

UNIVERSITY OF MORATUWA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CS 4202 – RESEARCH & DEVELOPMENT PROJECT
FINAL YEAR PROJECT REPORT
MANIPULATION DETECTION IN STOCK TRADING USING
MACHINE LEARNING

PROJECT GROUP – STACKSCOUT (GROUP 05)

A.M.N.V Chandrasoma (130078X)

H.Y Mudalige (130380P)

K.M.S.A Munasinghe (130383D)

B.V Vithanage (130620E)

Internal Supervisor

Dr. H. M. N. Dilum Bandara

External Supervisors

Dr. Rasika Withanawasam

Dr. Ashoka Koralage

Coordinated By

Dr. Charith Chitraranjan

THIS REPORT IS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE OF
ENGINEERING AT UNIVERSITY OF MORATUWA, SRI LANKA

December 20, 2017

Declaration

We, the project group StackScouts (Group 5) hereby declare that except where specified reference is made to the work of others, the project “Manipulation Detection in Stock Trading” is our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgement.

Signatures of the Candidates

1. A.M.N.V Chandrasoma (130078X)
2. H.Y Mudalige(130380P)
3. K.M.S.A Munasinghe(130383D)
4. B.V Vithanage(130620E)

Supervisor:

.....

(Signature & Date)

Dr. H.M.N. Dilum Bandara

Project Coordinator:

.....

(Signature & Date)

Dr. Charith Chitraranjan

External Supervisor:

.....

(Signature & Date)

Dr. Rasika Withanawasam

Abstract

Stock traders should respond to a variety of regulations that are meant to ensure proper functioning of the stock market. However, there are some agents who are eager to violate the trust of the investors to gain personal benefits. While the stock trading platform is expected to detect such manipulations, it is a nontrivial task given the volume of data, velocity at which they arrive, large number of buyers and sellers, and complexity of manipulations. However, it is sufficient to detect such manipulations at the end of the day, before the orders are settled.

A couple of supervised machine-learning and rule-based techniques are proposed in related work to detect anomalous trading behaviors. However, such solutions are tight to a specific trading platform, can detect only the known manipulations, and relay on large volume of labelled data. We propose an unsupervised methodology to overcome some of these limitations while focusing on detecting manipulation time or order frames. We focus on Momentum Ignition based manipulations using a real dataset from one of the world's largest trading platforms. The proposed techniques are based on information entropy, clustering, and visualization. Proposed solution is effective because it does not need labeled data for the analysis. Moreover, we developed a tool which could be used by surveillance teams to analyze potential manipulations while simulating the orderbook. The proposed solution was able to gain a F-measure of 78% based on an analysis conducted using a set of real and synthetic stock market datasets.

Acknowledgement

First and foremost, we would like to express our sincere gratitude to our project supervisor, Dr. H.M.N. Dilum Bandara for the invaluable guidance and dedicated involvement at every step throughout the process.

We would also like to thank our external supervisors Dr. Rasika Withanawasam and Dr. Ashoka Koralage for the valuable advices and the direction given to us regarding the project.

Apart from that we would like to thank the Surveillance team in MilleniumIT, for the continuous support, giving us the feedbacks and valuable KT sessions regarding the stock market domain.

We would like to express our warm gratitude to Dr. Charith Chitraranjan for coordinating the final year projects.

Finally, we would like to express our greatest gratitude to the Department of Computer Science and Engineering, University of Moratuwa for providing the support for us to successfully finish the project.

Table of Contents

1. Introduction.....	1
1.1. Background	1
1.2. Motivation	1
1.3. Problem Description.....	2
1.3.1. Problem Statement	2
1.3.2. Objectives	2
1.3.3. Contribution	2
1.4. Outline.....	3
2. Literature Review.....	4
2.1. Stock Market Manipulations	4
2.1.1. Pump-and-dump.....	5
2.1.2. Layering	5
2.1.3. Momentum Ignition	6
2.2. Stock Market Trading Behavior.....	7
2.2.1. Trading types	7
2.2.2. Order Book.....	8
2.2.3. Message Types.....	9
2.2.4. Auctions	10
2.3. Supervised Detection.....	11
2.3.1. One Class Support Vector Machine with Hidden Markov Model.....	11
2.3.2. Trading Networks	19
2.4. Unsupervised Detection	20
2.4.1. Time Series Contextual Anomaly Detection	20
2.4.2. Mathematical Model	21
2.4.3. Clustering.....	24
3. Design	27
3.1. Workflow Diagram	27
3.2. Layered Architecture.....	28
3.2.1. Presentation Layer	28

