowDumping = 1)$$

Figure 3.11 Initial labelling logic

Had it used *isOpenToClosePriceAbove10P* or *isOpenToClosePriceAbove20P* features in the place of *isOpenToClosePriceAbove15P* feature with all other features, it would have detected 10% or 20% price increase during a 10 minutes window respectively and the number of detected cases would also vary based on that where there are a lesser number of cases detected with 20% price increase.

After analysing data downloaded for 4 months and 6 months periods, it was observed that to have a sufficient number of training instances, data set needs to span to contain few years of data. Therefor a data set of 2 years was downloaded from Binance and it has taken approximately 2.5 days to complete on a ml.m5.12xlarge aws

sagemaker notebook instance and 1 day to enrich new features listed on **section 4.3** and few hours to label each data points in csv files on a ml.m5.12xlarge aws sagemaker notebook instance. Each downloaded csv file for a coin were checked for a pump and dump using Figure 3.11 and 3 random samples outside of the vicinity of 20 minutes from detected cases are also selected, labelled and saved as non-pump and dump cases to finally compose a binary classified dataset. A 20 minute of non-overlapping vicinity was selected such that to not include an anomalous case into a randomly picked non-anomalous window.

While numerically generating labels as PND and non PND for each csv file for a coin, line charts for features such as Volume, Price, Turnover are drawn by centring the selected data point at the middle of time axis such that neighbouring data points of 20 minutes to the left and right are selected. These graphs were just to be reviewed later by a human user.

Timestamp for the selected data point is formatted in yyyy-mm-dd_HHMMSS format and all above graph artifacts were saved under the name of the coin as shown in Figure 3.12 and Figure 3.14. In those figures, each level is expanded once to increase clarity, ex: in the place of coin name in Figure 3.14 Director hierarchy within s3 bucketthere can be many coin names. The reason for having below like directory structure was enabling repeatable experiments while unaffecting other experiments. A new set of data can be downloaded expanding at level 1 in Figure 3.12 Directory hierarchy within s3 bucket, any feature combination or encoding technique could be experimented to be expanded at level 3. Any number of coins could be detected as PND at level 6.1 and non-PND at level 6.2 and any coin can have as many timestamps detected at level 7.1 and 7.2 and so on.

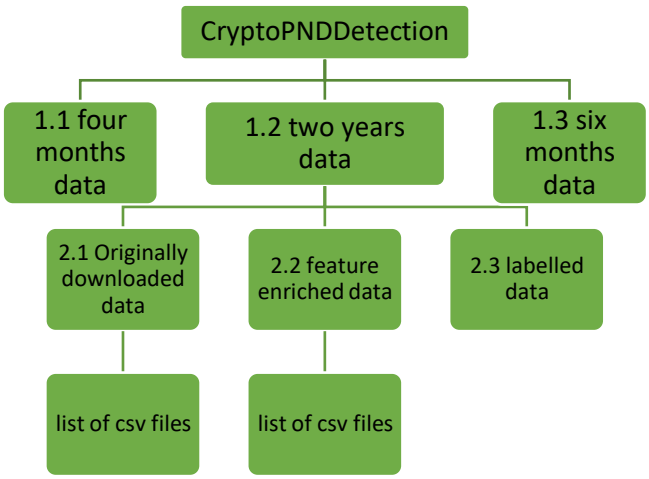


Figure 3.12 Directory hierarchy within s3 bucket

As an example for a PND labelled instance, Figure 3.13 coin and its Price, Volume, and Turnover variation charts during that period. Note that the aforementioned timestamp (2018-07-03_233400) was the middle point of the time axis in each graph and each dot on the graph represents a data point of 1-minute OHLCV candle. On these graphs it could be clearly seen the rapid movement of Price and Volume giving its peak (pumping stage) at the anomalous timestamp and then settling back to its true value after the timestamp on dumping stage.

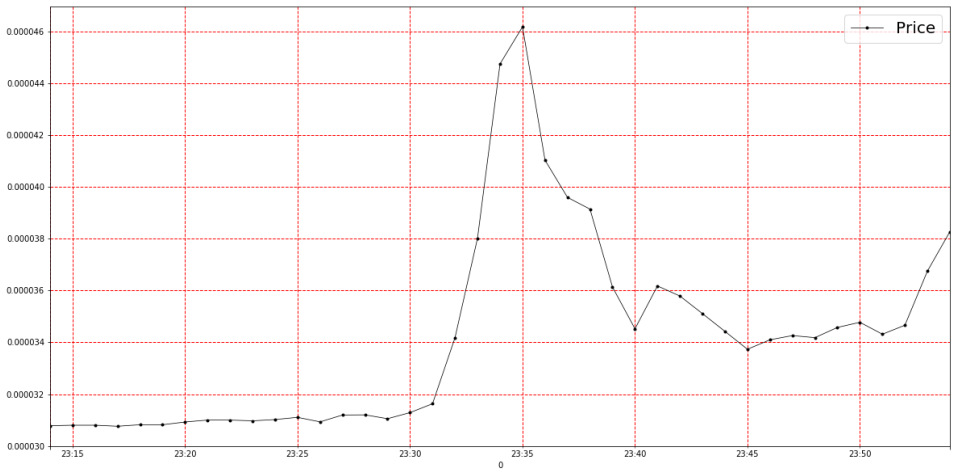


Figure 3.13 PND Price variation of APPBTC 2018-07-03_233400

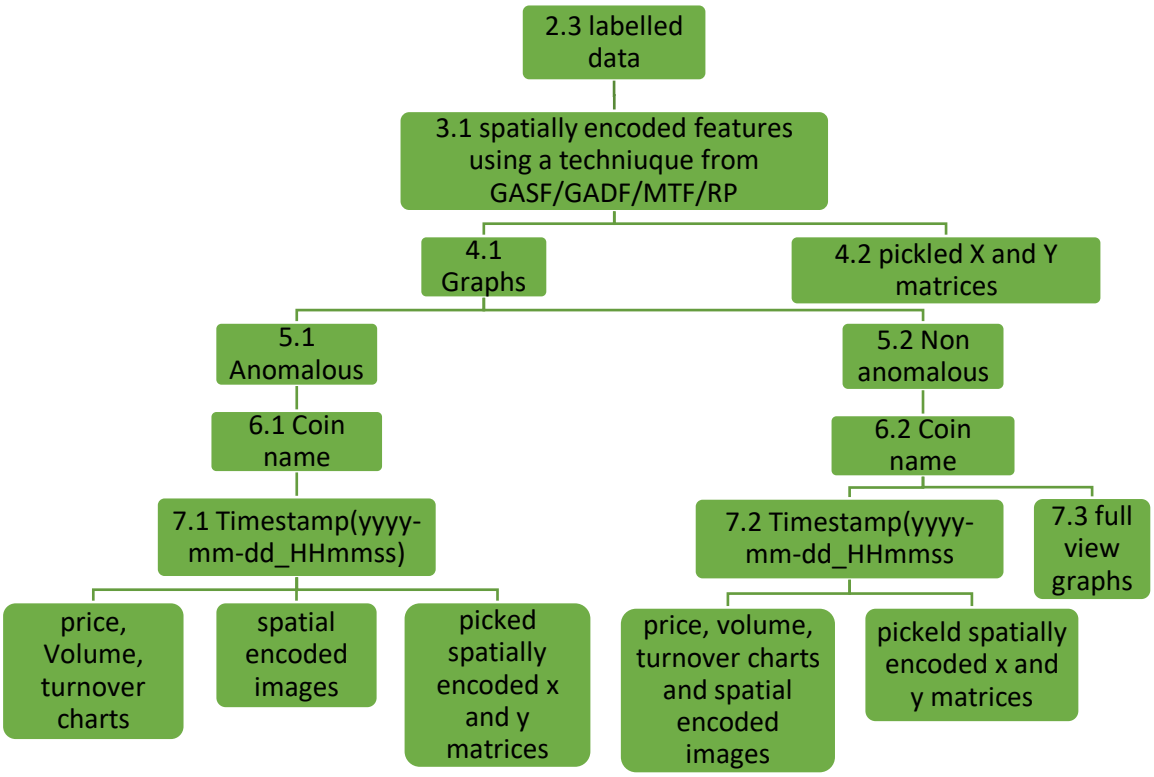


Figure 3.14 Director hierarchy within s3 bucket

When a particular time instance (say=t) was detected as an PND anomalous, a time window of [t-T min, t+T min] was taken for spatial encoding using the techniques listed in **section 3** by picking a set of features out of OHLCV, Price, Volume and Turnover. Number of experiments were performed to find an optimal value for T and the choice of features for spatial encoding and the temporal to spatial encoding technique. More details on this are explained in **Experiments** section. Similarly Figure 3.17, Figure 3.18 and Figure 3.19 it can be observed that the shape of these graphs at the timestamp point does not have the shape of a PND.

Above labelling process was repeated for all the downloaded coins while selected set of features are encoded using a technique out from GASF, GADF, MTF, RP resulting an encoded matrix X and the corresponding binary label Y (1=PND, 0=non-PND) to be picked dumped along with the above line charts so that after reviewing the graph, it could be allowed to be used as a real label to be incorporated while model training in a later stage.