3.2.2.	Pre-Processing Layer	28
3.2.3.	Processing Layer	30
3.2.4.	Data Access Layer	33
4.	Implementation	35
4.1.	Tools and Technologies	35
4.2.	Preprocessing Module	36
4.3.	Processing Module	39
4.3.1.	Clustering Module	39
4.4.	Dashboard Implementation	42
4.4.1.	Session File Uploader	42
4.4.2.	Trading Data File Uploader	44
4.4.3.	Window Size Chooser.....	45
4.4.4.	Entropy Display	47
4.4.5.	Price Gap Variation Display	47
4.4.6.	Clustering Display	48
4.4.7.	Orderbook Simulation Display	49
4.4.8.	Frame Summary Display	49
5.	Performance Analysis	52
5.1.	Data Preparation	52
5.2.	Performance Measurement.....	53
6.	Summary	57
6.1.	Conclusion.....	57
6.2.	Challenges	57
6.3.	Future Work	58
7.	References.....	59

List of Figures

Figure 2.1 - Momentum Ignition scenario.	7
Figure 2.2 - Layout of an order book.	8
Figure 2.3 - Order book message.	10
Figure 2.4 - Auction time orderbook behavior.	11
Figure 2.5 - Single order disruptive trading.	12
Figure 2.6 - Multi order disruptive trading triggers the bid price to increase quickly and drop also quickly after the spoofing orders are cancelled.	12
Figure 2.7 - Detecting system.	13
Figure 2.8 - Support vector machine to detect normal and abnormal trading.	14
Figure 2.9 - Joint PDF of price vector component and volume vector component	16
Figure 2.10 - SMOTERUS approach for MSFT data.	16
Figure 2.11 - (a) Example states of observed trading behaviors (b) structure of the designed HMM with disruptive states	17
Figure 2.12 - Graphical illustration of abnormal trading motifs (A)Self-loop (B)Two-node loop (C)two-node multiple arcs	19
Figure 2.13 - Price variation in a Pump and Dump scenario	22
Figure 2.14 - Spoofing trading example	23
Figure 2.15 - Different cluster boundaries in same set of clusters	24
Figure 3.1 - Workflow diagram.	27
Figure 3.2 - High level architecture design.	28
Figure 3.3 - Anomaly Score.	31
Figure 3.4 - Elbow method to generate dynamic K value.	32
Figure 3.5 - Information entropy of execution types.	33
Figure 4.1 - The class diagram of the system.	35
Figure 4.2 - Window design.	36
Figure 4.3 - Orderbook design.	37
Figure 4.4 - Buy orders price points.	38
Figure 4.5 - Sell orders price points.	38
Figure 4.6 - Dictionary structure of order details.	38
Figure 4.7 - Price gap window without manipulations.	41
Figure 4.8 - Price gap window with a manipulation.	42
Figure 4.9 - Session file uploader.	43

Figure 4.10 - Session details displayed by the tool.....	43
Figure 4.11 - Trading data file uploader.	44
Figure 4.12 - Trading data statistics shown in the tool.	44
Figure 4.13 - Window size input fields and other options.	45
Figure 4.14 - Tool after the end of pre-processing.	46
Figure 4.15 - Entropy variation displayed in the tool.	47
Figure 4.16 - Price gap variation display in the tool.	48
Figure 4.17 - Clustering result visualization.	48
Figure 4.18 - Orderbook Simulation.	49
Figure 4.19 - Frame summary.	50
Figure 4.20 - Broker summary.	50
Figure 4.21 - After generating the processing result.	51
Figure 5.1 - Synthetic Data Creation.	53
Figure 5.2 - Entropy variation of the selected dataset.	55
Figure 5.3 - Price gap variation of a normal time frame (Time frame 4).	55
Figure 5.4 - Price gap variation of the manipulated time frame (Time frame 19).	56

List of Tables

Table 4.1 - Session table details.....	43
Table 5.1 - Tool performance analysis.	53
Table 5.2 - Confusion matrix for Entropy.	54
Table 5.3 - Confusion matrix for Clustering.....	54
Table 5.4 - Accuracy measurements.	54