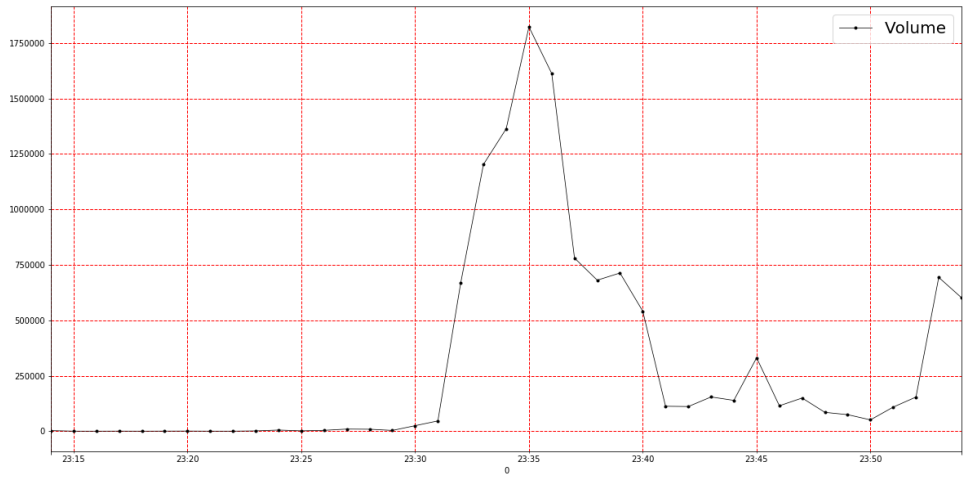


Figure 3.15 Volume variation during PND of APPBTC 2018-07-03_233400

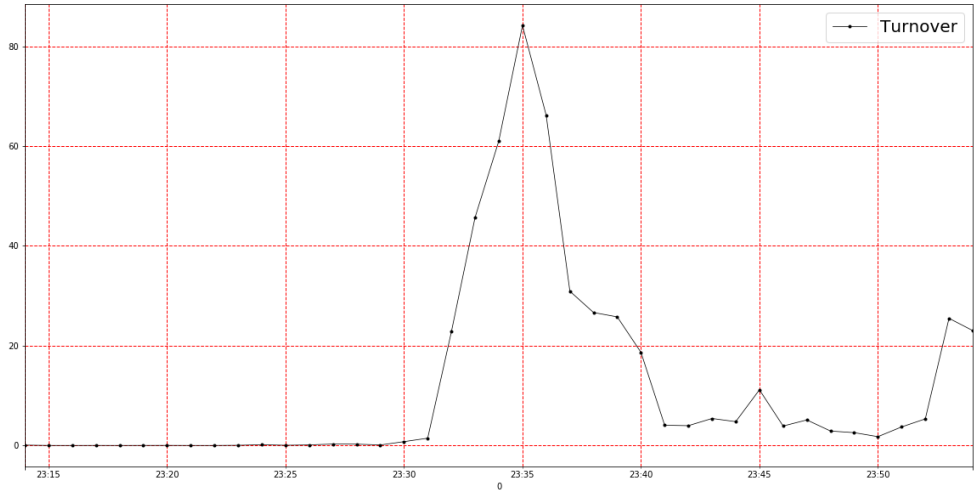


Figure 3.16 Turnover variation during PND of APPBTC 2018-07-03_233400

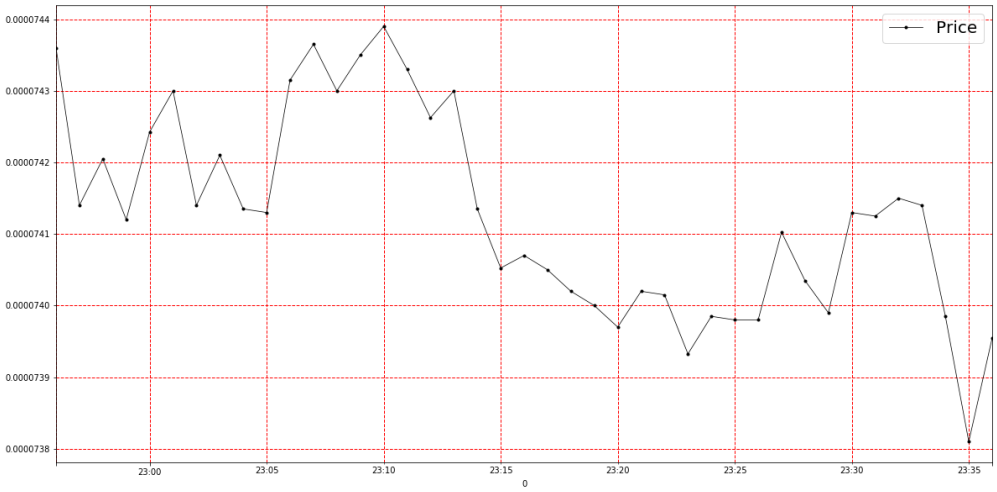


Figure 3.17 Price variation during non PND of APPCBTC 2018-05-04_231600

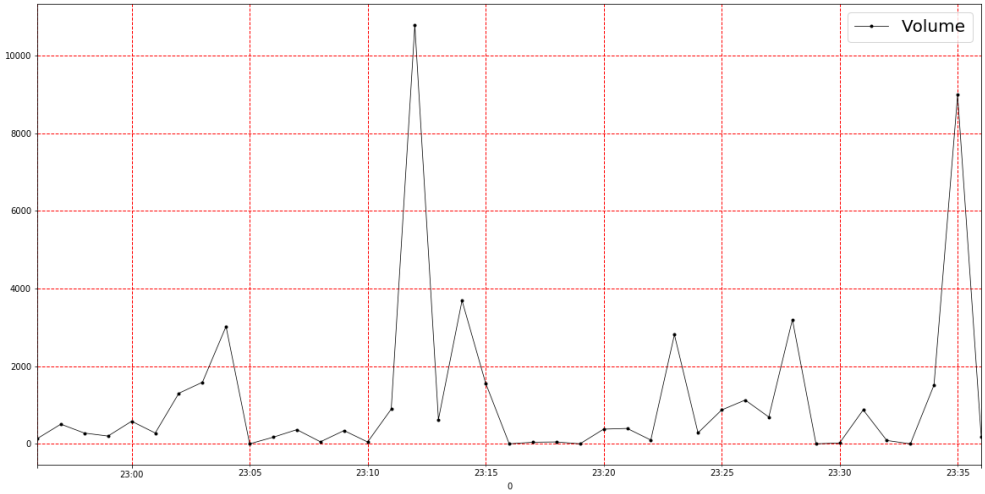


Figure 3.18 Volume variation during non PND of APPCBTC 2018-05-04_231600

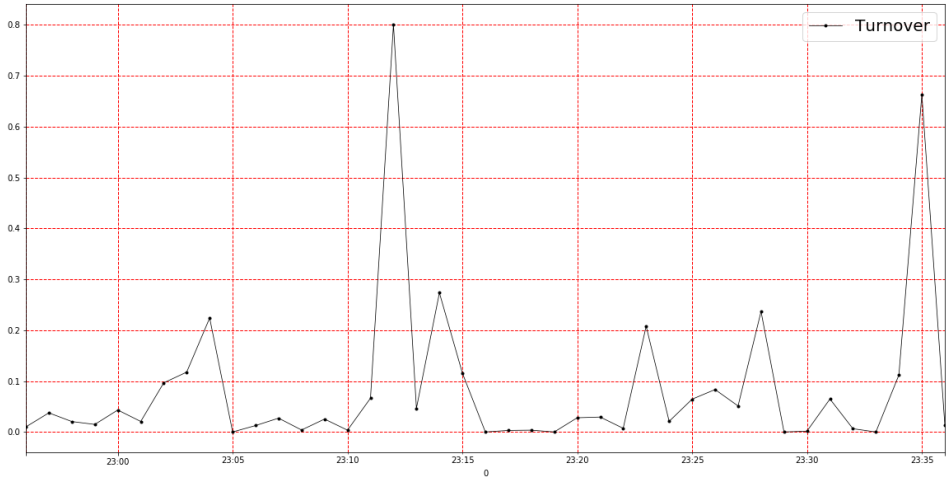


Figure 3.19 Turnover variation during non PND APPCBTC 2018-05-04_231600

3.6 Temporal to spatial encoding

In this research, to make the use of computer vision over the pump and dump classification, n-dimensional financial timeseries data were required to be represented in the form of images in order to train convolution neural networks. This is most intuitive graphical representation of a time series data simply by drawing the chart for feature such as trade price or trade volume against time over a particular time window. ex: - 10 minutes, and Figure 1.3 above shows one of such line charts. Although in this research such images were not directly used for model training. Listed below are the details of such methods intended to be used to convert and represent time series data as images in the process known as temporal to spatial conversion.

3.6.1 Gramian Angular Field (GAF)

Given a time series data $X=\{x_1,x_2,...,x_n\}$of n real valued observations, X is min-max scaled to interval [-1,+1] as given in Figure 3.20 below.

$$\tilde{x_i} = \frac{(x_i - max(X) + (x_i - min(X))}{max(X) - min(X)}$$

Figure 3.20 Normalizing time series data

Sources: [5]

Above rescaled values are then represented in polar coordinates by encoding the value as an angular cosine and the timestamp as the radius as shown in Figure 3.21 below.

$$\begin{cases} \phi = \arccos(\tilde{x_i}), -1 \leq \tilde{x_i} \leq 1, \tilde{x_i} \in \tilde{X} \\ r = \frac{t_i}{N}, t_i \in \mathbb{N} \end{cases}$$

Figure 3.21 GAF angular cosine representation

Sources: [5]

In above equation t_i is the timestamp and N is a constant factor to regularize the span of the polar coordinate system [13]. With this novel way to represent time series data, when time is increased corresponding values warp among different angular points on the spanning circles. This representation has few special properties such as bijectivity since cos(α) values are monotonic in α € [0,π] and polar coordinates still preserve the temporal relations [13]. After above transformation of each point in rescaled time series, angular correlation of each point is calculated by considering the trigonometric sum between each point to calculate below GAF matrix as in Figure 3.22.

$$\begin{aligned} G &= \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{bmatrix} \\ &= \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \end{aligned}$$

Figure 3.22 Gramian angular field matrix

Sources: [5]

Above G is called the Gramian matrix and it provides a way to represent and preserve the temporal dependencies of a time series such that when time increases the position moves from top left to bottom right direction and ends up being a large n*n matrix where n is the number of points in the time series. Figure 3.23 below illustrates the steps converting a time series data into a GAF.

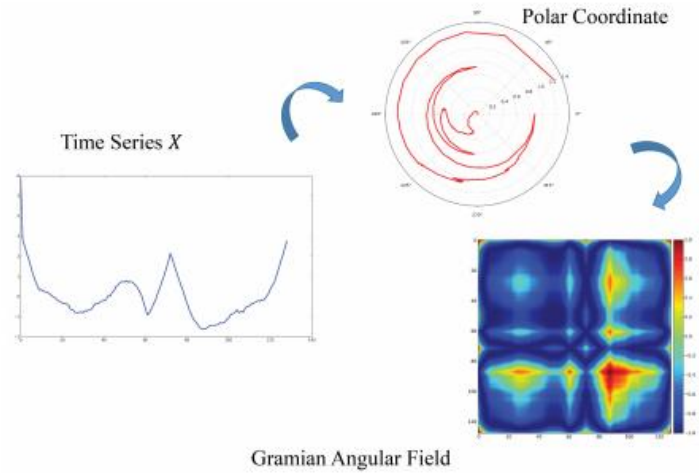


Figure 3.23 Gramian Angular Field encoding summary

Sources: [5]

3.6.2 Markov Transition Field (MTF)

In Markov transition field representation, the idea of Markov transition probabilities is taken into consideration. First, the time series data is realized into two axes one to represent the magnitude and other to represent time. Given a time series X, the magnitude axis of the data is then divided into Q quantile bins and assign each point of the time series x_i to a corresponding bin q_j where j € [1, Q]. Then a Q x Q weighted adjacency matrix is created by counting the transitions among each quantile bins so that a first order Markov chain is represented. In this matrix $w_{i,j}$ represents the frequency of points falling to quantile q_i is followed by the points falling to quantile q_j. After normalizing such that $\sum w_{i,j} = 1$, this becomes the Markov transition matrix W. Although, Markov transition matrix W loses the temporal dependencies along time axis causing an information loss. To overcome this issue, Markov transition field [13] is defined as below Figure 3.24.

$$M = \begin{bmatrix} w_{ij}|x_1 \in q_i, x_1 \in q_j & \cdots & w_{ij}|x_1 \in q_i, x_n \in q_j \\ w_{ij}|x_2 \in q_i, x_1 \in q_j & \cdots & w_{ij}|x_2 \in q_i, x_n \in q_j \\ \vdots & \ddots & \vdots \\ w_{ij}|x_n \in q_i, x_1 \in q_j & \cdots & w_{ij}|x_n \in q_i, x_n \in q_j \end{bmatrix}$$

Figure 3.24 Markov transition field

Sources: [5]

Markov transition field is an n x n matrix where each element $m_{i,j}$ represents the Markov transition probability from the quantile time instance i belongs into the quantile time instance j belongs to. If the time series represents a large amount of data, to make the matrix more manageable and computation more efficient, above Markov transition field matrix is resized by averaging the pixels in non-overlapping m x m patches with a blurring kernel by taking the summation over the m x m patch and then averaging by dividing by m^2 [13]. Figure 3.25 summarize the steps in generating a Markov transition field and blurred Markov transition field.

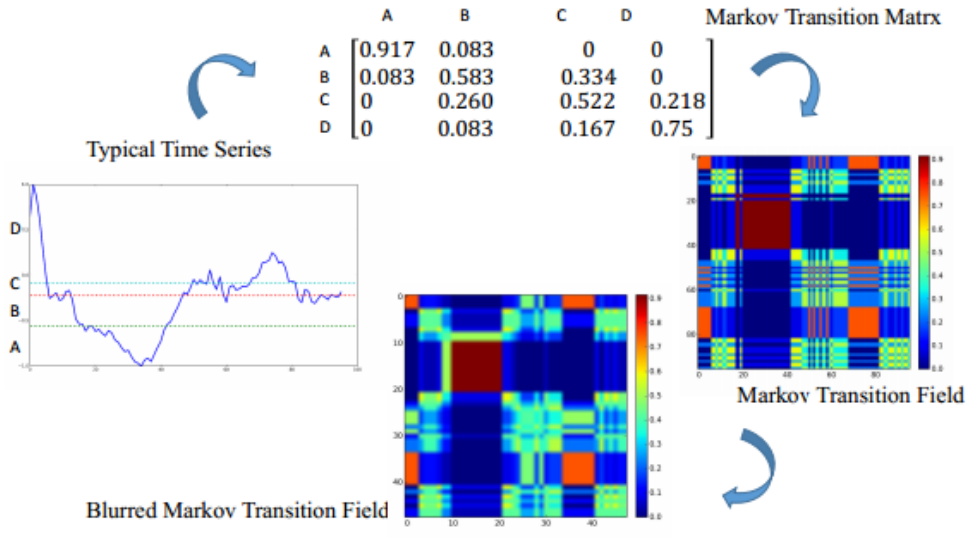


Figure 3.25 Markov Transition Field encoding summary

Sources: [5]

3.6.3 Recurrence Plots (RP)

Recurrence behaviors such as seasonality is often present in time series data and recurrence plot is one of the methods to visualize such behaviors in time domain. Recurrence plots create a representation for m-dimensional trajectories in a two-dimensional phase space [23] and it can represent points in the trajectory which has returned to previously visited states. Recurrence plot visualize the recurrence matrix

$R_{i,j}$ in which there will be value 1 at position (i, j) if the m-dimensional trajectory of the time series at time j is close within a pre defines neighborhood to the observation at time i and value 0 otherwise. This is mathematically given in Figure 3.26 shown below.

$$R_{i,j} = \Theta(\epsilon - ||\vec{x}(i) - \vec{x}(j)||), \vec{x}(\cdot) \in \Re^m, i, j = 1..N$$

Figure 3.26 Recurrence matrix equation

N is the number of states and $\vec{x_i}$ and $\vec{x_j}$ are the subsequences observed at positions i and j and ||.|| is the Euclidean norm between those observations. ∈ is a threshold for closeness and θ is the Heaviside step function given by below Figure 3.27

$$\Theta(z) = \begin{cases} 0, & \text{if } z < 0 \\ 1, & \text{otherwise} \end{cases}$$

Figure 3.27 Heaviside function

The resultant matrix is N x N pixel image with black at value 1 and white at value 0. Figure 3.28 below shows few samples of such black and white recurrence plots [23].

Figure 3.28 Sample Recurrence plots; totally random noise (left); random walk (middle); periodic composition of sine and cosine (right)

Sources: [23]

Despite its simplicity recurrence plots requires a value for its closeness threshold parameter ∈ and determining an appropriate value is not intuitive and cannot generalize and hence need to follow heuristics. To eliminate ∈ it can be introduced color to the image representing the distance. With this change recurrence matrix directly represents distances depicting how close each observation in its trajectories

and known as unthresholded recurrence plots [23]. Figure 3.29 below shows such generated recurrence plots. [15]

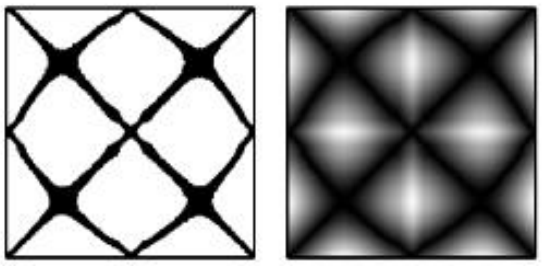


Figure 3.29 Thresholded (left) and an unthresholded (right) recurrence plots, generated from a same time series

Sources: [15]

3.7 Evaluating labelling logic

In this research, the decision to use a numerical labelling process was only due to the dearth of labelled data available in the context. However, it is possible to raise the question, why can't the labelling process itself be used for the final task not going with machine learning or machine vision. Well, the idea behind this was detecting the unknown unknowns, where after a fine training, the models are expected to detect cases which have gone as false positives and false negatives from the numerical labelling logic/adaptive threshold-based models such that visual encoding technique and neural training captures a new set of features to identify that was not detected via the conventional techniques.

However, in order to have a measure over the accuracy of the labels created by the numerical labelling logic mentioned in **section 4.4**, a random sample of the labeled (price and volume charts) detected by above logic was shared without the labels to 5 market surveillance experts in London Stock Exchange group such that the same instance was reviewed by multiple experts(at least two reviewers) to get a kappa measurement as given in Table 3.2. Out of the 5 experts, 2 of them returned without giving an evaluation with the stance that they need more data such as trade participant data, number of buy/sell trades to be precise of labelling as a pump and dump. 3 out of

5 experts agreed with the numerically labels to be correct. Therefore, labels were accepted as trust worthy.

Table 3.2 Kappa values for the labels

		Expert 1		
		Agree the Label	Disagree the Label	Sub Total
Expert 2	Agree the Label	A=42	B=9	A+B=51
	Disagree the Label	C=0	D=19	C+D=19
	Sub Total	A+C=42	B+D=28	A+B+C+D=70

$$Expected\ agreement = \frac{\big((A+B)*(A+C)\big)+\big((C+D)*(B+D)\big)}{A+B+C+D}$$

Figure 3.30 Calculating expected agreement

$$Kappa = \frac{(Observed\ agreement) - (Expected\ agreement)}{(A+B+C+D) - (Expected\ agreement)}$$

Figure 3.31 Calculating kappa value

Considering observed agreement as A+D and Expected agreement as Figure 3.30 Kappa value of **0.7169** was calculated for the labelling according to Figure 3.31 over the expert reviews, which falls to the range [0.61-0.81] known as the substantial band. However, since the numerical labelling was based on hardcoded thresholds, it is susceptible to noise and is not capable of identifying a particular shape out of the graphs. Appendix E, Appendix F, Appendix G and Appendix H has listed some of the cases reviewed by the experts and their real feedback on the charts. It can be seen that majority of the original label which were hidden to the expert agree with the feedback.

From the expert feedback it could be observed that some of the experts were requesting level 2 data (data only available at exchange level) to proceed with the decision whereas some experts did agree with the original label based on the given shape as to a pump and dump.

3.8 Generating a synthetic data set using SMOTE

After obtaining the reviews from the experts over the initial set of labels as mentioned in section 3.7 above, a pool of PND positive and negative cases was created so that it can be used as the ground truth to generate a synthetic data set using the originally downloaded data. This approach was considered owing to the unavailability of a labelled data set on this problem of pump and dump detection. In this phenomenon only the expert agreed labels generated by the above numerical labelling logic were used such that SMOTE (Synthetic Minority Oversampling Technique) technique can be applied to generate a balanced synthetic data set. The choice of SMOTE technique was due to availability of public python APIs to computationally apply SMOTE over any data set. Python's imblearn library has provided a comprehensive set of functions to apply three well known variants of SMOTE technique [24]. Figure 3.32 below shows the class distribution of the expert reviewed cased before applying SMOTE.

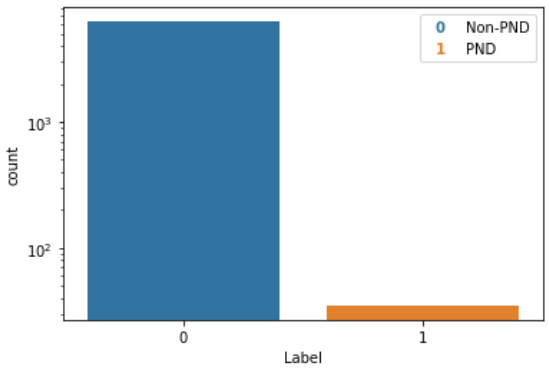


Figure 3.32 Class distribution before applying SMOTE

Figure 3.33 below shows the feature distribution after PCA was carried out on 2D. Original dimensionality of the data before doing PCA for visualization was 84 features such that each feature was generated by introducing lagged features over original features in a window of 10 minutes.

Ex:-

Price → Price t-10, Price t-9,…,Price t-1, Price, Price t+1, …, Price t+10.

→ resulting 21 new Price features.

Similarly, 21 new features were introduced to Price, Volume, Close and High features accordingly resulting total of 84 lagged features. This was because to apply SMOTE, features need to represent a feature space for each data point rather than a single data point and Pump and dump is a context anomaly problem where the neighboring data points should also consider for the detection. In this case 10 minutes to the left and 10 minutes to the right from each data point were considered.

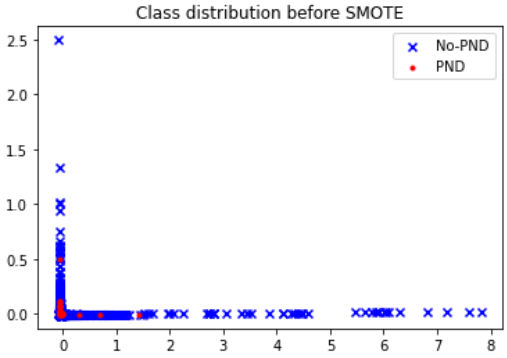


Figure 3.33 Class distribution before SMOTE after PCA

3.8.1 General SMOTE

When the data used for a model training are imbalanced between classes, the models are trained with a biased towards the majority class and its prediction results not generalized. But when the class distribution is heavily imbalanced as shown in Figure 3.32, it has detrimental consequences to the model. Therefore, class balancing plays a vital role in machine learning. There are different approaches to tackle this issue and the most common and intuitive way is to oversample the minority class and under sample the majority class. Although just oversampling the minority class to replicate the same instance multiple times in dataset until the class distribution is balanced enough causes the trained models having to see the same data point multiple times during the training and brings about model overfitting.

SMOTE is an oversampling technique which allows to synthetically generate samples over the minority class using the idea of k nearest neighbor. In general smote the minority class is oversampled by randomly selecting each minority data point and generating synthetic samples along the line segments combining that sample with each of its k nearest neighboring minority classes. Figure 3.34 below shows the class

distribution after applying general SMOTE over the expert annotated data set and Figure 3.35 below shows the visualization after doing PCA to 2D over the data set.

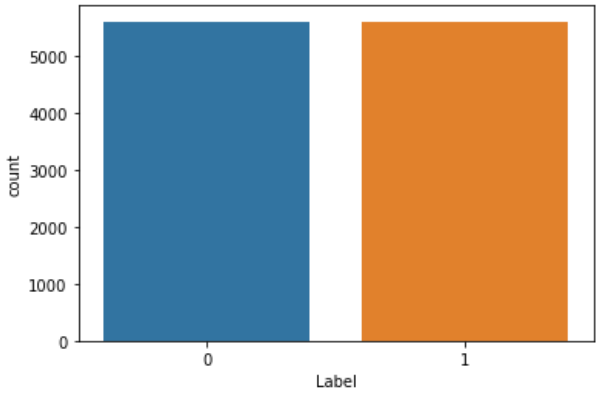


Figure 3.34 Class distribution after SMOTE

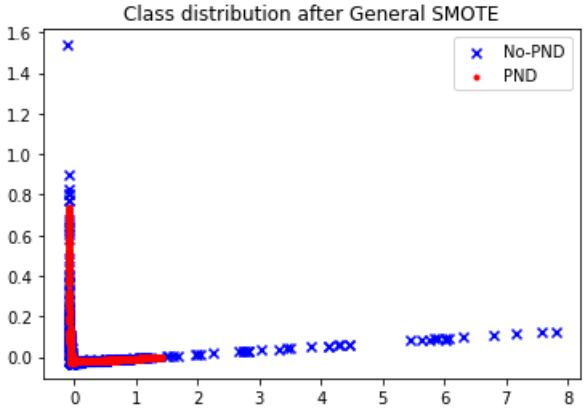


Figure 3.35 Class distribution visualized with PCA after general SMOTE

3.8.2 Borderline SMOTE

Borderline SMOTE is a popular extension to SMOTE technique such that the minority class instances which are misclassified over a k nearest neighboring are chosen to oversample and more synthetic samples are generated close to the cases where the classification is difficult due to being close to the border of the classes. Figure 3.36 below shows the class distribution of after applying borderline SMOTE and visualized with PCA to 2D.

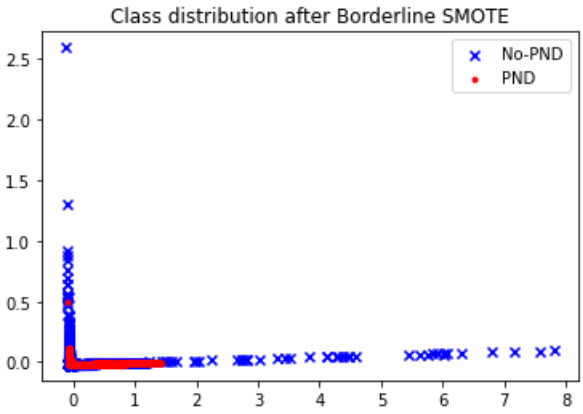


Figure 3.36 Class distribution visualized with PCA after borderline SMOTE

3.8.3 Adaptive Synthetic (ADASYN) SMOTE

In this SMOTE variant, synthetic samples from the minority class are generated being inversely proportional to the density of the minority class cases such that more samples for the minority class are generated in the regions where the minority class density is low. Figure 3.37 below shows the class distribution after ADASYN SMOTE was applied over the expert reviewed samples.

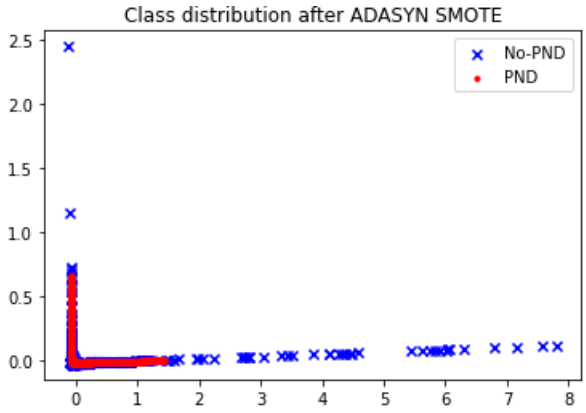


Figure 3.37 Class distribution visualized with PCA after ADASYN SMOTE

Hence 3 pools using 3 SMOTE variants were created using the expert annotated data which contained even the cases that were labelled as false positives by the numeric logic given in section 3.4 as well.

Random samples from above created SMOTEd pool were chosen and then applied to the time instances of the target data frames around 3 hours apart to each other avoiding seasonality by selecting points in t + 3h + [-20 min, +20min] range. Each SMOTE window values for selected feature were scaled accordingly to match the target feature window selected as shown on below example.