List of Abbreviations

ADIST	Anomaly Detection in Stock Trading
API	Application Programming Interface
CAD	Contextual Anomaly Detection
FP	False Positive
FN	False Negative
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HMMAS	Hidden Markov Model with Abnormal States
IDE	Integrated Developing Environment
MFST	Medifirst Stock Trading
OCSVM	One Class Support Vector Machine
PSO	Particle Swarm Optimization
PDF	Probability Density Function
REST	Representational State Transfer
RSS	Rich Site Summary
RUS	Random Under Sampling
SMOTE	Synthetic. Minority Oversampling Technique
SOM	Self Organizing Map
SSE	Sum of Square Error
TN	True Negative
TP	True Positive
VWAL	Volume Weighted Average Lifecycle

1. Introduction

1.1. Background

In the stock market domain, the intention of doing trades is to create profit out of the stock trades. Traders usually refer to the current and historical data of the involving parties before doing a stock trade. Hence, the look or the behavior of the stock matters in this context. One can manipulate the stock market using false trading where he/she can trade to change stock market attributes without targeting actual profit for those trades. Stock market manipulation refers to the activities of those traders who use carefully-designed trading behaviors to push up or down the underlying equity prices for making profits. Manipulations can be categorized as action based (e.g., not bidding in an auction and closing down a plant), information based (e.g., spreading false rumors and news), and trade based (e.g., performing a series of trades to mislead the market). With increasing volumes and frequency of trading, price manipulation can be extremely damaging to the proper functioning and integrity of capital markets. If every trader gets all the trading related information immediately and simultaneously, people who win and lose will be selected randomly and that is the expected behavior of a real stock market.

Changing the stock market behavior because of a twitter message can be given as an example for unstructured data manipulation [1]. Apart from that traders are doing many disruptive trading in stock market using structured data. Pump and dump, insider trading, Momentum Ignition and Spoofing Corner are some examples [2]. These names have originated from the starting behavior of each manipulation.

1.2. Motivation

Currently trading platforms use a set of rules to check whether a trading is a disruptive or not. Example rules include the following:

1. If a trader pushes the price up with small trades for large number of times within a short period, trigger an alert.
2. If a trader pushes the price up with small trades for large number of times within a short period, and if he does that number of times within the day, trigger an alert.
3. If a trader cancels a significant number of orders within a short period of time, and if he does that number of times within the day, trigger an alert.

There are supervised machine learning techniques such as [3] to detect manipulations in stock tradings. Both these techniques cannot identify new disruptive trading behaviors that have not

occurred earlier. Therefore, by using a system which can detect both observed and unobserved manipulation patterns will gain a more advantage for users. As this is a manual process the need of human involvement is too much. Surveillance team has to get different threshold values for them to differentiate for anomaly behavior and normal behavior. Different types of clients are trading in stock market. Client's trust on the trading platform is must needed.

1.3.Problem Description

1.3.1. Problem Statement

The problem that this project plans to address can be formulated as follows:

Automatically detect and analyze fraudulent behaviors in stock trading

Momentum ignition is a scenario which tries to instigate other participants to buy or sell quickly, the instigator of momentum ignition can profit either having taken a pre-position or by laddering the book, knowing the price is likely to revert after the initial rapid price move, and trading out afterwards. Since it tries to change the price movement in trading, we focus on that manipulation pattern in stock trading manipulation detection in a unsupervised approach because it is needed to handle large volume of data and visualizing them too.

Moreover, it should be easier to integrate new modules to detect other manipulation patterns in stock trading.

1.3.2. Objectives

The objectives of this project can be stated as follows:

1. To conduct a comprehensive literature survey to understand common manipulation patterns in stock market.
2. Identify suitable machine learning algorithms.
3. Design and implement an automated system to detect many disruptive time frames or order frames which have occurred in a stock market data.
4. Issue alerts when disruptive trading take place.
5. Analyze the performance of the proposed solution based on throughput and detection accuracy.

1.3.3. Contribution

This project makes the following contributions:

1. Identified different manipulation detection in stock trading in literature survey.

2. Applied suitable machine learning techniques and visualization techniques to detect manipulation.
3. Developed an automated tool to detect manipulated time/order frames with visually.