- Say, Selected SMOTE window = [**10**, 20, 30, 40, 50, 40, 30, 20, 10]
- Target window = [**1000**, 2000, 1000, 1500, 6200, 2100, 900, 7000, 200]
- Then Resulting target window = [**1000**, 2000, 3000, 4000, 5000, 4000, 3000, 2000, 1000]

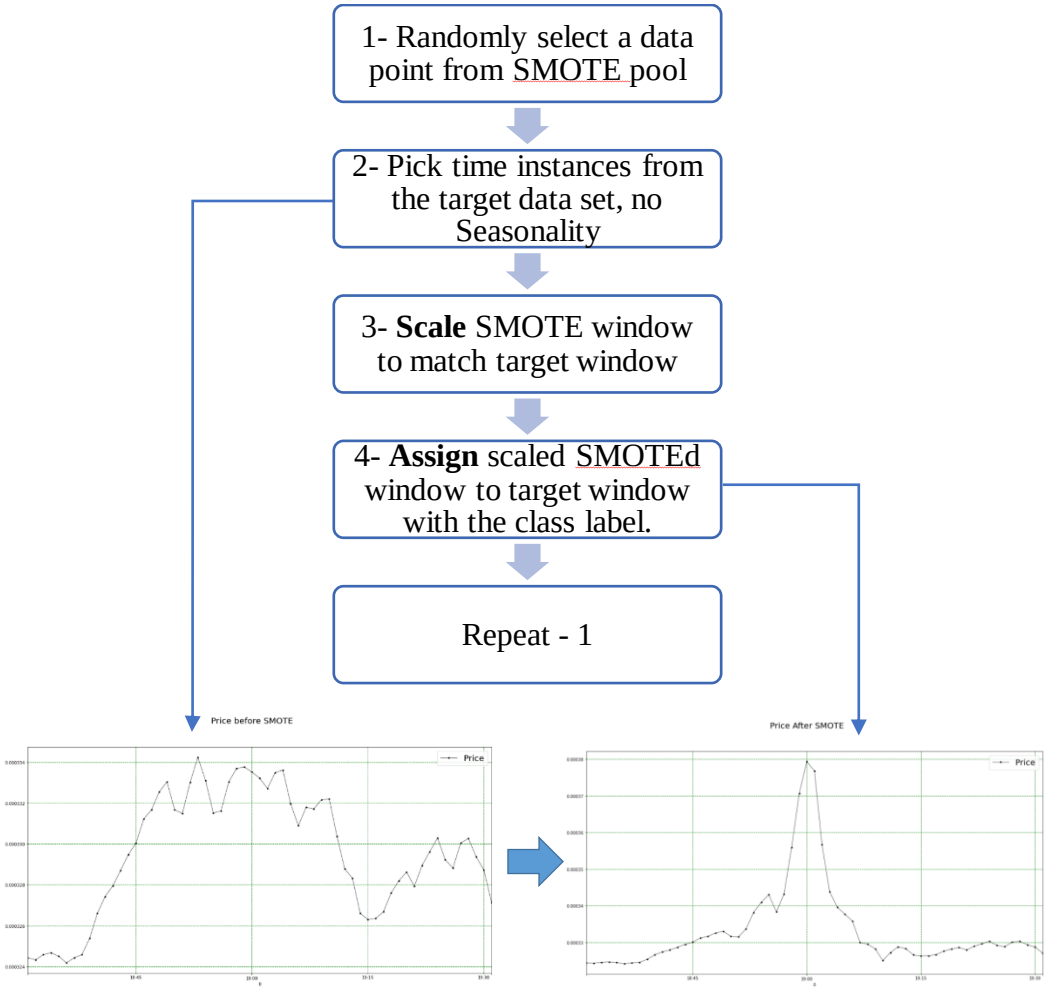


Figure 3.38 Applying SMOTEd instances to target data points

This way, the resulting window followed a similar shape as the SMOTE window in a different scale. Figure 3.38 above summarizes this procedure graphically. Finally, a synthetic data set was created by using 188 data files downloaded from binance each containing 1-minute OHLCV data for 2 years duration from Jan 1st 2018 to Jan 1st 2020 containing approximately 1 million data points (1-million minutes) and each pump and dump shape was captured over a period of 20 minutes so that 10 minutes to the left containing the pumping shape and 10 minutes to the right containing dumping shape. Figure 3.39 below shows the variation of Price feature over a duration of 2 years before and after applying SMOTE samples.

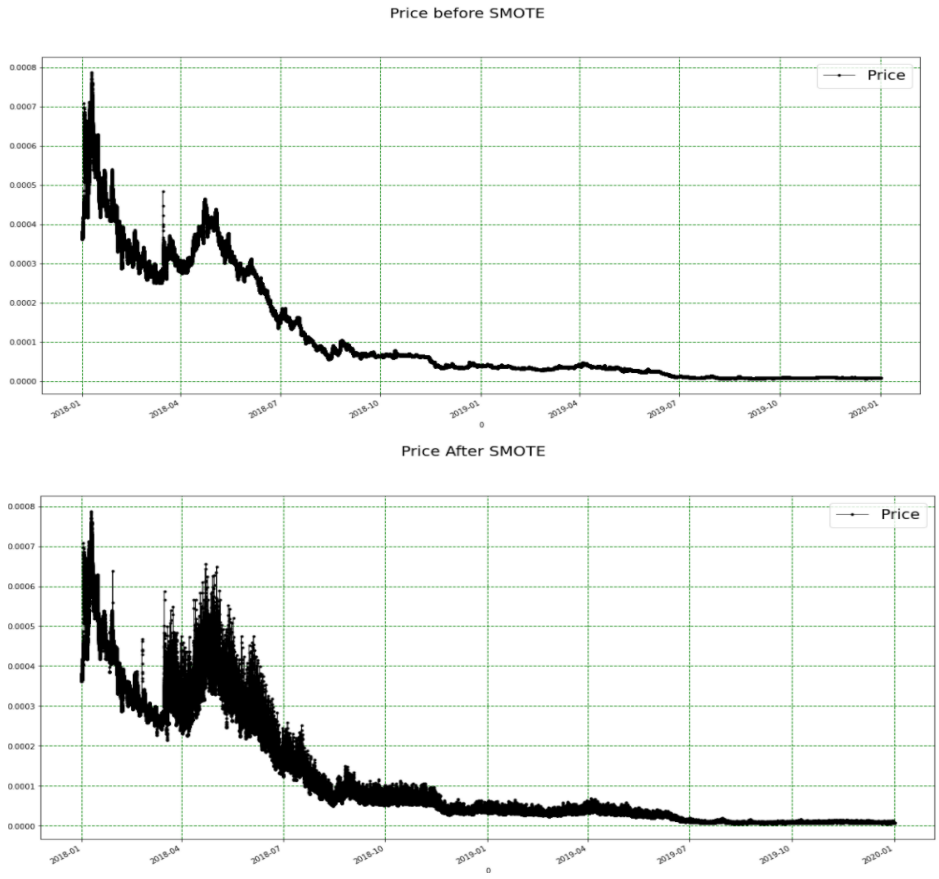


Figure 3.39 Before and after applying SMOTE cases over the span

Because of selecting target time instances 3 hours apart from each other in the target data frames to apply SMOTE samples, the class distribution for a given coin even after applying synthetic samples was like below Figure 3.40.

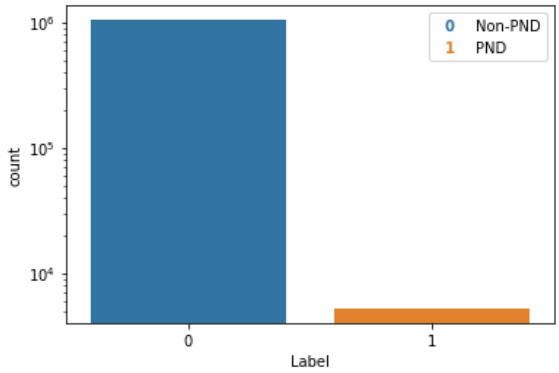


Figure 3.40 Class distribution for a coin after applying synthetic samples 3hrs apart

Therefore, still it could be seen that class imbalance problem was there such that positive class was having around 5200 cases and negative class was having around 1 million cases. Because of this nature, in order to do model training a data set was created by extracting an equal number of positive and negative cases from a set of 20 files out of the originally downloaded files on which the synthetic samples were applied, and the models were trained based on this data set. Figure 3.41 below shows the class distribution of the balanced data set created for the model training this way.

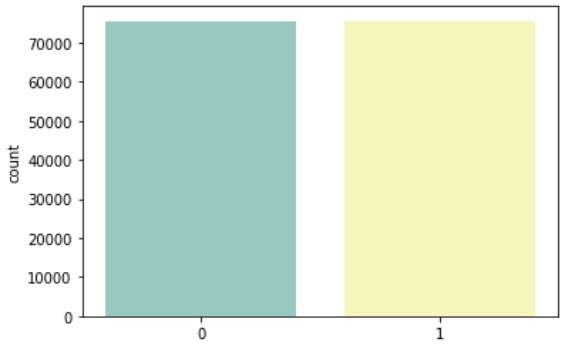


Figure 3.41 Balanced class distribution set for model training

CHAPTER 4

EXPERIMENTS

4 Experiments

A number of experiments were conducted on Amazon SageMaker notebooks trying to figure out an optimal mix of the feature for encoding, temporal to spatial encoding techniques to be used, encoded time window size, class balance between PND and non-PND, hyperparameters of different simple CNN architectures etc. Figure 4.1 shows those variables involved in the experiment.

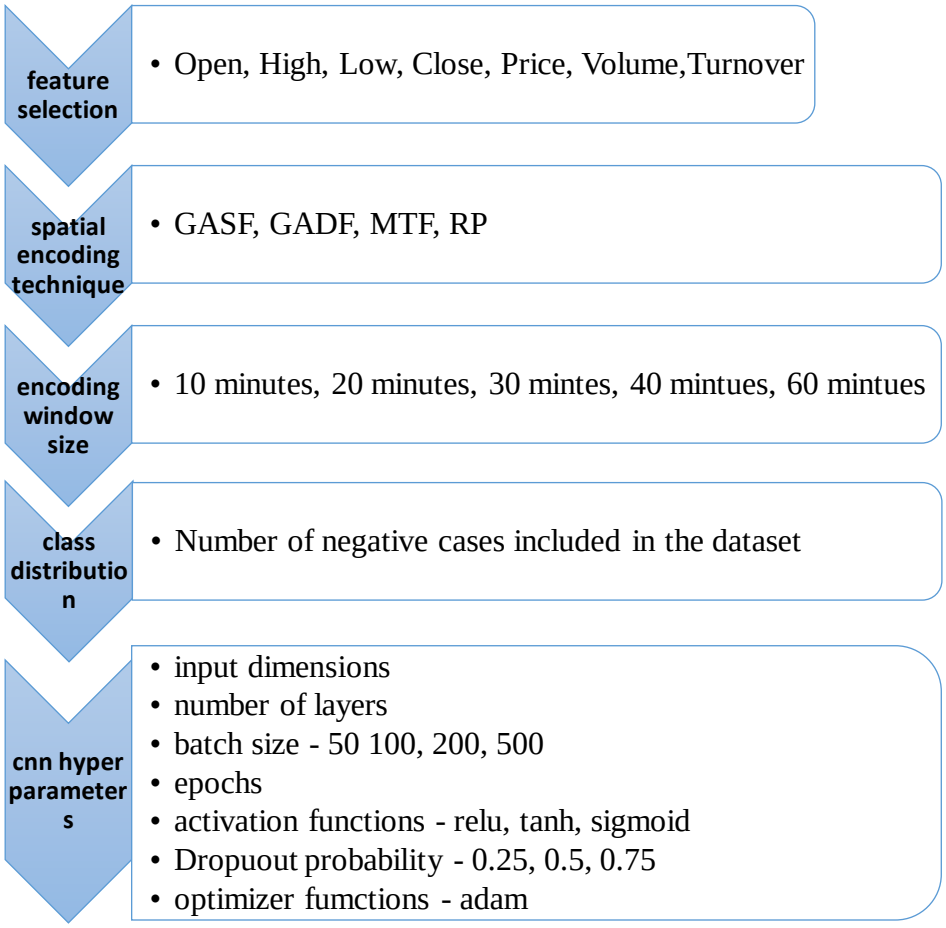


Figure 4.1 Variables of the experiment

Different notebook instances were created to support variations in above setup while increasing the repeatability on each experiment independently and the s3 directory structure given Figure 3.12 are also in the favor of this. The intended lifecycle of the whole process is as given in Figure 4.2.

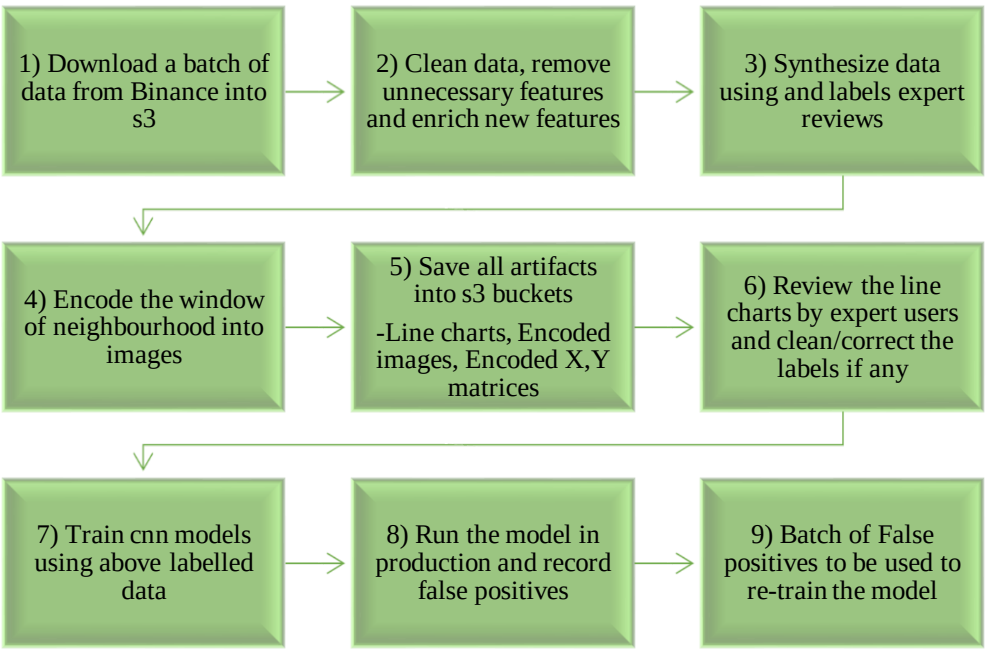


Figure 4.2 Summary of experiment lifecycle

4.1 CNN hyper parameter tuning

Multiple CNN architectures of which most were derived from the previous work by [18] were experimented with different spatial encoding techniques as given in section 3.6. Mainly below hyper parameters were tuned to derive the best model fitting.

- Batch size – 50, 100, 200, 500
- Number of epochs – 10, 20, 50, 80
- Activation functions – tanh, sigmoid, relu
- Whether to use pooling layers – yes, no
- Whether to use dropout layers – yes, no
- Dropout probability used in dropout layers – 0.25, 0.5, 0.75

A sample code used to achieve this is shown in Figure 4.3 below. After identifying the best parameters for model fitting, 10-fold cross validation was performed over the model created with those parameters and results were recorded on separate sagemaker notebook instances to increase repeatability.

```
from sklearn.model_selection import GridSearchCV

from keras import Sequential

from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

from keras.wrappers.scikit_learn import KerasClassifier

def create_model_64(dropout=True, pooling=True, dropoutp1=0.25, dropoutp2=0.5, active):

    cnn=Sequential()

    cnn.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation= active,
input_shape=(INPUT_MATRIX_WIDTH, INPUT_MATRIX_WIDTH, ENCODED_FEATURES)))

    if pooling==True:

        cnn.add(MaxPooling2D(pool_size=(2,2)))

    cnn.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation= active))

    if pooling==True:

        cnn.add(MaxPooling2D(pool_size=(2,2)))

    if dropout==True:

        cnn.add(Dropout(dropoutp1))

    cnn.add(Flatten())

    cnn.add(Dense(128, activation= active))

    if dropout==True:

        cnn.add(Dropout(dropoutp2))

    cnn.add(Dense(1, activation='sigmoid'))

    cnn.compile(loss='binary_crossentropy', optimizer= active, metrics=['accuracy'])

    return cnn

model = KerasClassifier(build_fn=create_model_64, epochs=25, verbose=0)

activation=['relu', 'tanh', ‘sigmoid’]

dropout=[True, False]

pooling=[True, False]

batch_size=[50, 100, 200]

dropoutp1 =[0.25, 0.5, 0.75]

dropoutp2 =[0.25, 0.5, 0.75]

param_grid=dict(dropout=dropout, pooling=pooling, batch_size=batch_size, dropoutp1=dropoutp1,
dropoutp2=dropoutp2, active=activation)

grid=GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)

grid_result = grid.fit(X_data, Y_data)

print("best %f using %s" %(grid_result.best_score_, grid_result.best_params_))
```

Figure 4.3 Hyper parameter tuning example

It took around a week to do the grid search over different CNN architectures on a ml.m5.12xlarge aws Sagemaker notebook instance.

4.2 CNN Model training

As given on section **2**, there was a strong evidence that even simple CNN architectures have been able to identify time series patterns over the financial charts after being spatially encoded. The CNN architecture given in [25] was used as the base architecture for most of the below models while varying number of layers, number of neurons under each layer and their hyperparameters such as dropout probabilities, batch size, epochs, whether to use pooling layer or not. Listed below are only a sub set of the experiments conducted which have given better results. Data set used for below experiments spanned 2 years from Jan 1st 2018 to Jan 1st 2020 from BTC paired crypto currencies' 1-minute OHLCV data taken from Binance exchange. All the models were 10-fold stratified cross validated in each of below experiments with 20% validation split. Results were taken as after observing the epoch value before model was overfit and as the average of each fold.

As shown in Figure 4.1 several experiments were carried out to identity the optimal features to spatially encode with different input dimension and numerous CNN architectures and shown below are only the best models out of that. Figure 4.4 and Figure 4.5 below shows the two CNN architectures identified to give the best results in this problem.

Layer (type)	Output Shape	Param #
conv2d_27 (Conv2D)	(None, 21, 21, 64)	576
conv2d_28 (Conv2D)	(None, 21, 21, 64)	16448
dropout_13 (Dropout)	(None, 21, 21, 64)	0
flatten_14 (Flatten)	(None, 28224)	0
dense_27 (Dense)	(None, 128)	3612800
dropout_14 (Dropout)	(None, 128)	0
dense_28 (Dense)	(None, 1)	129

Total params: 3,629,953
Trainable params: 3,629,953
Non-trainable params: 0

Figure 4.4 CNN3 architecture

Layer (type)	Output Shape	Param #
conv2d_21 (Conv2D)	(None, 21, 21, 64)	576
conv2d_22 (Conv2D)	(None, 21, 21, 64)	16448
dropout_21 (Dropout)	(None, 21, 21, 64)	0
flatten_11 (Flatten)	(None, 28224)	0
dense_21 (Dense)	(None, 256)	7225600
dropout_22 (Dropout)	(None, 256)	0
dense_22 (Dense)	(None, 1)	257

Total params: 7,242,881
Trainable params: 7,242,881
Non-trainable params: 0

Figure 4.5 CNN5 architecture

4.2.1 Price and Volume encoded with GADF CNN3

Price and Volume features were encoded using GADF technique over a neighborhood of 20 minutes (10 mints to the left and 10 mints to the right) resulting each data point/candle to be of dimension (21, 21, 2). Figure 4.4 below shows the CNN 3 model architecture and Figure 4.6 below shows the averaged confusion matrix during 10-fold cross validation. The model was re trained with the full data set again and saved as a .h5 file with the weights and parameters so that it can be loaded again to predict over unseen data. Code used to generate GADF CNN3 is available at Appendix B. Table 4.1 below shows the averaged results from 10-fold cross validation.

Table 4.1 10-fold cross validation results of GADF CNN3 model

	Macro average	Standard deviation
Precision	98.86%	0.1351
Recall	98.83%	0.1414
F1-score	98.83%	0.1414

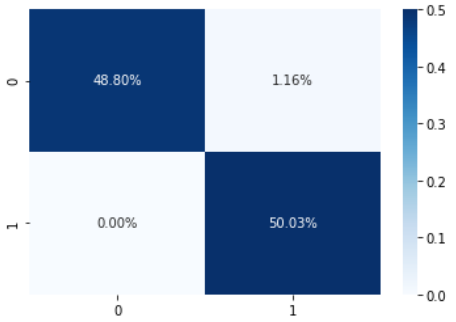


Figure 4.6 GADF CNN3 averaged confusion matrix

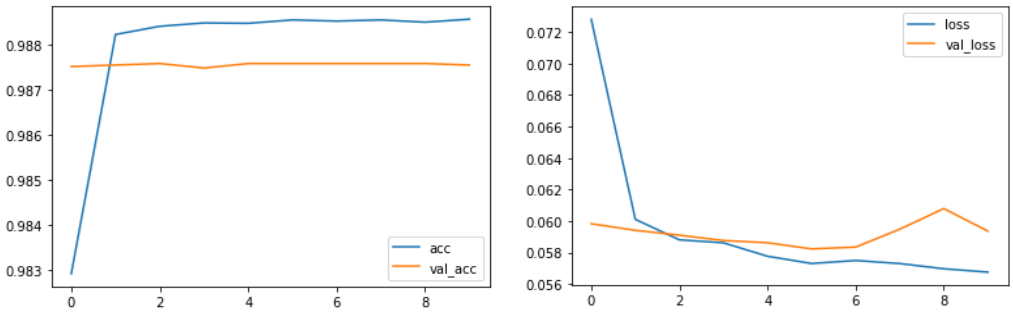


Figure 4.7 GADF CNN3 model fitting graph with epochs

4.2.2 Price and Volume encoded with GADF CNN5

Similar to section 4.2.1, Price and Volume features were encoded using GADF technique over a neighborhood of 20 minutes (10 mints to the left and 10 mints to the right) resulting each data point/candle to be of dimension (21, 21, 2). Figure 4.5 below shows the CNN 5 model architecture and Figure 4.8 below shows the averaged confusion matrix during 10-fold cross validation. Then the model was re trained with the full data set and saved as a .h5 file with the weights and parameters so that it can be loaded again to predict over the unseen data. Code used to generate GADF CNN5 is available at Appendix C. Table 4.2 below shows the averaged results from 10-fold cross validation with standard deviation.

Table 4.2 10-fold cross validation results of GADF CNN5 model

	Macro average	Standard deviation
Precision	98.82%	0.2116
Recall	98.78%	0.2117
F1-score	98.79%	0.2019

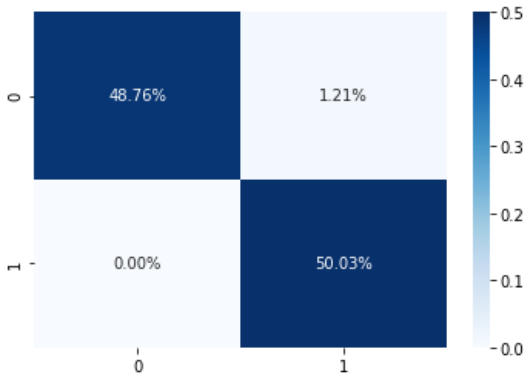


Figure 4.8 GADF CNN5 averaged confusion matrix

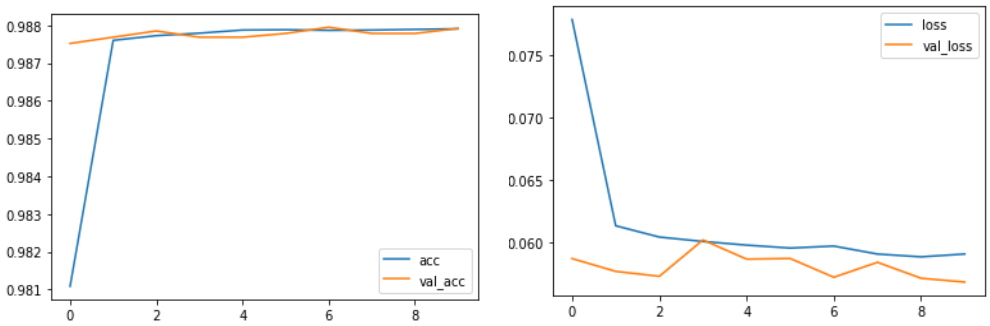


Figure 4.9 GADF CNN5 model fitting graph with epochs

4.2.3 Price and Volume encoded with GASF CNN5

Same CNN5 architecture was used with GASF encoding for Price and Volume features were encoded over a neighborhood of 20 minutes (10 mints to the left and 10 mints to the right) resulting each data point/candle to be of dimension (21, 21, 2). Figure 4.10 below shows the model fitting graph and Table 4.3 below shows the averaged results from 10-fold cross validation.