Our approach is an automated system which only rely on stock market end of the day data. Data will be fetched into different machine learning techniques and identify which are the time frames with manipulative behavior. Apart from that, this tool supports visualization techniques to further analyze and identify whether the given frame can be a fraudulent time frame or not.

1.4.Outline

Rest of the report is organized as follows. Chapter 2 presents the literature review on stock market manipulations, difference between the normal and manipulated behavior, types of stock market manipulations, and detection techniques. Chapter 3 presents the proposed design including the layered architecture of the proposed tool. Implementation details are presented in Chapter 4. Chapter 5 presents concluding remarks and future work.

2. Literature Review

In this chapter we analyze the related work. Understanding stock market manipulation has a significant value in our project. Section 2.1 presents a detailed description about manipulation types and their behavior. Section 2.2 will describe trading behavior of the stock market which explains the message type of order, orderbook behavior and trading types and it is essential to do because it helps to identify different features. Section 2.3 presents about supervised way of detecting stock market manipulations followed with Section 2.4 which describes unsupervised way of detecting manipulation in stock trading.

2.1. Stock Market Manipulations

Allen and Gale [4] classify stock market manipulations into three main categories as follows:

1. Action-based Manipulation – In these sort of manipulation, manipulator abuse resources so as to influence the estimation of those advantages, or the costs of their yields or sources of info. For example, American Steel and Wire Company's directors short sold the company's stock, at that point reported a shutdown of its plants which caused a vast decrease in the stock cost. They at that point secured their short positions at the low cost, and re-opened the plants. Another illustration could be a power generator that pronounces a plant blackout keeping in mind the end goal to drive up the cost of power so as to expand the payout on a power subordinates contract.
2. Information-based Manipulation - This involves the release of false or misleading information that causes prices to change in a way that benefits the financial position of the fabricator. Manipulator uses social medias like twitter, Facebook, RSS feeds and many other forms of communication to send false or misleading announcements to the public very easily.
3. Trade-based Manipulation - In this case, the manipulator buys or sells in quantity, knowing that due to asymmetric information and trade processing and inventory costs prices will move in the direction of his trades.

Detection of manipulation is a complex problem because,

1. One or more manipulators can be involved for a manipulation.
2. Manipulators use different features to do the manipulation.
3. High frequency trading platform.
4. Manipulators frequently change their patterns to do manipulations.

However, at the end of the day if there is a manipulation, it should happen after a trade. Moreover, this is the only form of manipulation that could be initiated within the trading platform. Therefore, we can keep our focus only on trade-based manipulations. Trade-based manipulation can be further classified based on their manipulation patterns. Next, some of the popular ones are discussed next.

2.1.1. Pump-and-dump

Pump-and-dump schemes involve touting a company's stock through false or misleading statements in the marketplace to artificially inflate (pump) the price of a stock. Once fraudsters stop hyping the stock and sell their shares (dump), the price typically falls. Although pump-and-dump schemes have existed for many decades, the emergence of the Internet and social media has provided a fertile new ground for fraudsters. False or misleading information can now be disseminated to many potential investors with minimum effort, anonymously, and at a relatively low cost.

2.1.2. Layering

In this type of manipulations, securities traders try to manipulate the price of a stock ahead of transactions that they wish to execute, creating more advantageous executions for themselves. It is a variety of a stratagem that has come to be called spoofing, itself an element of high-frequency trading.

Through layering, a trader tries to fool other traders and investors into thinking that significant buying or selling pressure is mounting on a given security, with the intent of causing its price to rise or fall. The trader does this by entering multiple orders that he has no intention of executing but instead plans to cancel.

Buying Example

A trader is looking to buy 1,000 shares of ABC stock, which is trading at LKR 2,000/- per share. In hopes of pushing its price down, he enters 4 large orders to sell:

1. 10,000 shares at LKR 2,005/- per share
2. 10,000 shares at LKR 2,010/- per share
3. 10,000 shares at LKR 2,015/- per share
4. 10,000 shares at LKR 2,020/- per share

Note that the trader has layered these sell orders at incrementally higher prices above the current market price. Thus, they will not execute unless the current market price moves upward.

The trader intends to make other market participants believe that selling pressure is mounting among holders of ABC stock and that the price thus is bound to fall below LKR 2,000/-per share.