Table 4.3 10-fold cross validation results of GASF CNN5 model

	Macro average	Standard deviation
Precision	98.78%	0.1244
Recall	98.75%	0.1307
F1-score	98.75%	0.1307

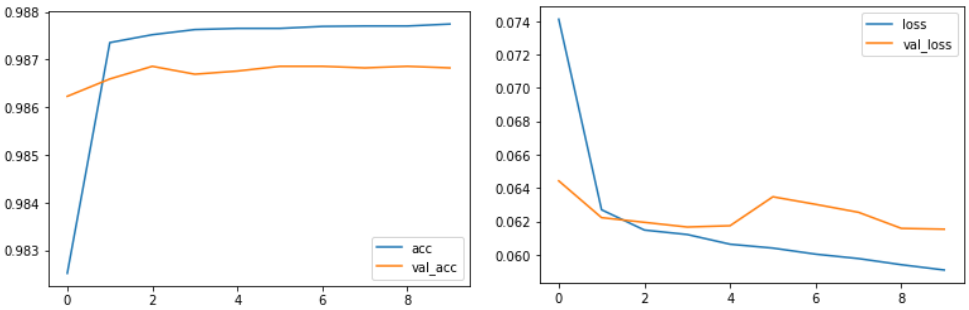


Figure 4.10 GASF CNN5 model fitting graph

4.2.4 Price and Volume encoded with MTF CNN3

Price and Volume features were encoded with Markov Transition field over a neighborhood of over a neighborhood of 20 minutes (10 mints to the left and 10 mints to the right) resulting each data point/candle to be encoded to dimension (21, 21, 2) and trained with CNN3 architecture given in Figure 4.6. Table 4.4 below shows the averaged results over 10-fold cross validation with standard deviation of results.

Table 4.4 10-fold cross validation results of MTF CNN3 model

	Macro average	Standard deviation
Precision	98.77%	0.1536
Recall	98.72%	0.1633
F1-score	98.73%	0.1624

Figure 4.11 below shows the model fitting graphs and can see it stabilizing at 10 epochs.

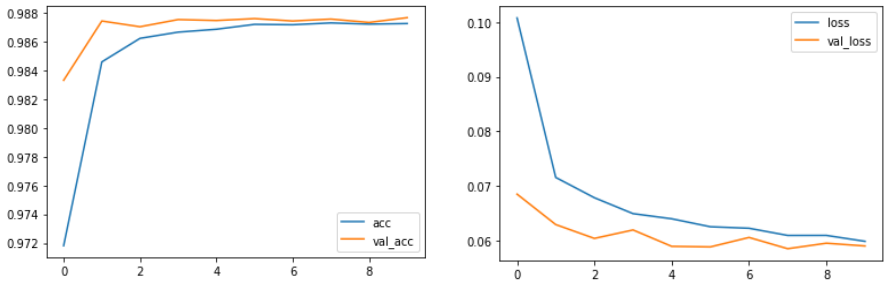


Figure 4.11 MTF CNN3 model fitting graphs

4.2.5 Price and Volume encoded with RP CNN3

Price and Volume features were encoded using Recurrence plot technique over a neighborhood of 20 minutes (10 mints to the left and 10 mints to the right) resulting each data point/candle to be encoded to dimension (21, 21, 2) and trained with CNN3 architecture given in Figure 4.6. Below Table 4.5 shows the averaged results from 10-fold cross validation and it can be seen that, standard deviation for these results were higher than that of the previous models.

Table 4.5 10-fold cross validation results of RP CNN3 model

	Macro average	Standard deviation
Precision	94.78%	4.5097
Recall	94.43%	5.3758
F1-score	94.42%	5.485

Figure 4.12 given below shows the model fitting graphs over the data set created in section 3.8.

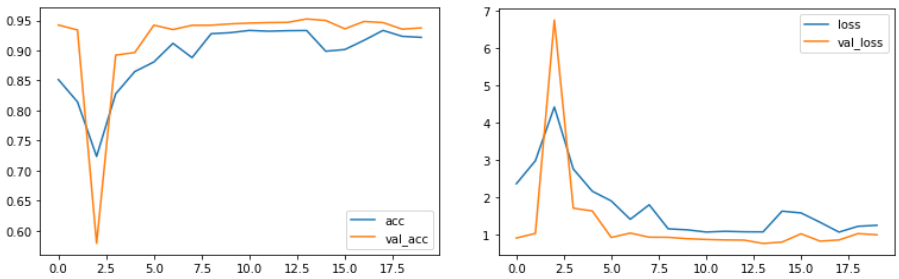


Figure 4.12 RP CNN3 model fitting graphs

4.3 Benchmark Experiments

In order to compare the model performance, the previous work by Josh Kamps and et al [1] was selected as the benchmark model since that was the only research that could be found in the literature review in the same context of pump and dump detection with open access to data and code. The same research has been referred in many other researches such as [14] and [12] as the ground truth for their results and model evaluations. However, there were few other research conducted on the same topic, having the labels taken by analyzing multiple public chat groups such as Discord and Telegram and accessing and analyzing exchange level data. However, such works

were not open to access with the code and data sets were not exposed to public for further development. This could be due to the restrictions in financial domain related research. As mentioned before, in this research, an initial labelling logic was used to annotate samples and a synthetic data set was created using SMOTE from expert reviewed cases as shown in section 3.8 for model training.

Suggested models in the work by Josh Kamps were re implemented using their public code on an aws Sagemaker notebook. In their work 3 set of parameters were suggested for detecting a pump and dump action.

1. Initial parameters
    a. Estimation window of 12 hours,
    b. 25% Volume increase
    c. 3% Price increase
2. Strict parameters
    a. Estimation window of 24 hours
    b. 400% Volume increase
    c. 10% Price increase
3. Balanced parameters
    a. Estimation window of 12 hours
    b. 300% Volume increase
    c. 5% Price increase

Above estimation window size was the width of the rolling window they used for the calculation and they suggested to detect a Price anomaly by calculating the moving average of Close Price over the estimation window of γ as shown in Figure 4.13.

$$\mu_\gamma(x) = \frac{\sum_{i=x-\gamma}^{x} x_{close}}{\gamma}$$

Figure 4.13 Rolling window average of close price

Sources: [1]

Given a Price threshold as €, a point was considered Price anomalous when the High Price for that point of time exceeds the value as shown in Figure 4.14.

$$price\_anomaly(x) = \begin{cases} True, x_{high} > \epsilon \cdot \mu(x) \\ False, x_{high} \leq \epsilon \cdot \mu(x) \end{cases}$$

Figure 4.14 Price anomaly detection

Sources: [1]

Similarly given a Volume threshold €, a point was considered Volume anomalous when the Volume exceeds it rolling window average as shown in Figure 4.15.

$$volume\_anomaly(x) = \begin{cases} True, x_{volume} > \epsilon \cdot \mu(x) \\ False, x_{volume} \leq \epsilon \cdot \mu(x) \end{cases}$$

Figure 4.15 Volume anomaly detection

Sources: [1]

According to their paper, a point was labelled as a pump and dump anomalous only if both Price anomaly and Volume anomaly are detected which was then followed either by a Price or Volume dip. Above three benchmark models with the 3 set of parameters were applied over the synthetic data set prepared under section 3.8 and their predictions were cross checked with the synthetic labels to calculate the results given in **RESULTS** section.

CHAPTER 5

RESULTS

5 RESULTS

Averaged results for the experiments with 10-fold stratified cross validations conducted under section 4 can be summarized as shown in Table 5.1 below along side with the standard deviation at each fold. Other than for RP CNN3 model, it could be seen that results were quite stable with a minimum standard deviation.

Table 5.1 CNN models 10-fold cross validation results summary

Model	Precision		Recall		F1	
GADF-CNN5	98.82%	±0.2019	98.78%	±0.2117	98.79%	±0.2116
GASF-CNN5	98.78%	±0.1244	98.75%	±0.1307	98.75%	±0.1307
GADF-CNN3	**98.86%**	**±0.1351**	**98.83%**	**±0.1414**	**98.83%**	**±0.1414**
MTF-CNN3	98.77%	±0.1536	98.72%	±0.1633	98.73%	±0.1624
RP-CNN3	94.78%	±4.5097	94.43%	±5.3758	94.42%	±5.485

Table 5.2 below has summarized the results of the prediction on the unseen data by GADF-CNN3, GADF-CNN5, MTF-CNN3, RP-CNN3 models explained under **Experiments** section while comparing the 3 benchmark models suggested by Josh Kamps at el in [1].

Table 5.2 Model evaluation

Classifier	Precision	Recall	F1
Kamps (initial)	65.23%	52.06%	53.54%
Kamps (balanced)	75.69%	53.36%	55.85%
Kamps (strict)	90.80%	55.21%	59.15%
GADF-CNN5	**99.07%**	**99.98%**	**99.52%**
GADF-CNN3	98.88%	99.97%	99.42%
MTF-CNN3	96.27%	99.95%	98.04%
RP-CNN3	53.90%	95.70%	55.75%

Results shown in Table 5.2 were based on the unseen data for the models and those data were also synthetically created as explained in section 3.8. Each file contained around 1 million data points for a coin in the unseen data files and the class distribution for each file looked like as shown in Figure 3.40. Therefore it can be seen that, results as significant as in Table 5.2 were taken from a highly imbalanced unseen data set from the models trained over a balanced data set shown as Figure 3.41. Best

results over the unseen data were given by GADF-CNN5 model with Precision Recall and F1 values in the scale of 99%.

CHAPTER 6

DISCUSSION

6 DISCUSSION

In this research, convolution neural networks were trained to classify anomalous pump and dump activities on crypto currencies by using a synthetic data set created by SMOTE technique on top of an expert annotated data set. The classical problem was originally in the time series classification domain. Owing to the reason that shape of the charts during a pump and dump was known to follow a specific pattern in Price and Volume features, the problem was converted and reimagined as an image classification problem by using temporal to spatial encoding techniques suggest over the recent research literature. Gramian angular difference field, Gramian angular summation field, Markov transition field and recurrence plots were experimented for the above spatial encoding and from the results, it can be seen that GADF, GASF and MTF encoding techniques have given significant results beating the state of the art models. To the best of our knowledge, this was the first of such attempts in the published literature to do pump and dump detection via a computer vision technique.

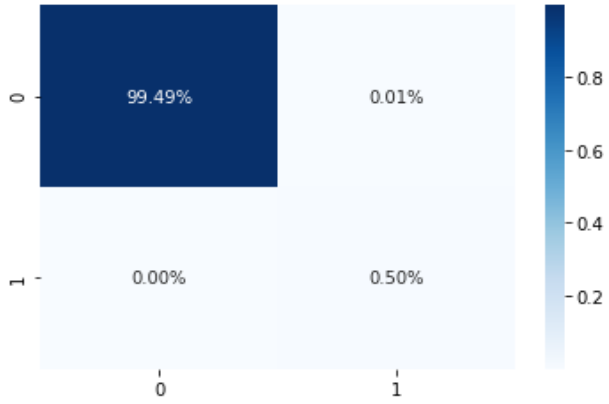


Figure 6.1 GADF CNN5 predictions confusion matrix

Figure 6.1 shown above was the confusion matrix for the predictions made by GADF-CNN5 model with the unseen data set which was heavily imbalanced. Other than for 0.01% of false positives, the GADF-CNN5 model has given no false negatives and has correctly classifies the cases with a F1 value of 99.52%. These results clearly showed the superiority of using an image classification technique to this problem of pump and dump classification over the conventional threshold-based techniques as suggested in previous work by Josh Kamps [1]. Having predefined parameters would

not give the best results when the pump and dump activities were carried out in different scales creating short term bubbles which were succeeding reversed in the coin's Price and Volume features during the course of actions. Encoding the time series as a 2D image has introduced different features to represent the temporal inter-dependencies of the timeseries values and enabled the models to search a smaller parameter space in 2D to recognize and classify the labels to achieve significant results over this particular task. Even the shallow CNN architectures as given in Figure 4.4 and Figure 4.5 have produced above significant results as high as 99.52% F1 value proving its applicability with a minimal computational power.

6.1 Contribution

In this research it was shown how a computer vision technique was used to solve a time series classification problem of pump and dump detection by using a novel approach of temporal spatial encoding using GAF, MTF to achieve significant results. The suggested CNN models have the advantages of not having to manually feature engineer and simple in architecture while computationally less complex as well. As mentioned previously, dearth of publicly available models with code and labelled data set were the major hindrance for this research. Therefore, all the code and data used in this research was made open to the public (https://github.com/aaivu/aaivu-pump-and-dump-detection-crypto-currencies.git) in order to conduct further research and improvements on this context and to extend its applicability to other domains as well.

CHAPTER 7

CONCLUSION

7 CONCLUSIONS

Given the results seen in section 5, it can be concluded that with the use of a temporal to spatial encoding technique such as Gramian Angular difference field, Gramian Angular summation field or Markov transition Field, to encode crypto OHLCV time series data in the form of images, in order to train simple convolutional neural network architectures like shown in Figure 4.4 and Figure 4.5, it can perform time series classification tasks such as Pump and Dump detection over crypto currencies brings about the state of the art results for precision, recall and f1 values. However, recurrence plot (RP) encoding was not shown to give good results for the problem of PND detection while GASF and GADF techniques had given equally comparable results. Given the convolutional neural network models were properly trained over a balanced data set, during the predictions above models can even handle imbalanced or noisy data. Because, by converting the original problem into image classification domain, models would eventually behave like how an expert would look at a chart and do the classification.