If the scheme works, other traders eager to sell will enter orders below LKR 2,000/, anticipating that those orders to sell 40,000 shares soon will be re-entered at even lower prices.

The trader then will be able to purchase 1,000 shares of ABC at less than LKR 2,000/- per share and cancel those layered sell orders.

The trader runs a risk that orders to buy ABC will intervene, instead pushing the price above LKR 2,000/- per share. In this case, the trader will have to deliver up to 40,000 shares to buyers, shares that he might have to obtain at yet higher prices, incurring a large loss in the process.

Selling Example

A trader looking to sell 1,000 shares of ABC stock would do the opposite, to push its price up. He would enter 4 large orders to buy:

1. 10,000 shares at LKR 1,995/- per share
2. 10,000 shares at LKR 1,990/- per share
3. 10,000 shares at LKR 1,985/- per share
4. 10,000 shares at LKR 1,980/- per share

If the strategy works, people eager to buy will enter orders above LKR 2,000/- per share, expecting that the layered orders (which they do not know to be a mere ruse) will be re-entered at yet higher prices. The trader will be able to sell at over LKR 2,000/- per share and cancel those buy orders. Once again, there is a risk. Genuine orders to sell may intervene at less than LKR 2,000/- per share, forcing the trader to buy shares that he did not want, as those buy orders get executed.

2.1.3. Momentum Ignition

This scenario happens when manipulators attempts to trigger many other participants to trade quickly and cause a rapid price move. By trying to trick other participants to buy or sell quickly, the manipulator of momentum ignition can profit either having taken a pre-position or by laddering the book, knowing the price is likely to revert after the initial rapid price move, and trading out afterwards.

Momentum ignition does not happen in a matter of moments, but rather its culprits advantage from an ultra-quick response time. By and large, the manipulator takes a pre-position, prompts other traders in the market to exchange forcefully accordingly, causing a rapid price move, at that point manipulator trades out.

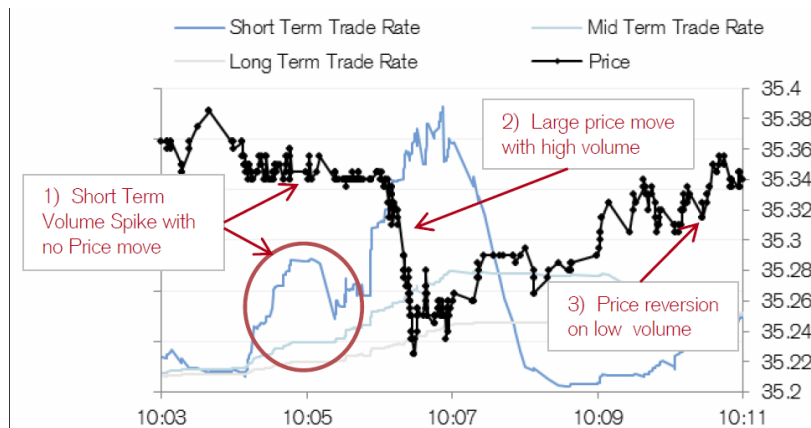


Figure 2.1 - Momentum Ignition scenario [5].

As shown in Figure 2.1, when we follow the blue line it shows a rapid growth in the center region. That means the trade rate was high in that region. Because of that we can see that the price has gone down rapidly (shown in dark black line), that is where the manipulator waits for the opportunity and make some profit out of it.

2.2. Stock Market Trading Behavior

An exchange is a market in which securities, commodities, derivatives and other financial instruments are being traded. The main function of an exchange is to make sure the fairness and orderly trading. As well as efficient dissemination of price information for any securities trading on that exchange. Exchanges give companies, governments and other groups a platform to sell securities to the investing public. The main purpose of surveillance team of a stock exchange is to catch any manipulative behaviour which make harm to the fairness of trading in a stock market.

2.2.1. Trading types

1. On book – sellers and buyers put their orders in the order book and those orders got matched according to the matching engine.
2. Off book – changing the ownership of equities by discussing among parties and then inform the exchange about the agreement. (these trades also get visible through orderbook as a market watch ticker)

To keep the fairness of the stock market they always keep track of the prices of off book trades. But for our project first we are not considering those off-book prices. Also, there cannot be an off-book trade happens without letting know the Exchange. In the starting phase, we are only considering about the on-book trades.