7.1 Future work

Further research on financial domain can be carried out to identify other types of market anomalies which are evident on the shape of the financial data by having encoded the discriminating time series features with the ground truth. Incorporating exchange level data such as, trade participant information, order size, trade side information and trade cancellations information as the inputs to the model training, could have given a broader view on trading dynamics of the exchange and it could have improved the true positive rate while reducing type-1 and type-2 error.

References

[1] B. K. Josh Kamps, "To the moon: defning and detecting cryptocurrency pump-and-dumps," *Crime Science,* pp. 1-18, 2018.

[2] T. B. M. V. Naftali Cohen, "Trading via Image Classification," 2019.

[3] A. Rosic, "What is Cryptocurrency? [Everything You Need To Know!]," BlockGeeks, [Online]. Available: https://blockgeeks.com/guides/what-is-cryptocurrency/.

[4] L. V. G. S. P. A. Pankaj Malhotra, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, Bruges, Belgium, 2015.

[5] Z. W. T. Oates, "Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks," in *29th AAAI Conference on Artificial Intelligence*, 2015.

[6] S. A.-E.-H. F. M. Mehrnoosh Mirtaheri, *Identifying and Analyzing Cryptocurrency Manipulations in Social Media,* IEEE, 2021.

[7] L. C. W. S. Hadi Mansourifar, *Hybrid Cryptocurrency Pump and Dump Detection,* Texas: arXiv.org, 2020.

[8] P. T. a. S. T. Teema Leangarun, "Stock Price Manipulation Detection Based on Mathematical Models," *International Journal of Trade, Economics and Finance,* Vols. Vol. 7, No. 3, June 2016.

[9] B. L. Jiahua Xu, "The Anatomy of a Cryptocurrency Pump-and-Dump Scheme," in *28th USENIX Security Symposium*, Santa Clara, CA, USA, 2019.

[10] M. Zaki, D. Diaz and B. Theodoulidis, "Financial Market Service Architectures: A “Pump and Dump” Case Study," in *2012 Annual SRII Global Conference*, San Jose, CA, USA, 2012.

[11] D. S. B. W. Tao Li, *Cryptocurrency Pump-and-Dump Schemes,* SSRN, 2018 Oct.

[12] L. C. W. S. Hadi Mansourifar, "Hybrid Cryptocurrency Pump and Dump Detection," *Computer Science Artificial Intelligence Cornell University,* 2020.

[13] F. V. a. T. Hagemann, "Cryptocurrency Pump and Dump Schemes: Quantification and Detection," in *IEEE, 2019.*, Beijing, China, 2019.

[14] A. M. F. S. J. S. Massimo La Morgia, "Pump and Dumps in the Bitcoin Era: Real Time Detection of Cryptocurrency Market Manipulations," in *ieeexplore.ieee.org*, 2020.

[15] F. Z. W. M. Ruinan Zhang, "Sequential Behavioral Data Processing Using Deep Learning and the Markov Transition Field in Online Fraud Detection," in *KDD, 2018, London, UK*, 2018.

[16] Y. G. a. J. D. Nima Hatami, "Classification of Time-Series Images Using Deep Convolutional Neural Networks," in *ICMV 2017*, France, 2017.

[17] Z.-X. C. a. C.-Y. Y. Chao-Lung Yang, "Sensor Classification Using Convolutional Neural Network by Encoding Multivariate Time Series as Two-Dimensional Colored Images," *Advances in Intelligent Single/Multiple Sensing Systems and Applications,* p. 15, 2019.

[18] Y.-C. T. Jun-Hao Chen, "ENCODING CANDLESTICKS AS IMAGES FOR PATTERN CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS," in *Financial Innovation, Springer*, 2020.

[19] P. C. a. J. B. Stan Salvador, "Learning States and Rules for Time Series Anomaly Detection," in *FLAIRS conference*, 2004.

[20] F. Victor and T. Hagemann, "Cryptocurrency Pump and Dump Schemes: Quantification and Detection," in *IEEE*, Beijing, China, 2019.

[21] P. S. Foundation, "pyts," pyts, 2021. [Online]. Available: https://pypi.org/project/pyts/.

[22] Binance, "python-binance," Binancae, 2017. [Online]. Available: https://python-binance.readthedocs.io/en/latest/overview.html.

[23] D. F. Silva, V. M. D. Souza and G. E. Batista, "Time Series Classification Using Compression Distance of Recurrence Plots," in *2013 IEEE 13th International Conference on Data Mining*, Dallas, TX, USA, 2013.

[24] K. W. B. L. O. H. W. P. K. N. V. Chawla, "SMOTE - Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. Volume 16, 2002.

[25] A. Z. K. Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ICLR 2015,* 2015.

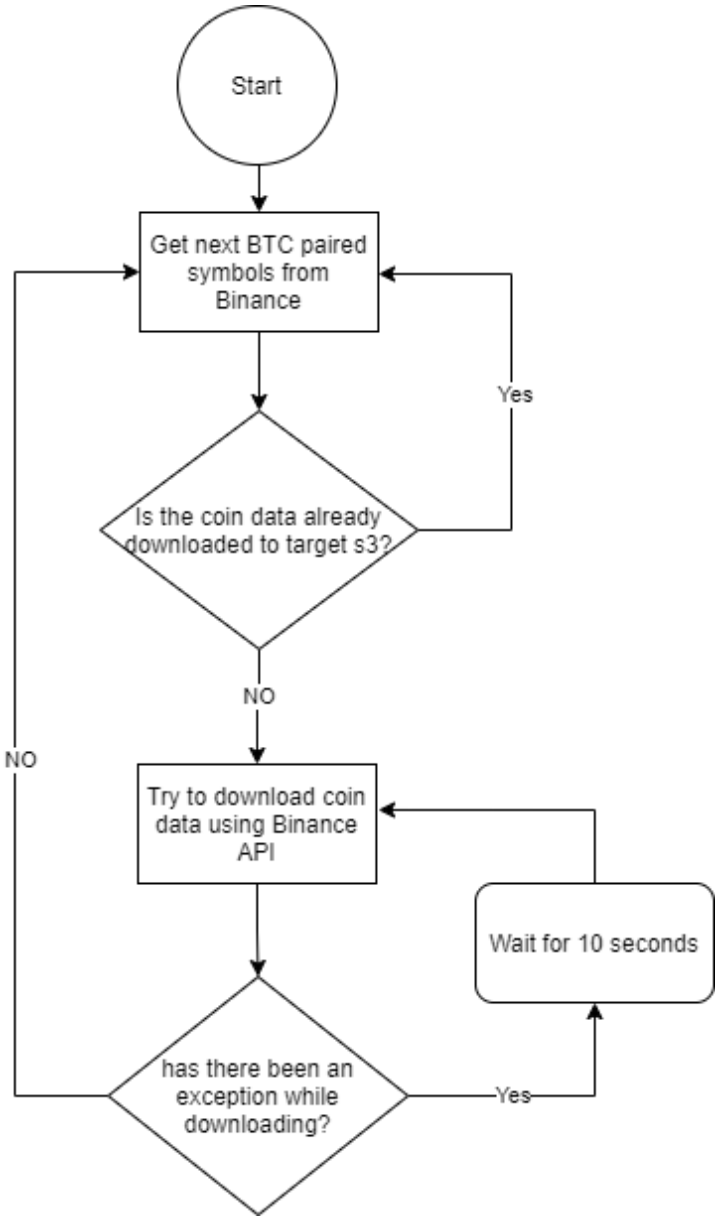
[26] Y. G. a. J. D. Nima Hatami, "Classification of Time-Series Images Using Deep Convolutional Neural Networks," in *Tenth International Conference on Machine Vision*, Vienna, Austria, 2017.

[27] X. Z. S. R. J. S. Kaiming He, "Deep Residual Learning for Image Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016*, 2016.

[28] M. C. R. M. T. J. K. Norbert Marwan∗, "Recurrence plots for the analysis of complex systems," in *Nonlinear Dynamics Group, Institute of Physics, University of Potsdam, Germany*, 2006.

Appendices

Appendix A Downloading data amidst of disconnections



Appendix B GADF CNN 3

```
def create_model():
    cnn=Sequential()
    cnn.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation='relu',
input_shape=(INPUT_MATRIX_WIDTH, INPUT_MATRIX_WIDTH, ENCODED_FEATURES)))
    cnn.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation='relu'))
    cnn.add(Dropout(0.25))
    cnn.add(Flatten())
    cnn.add(Dense(128, activation='relu'))
    cnn.add(Dropout(0.5))
    cnn.add(Dense(1, activation='sigmoid'))
    cnn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return cnn


kf = StratifiedKFold(n_splits=10)
history = []
confusions= []
classifReports= []
for train, test in kf.split(X_data, Y_data):
    cnn=create_model()
    x_train, x_test = X_data[train], X_data[test]
    y_train, y_test = Y_data[train], Y_data[test]
    hist = cnn.fit(x=x_train, y=y_train, validation_split=0.2, epochs=15, batch_size=500, verbose=0)
    history.append(hist)
    y_pred = cnn.predict(x_test)
    y_pred_R = np.round(y_pred)
    conf = confusion_matrix(y_test, y_pred_R)
    confusions.append(conf)
    clfr = classification_report(y_test, y_pred_R, output_dict=True)
    classifReports.append(clfr)
```

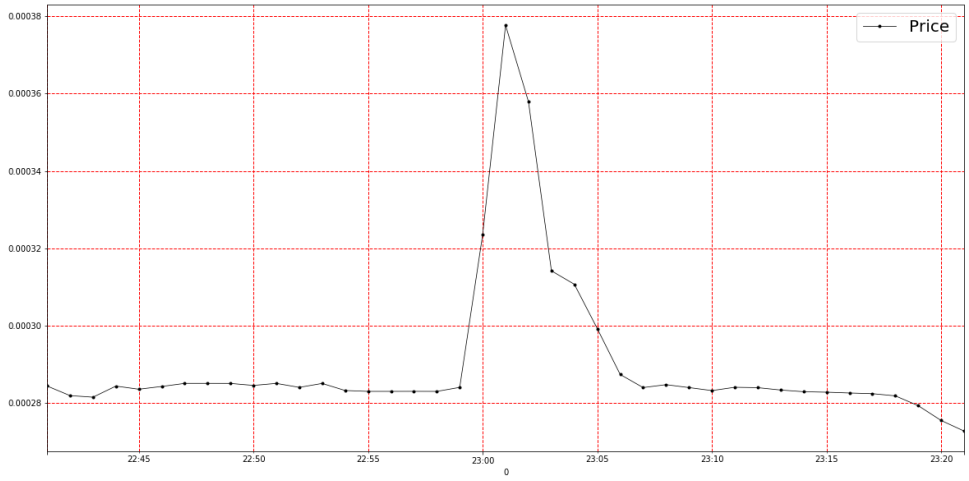
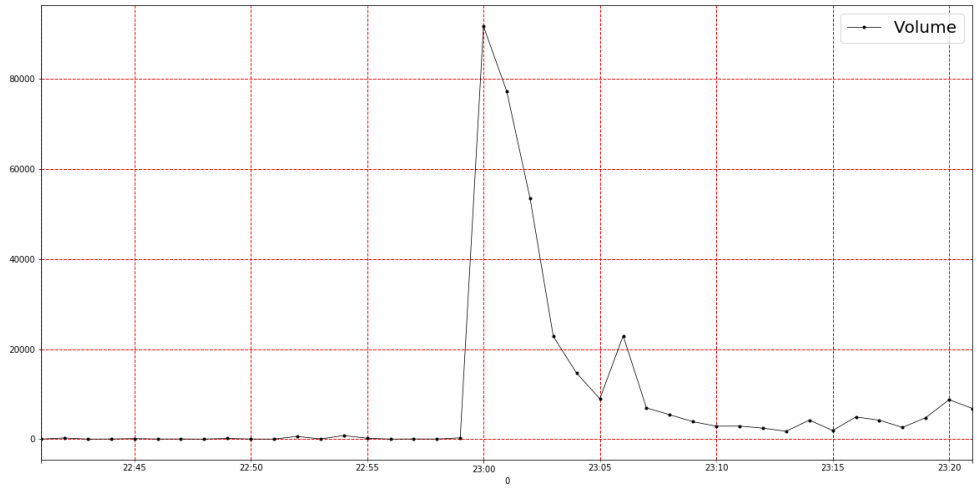
Appendix C GADF CNN 5

```
def create_gadfcnn5_model():
    cnn=Sequential()
    cnn.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation='relu',
input_shape=(21,21,2)))
    cnn.add(Conv2D(filters=64, kernel_size=(2,2), padding='same', activation='relu'))
    cnn.add(Dropout(0.25))
    cnn.add(Flatten())
    cnn.add(Dense(256, activation='relu'))
    cnn.add(Dropout(0.5))
    cnn.add(Dense(1, activation='sigmoid'))
    cnn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return cnn


kf = StratifiedKFold(n_splits=10)
history = []
confusions= []
classifReports= []
for train, test in kf.split(X_data, Y_data):
    cnn=create_gadfcnn5_model()
    x_train, x_test = X_data[train], X_data[test]
    y_train, y_test = Y_data[train], Y_data[test]
    hist = cnn.fit(x=x_train, y=y_train, validation_split=0.2, epochs=20, batch_size=500, verbose=0)
    history.append(hist)
    y_pred = cnn.predict(x_test)
    y_pred_R = np.round(y_pred)
    conf = confusion_matrix(y_test, y_pred_R)
    confusions.append(conf)
    clfr = classification_report(y_test, y_pred_R, output_dict=True)
    classifReports.append(clfr)
```