2.2.2. Order Book

An order book is an electronic list of buy and sell orders for a specific security or financial instrument(equity), organized by price points. The order book lists the details of shares being bid or offered at each price points. It also identifies the traders behind the buy and sell orders, although some choose to remain anonymous. The order book is dynamic and continuously updated in real time throughout the day. Exchanges such as Nasdaq refer to this order book as the “continuous book.” Orders that specify execution only at market open or market are maintained separately. These are known as the “opening (order) book” and “closing (order) book,” respectively. Following figure shows an example for a simple orderbook.

Broker ID	Bid Volume	Bid	Ask	Ask Volume	Broker ID
12	100	\$95	\$100	50	23
14	50	\$90	\$110	100	11

Figure 2.2 - Layout of an order book.

1. First three columns of the order book are to represent buy orders. Those columns are sorted as the maximum bid comes to the top. For differently priced orders it gives a price priority. For the orders with the same price order book gives the price, time priority, so that the order which put first given the priority to go up in the order.
2. Next part of the order book represents the sell orders. Those columns are sorted as the minimum ask comes to the top.
3. There is a separate order book for every equity(Instrument).

The order book information helps traders take better-informed trading decisions, since they can see which brokerages are buying or selling the stock and whether market action is being driven by retail investors or institutions. The order book also shows order imbalances, which may provide clues to the stock’s direction in the very short term. A massive imbalance of buy orders compared to sell orders, for instance, may shows a move higher in the stock due to buying pressure.

Aggressing orders – When an order comes to the book if there is an existing order in the book that can be matched with the new order that new order is called as an aggressing order. Those orders are not visible through the order book since they did not get to the book.

Passive orders – When an order comes to the book if there are no existing orders to get match with the new order, that new order is called as a passive order. They wait in the order book until it gets matched or expired.

Limit orders - A limit order is an order placed with the intension to take a profit by a bank or brokerage to buy or sell a set amount of a financial instrument at a specified price. because a limit order is not a market order. it may not be executed if the price set by the investor cannot be met during the period in which the order is left open. Trader specifies the price and the volume that he need that order to be. So that order will only get executed according to that values. Limit orders also allow an investor to limit the length of time an order can be outstanding before being canceled.

Market orders - An investor makes a market order through a broker or brokerage service to buy or sell an investment immediately at the best available current price. A market order is the default option and is likely to be executed because it does not contain restrictions on the price or the time frame in which the order can be executed. A market order is also sometimes referred to as an unrestricted order.

2.2.3. Message Types

There are various types of messages send by the exchange for the events happening in the exchange. All the parties interested in the stock market can use this message information to get an idea about what is going on in the stock markets. for a specific trader or a broker, stock market gives an access level. So, according to that the amount of detail which is visible for a specific user may vary.

Order Messages – A message sent these types of message with every order that puts to the orderbook. These messages consist of,

- Order quantity
- Execution type – new, amend, fill, cancel
- Order ID – unique ID with mix of strings and number.
- Broker ID – Broker firm who has the direct access to the exchange.
- Trader ID

- Client ID
- Client account ID
- Executed quantity
- Executed volume
- Tiff – Good till cancel, Fill or kill

Trades Messages - A message sent these types of message with every trade that happens. These messages consist of,

- Trade ID
- Buy/sell broker's ID
- Buy/ sell client's ID
- Trade status – New, Cancel, amend (two parties should agree)
- Tick – Deviation from the top of the book
- Market watch Messages
 - Best bid
 - Best offer
 - Total traded volume
 - Open/close price

Time	Type	Order ID	Size	Price	Direction
34200.025	1	16120456	18	5859100	-1

Figure 2.3 - Order book message.

2.2.4. Auctions

In trading, an auction refers to the process by which the prices of shares are determined before the open, after the close, or during intraday volatility auctions to build or stabilize the order book. They allow traders to place market or limit orders directly on an exchange.

The set price of a share on an exchange is represented as the highest amount that a bidder is willing to pay for it and the lowest amount that a seller is willing to take for it. Because there are several competing bidders and sellers, there are several offers and asking prices.

Liquidity is concentrated during auctions to maximize the volume and minimize the surplus left by unmatched orders. Orders may be entered, modified and cancelled during an auction period but no automated execution occurs. The indicative auction price and uncrossing volume will be updated whenever new orders are created, amended or deleted: resulting in a new auction price and volume.