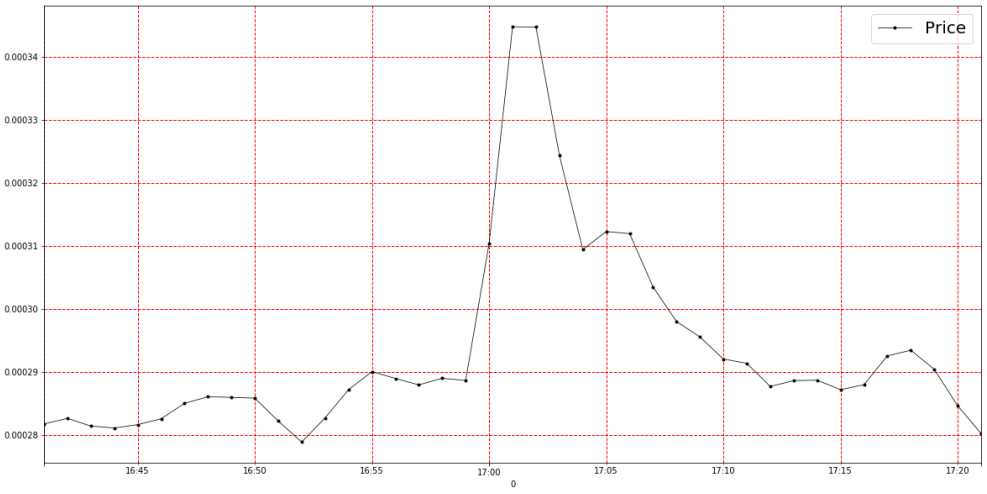
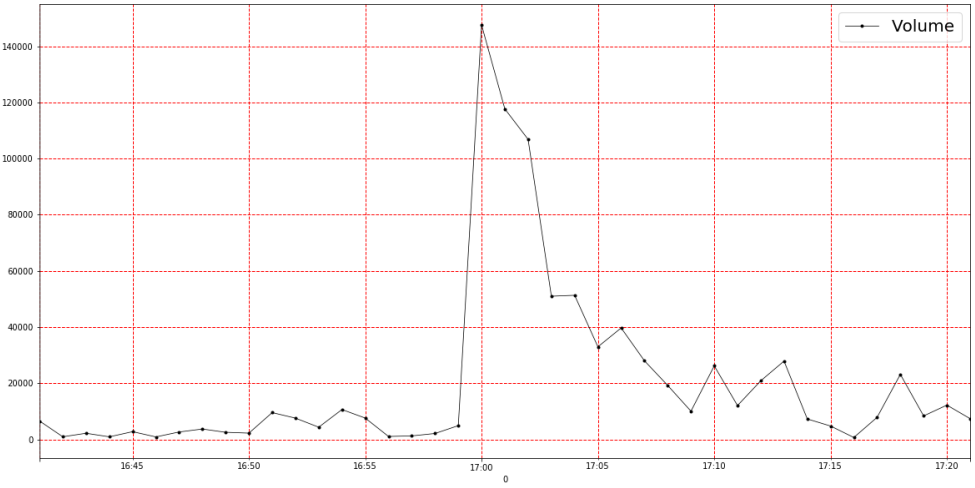
Appendix D Spatial to temporal encoding

```
def getGAFMatrix(df, feature, index, method='summation'):
        selectedWindow ← extract [index -10min, index +10min] window for the feature from df
        if selectedWindow.length() != DATA_POINTS_PER_WINDOW:
                return
        gaf = generateGramianAngulaField(window = timeseriesWindow, method = summation or difference)
        return gaf

def getMTFMatrix(df, feature, index, bins=10):
        selectedWindow ← extract [index -10min, index +10min] window for the feature from df
        if selectedWindow.length() != DATA_POINTS_PER_WINDOW:
                return
        mtf = generateMarkovTransitionField (window = selectedWindow, numberOfBins=bins)
        return mtf

def getRecurrencePlotMatrix(df, feature, index, threshold=None, span=10):
        selectedWindow ← extract [index -10min, index +10min] window for the feature from df
        if selectedWindow.length() != DATA_POINTS_PER_WINDOW:
                return
        rp = generateRecurrencePlot(threshold = threshold)
        return x_rp
```

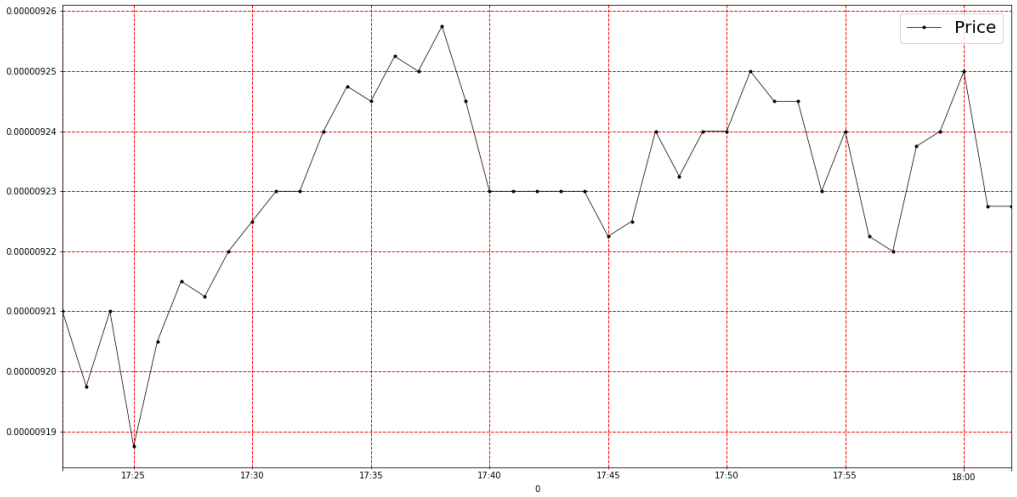
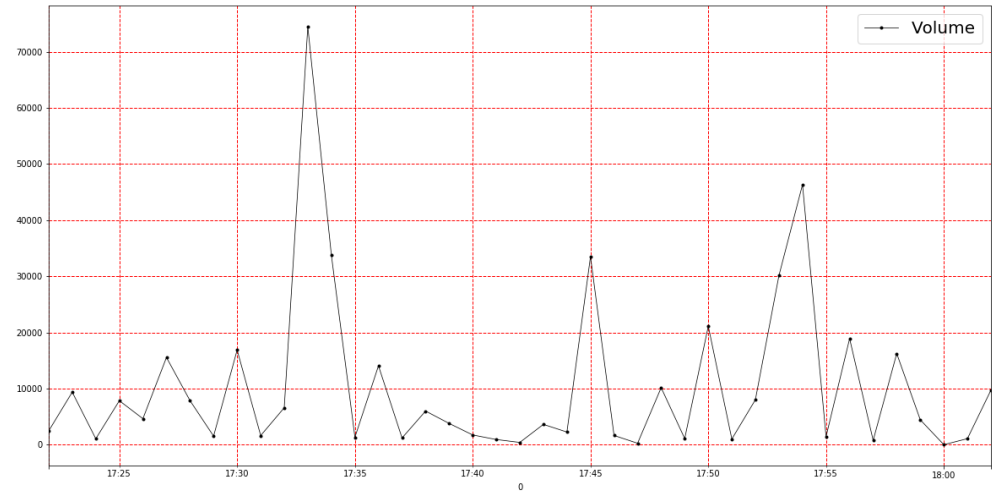
Appendix E ICNBTC expert comments and label

Coin and timestamp	ICNBTC: 2018-01-12_230100
Price chart	
Volume chart	
Numerical label	PND
Expert 1 Comment	PND - Traded Volume coincided with price move, likely to be as a result pump and dump type scenario. Social Media Ramping is a possibility, particular with Crypto in this case.
Expert 2 Comment	PND - Clear case of sudden large buy orders rapidly moving price higher

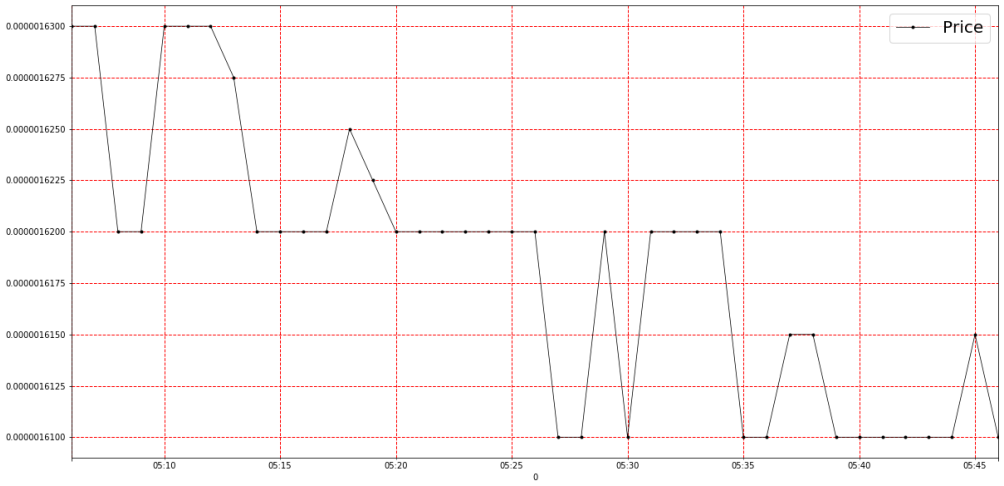
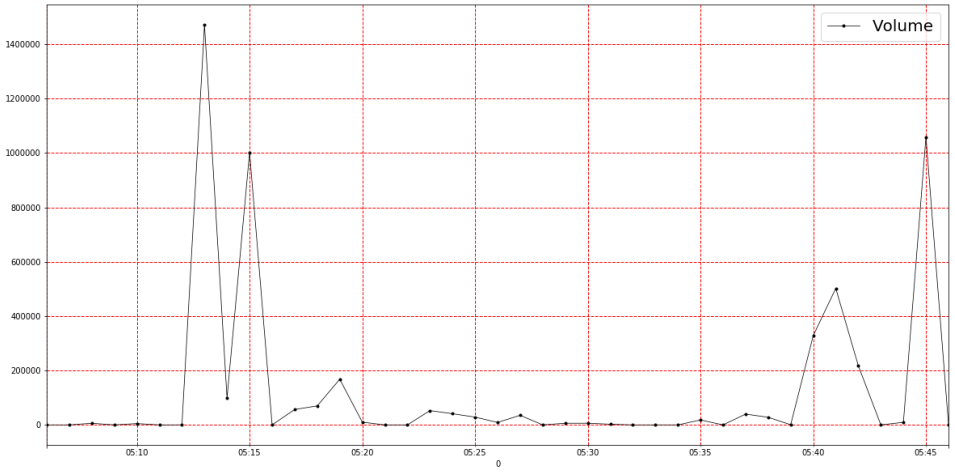
Appendix F MODBTC expert comments and label

Coin and timestamp	MODBTC: 2018-03-21_170100
Price chart	
Volume chart	
Numerical label	PND
Expert 1 Comment	Traded Volume coincided with price move, likely to be as a result of newsflow at 17:00 rather than pump and dump type scenario. Social Media Ramping is a possibility, particular with Crypto in this case. Given short timeframe, more likely to be Walking the Book up
Expert 2 Comment	PnD – Again, sudden surge of buy volume rapidly moving price higher.

Appendix G QKCBTC expert comments and label

Coin and timestamp	QKCBTC: 2019-01-27_174200
Price chart	
Volume chart	
Numerical label	Non-PND
Expert 1 Comment	No Pnd - Price rise was sustained across 30 minutes period, with volumes relatively stable, unlikely to be PnD
Expert 2 Comment	No PnD - The price moves before the volume spikes, suggesting traders were attracted by the move (as opposed to driving it)

Appendix H SCBTC expert comment and label

Coin and timestamp	SCBTC: 2018-06-29_052600
Price chart	
Volume chart	
Numerical label	Non-PND
Expert 1 Comment	No PnD - Price fall was sustained across 30 minutes period, with volumes relatively stable, unlikely to be PnD. Spike in volume at 5:13, 5:14 had little effect on Price
Expert 2 Comment	No PnD - No Correlation at all between the price and volume moves in this case