Opening Auction - At the start of the day (8.15 A.M. - 8.30 A.M.)

- Chooses the opening price of and equity
- Auction prices are not visible to anyone (Indication price is only visible not the book)
- Using the orders buyers and sellers can affect the opening price (Matching Logic based on quantity and prices)
- Orders that on the opening price are executed as a bulk at the end of the auction. No other orders get executes.

Closing Auction - At the end of day (4.15 P.M.-4.30 P.M.)

- Chooses the closing price of and equity

Broker ID	Bid Volume	Bid	Ask	Ask Volume	Broker ID
12	100	\$95	\$100	50	23
14	50	\$90	\$110	100	11

Figure 2.4 - Auction time orderbook behavior.

2.3.Supervised Detection

2.3.1. One Class Support Vector Machine with Hidden Markov Model

This is a model based on both OCSVM (One Class Support Vector Machine) and HMM (Hidden Markov Model) [7]. This hybrid model analyzes single-order trades and multi-order trades in two different stages. The OCSVM detects the single order manipulations and the HMM detects the multi order manipulations. Single-order manipulations are those which occur at a certain point in time. That addresses the manipulation that happens using a single order. An example is placing a large sized order at a longer period for creating a special attention to it.

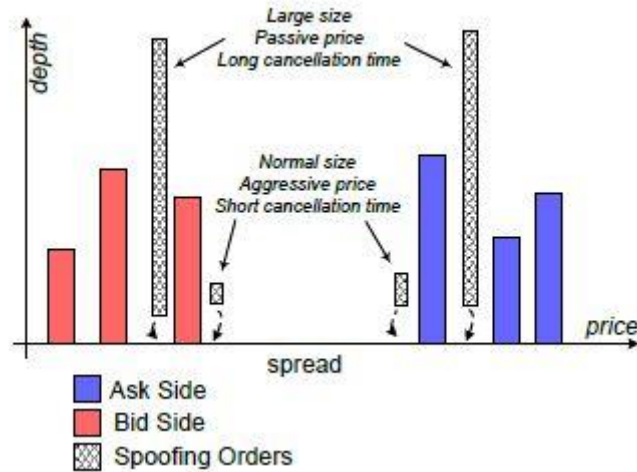


Figure 2.5 - Single order disruptive trading[7].

Multi-order manipulations are related to several other trades in several points in time. These are as an input sequence with no intention of execution. Such order sequences normally have successively increasing or decreasing prices.

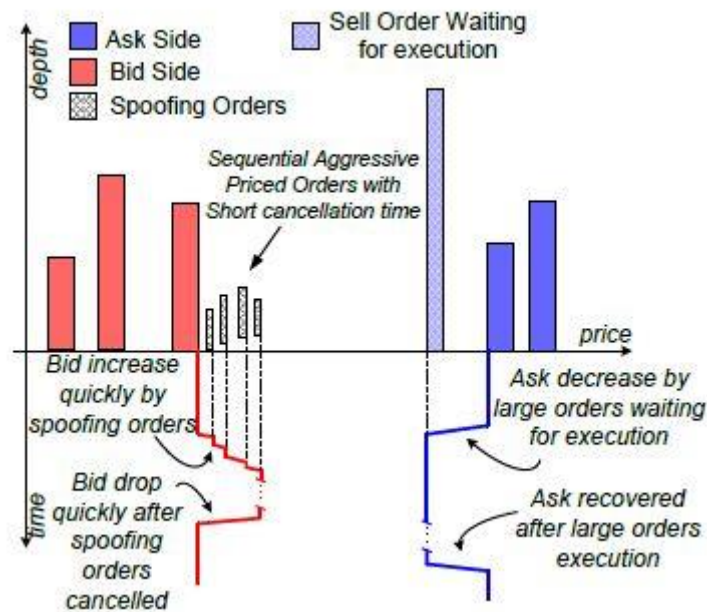


Figure 2.6 - Multi order disruptive trading triggers the bid price to increase quickly and drop also quickly after the spoofing orders are cancelled[7].

Figure 2.7 shows the main components of the overall system. In this model, there are three main engines. The input formulation engine, single-order manipulation detection engine, multi-order manipulation detection engine. The model detects anomaly behaviors in the following set of steps:

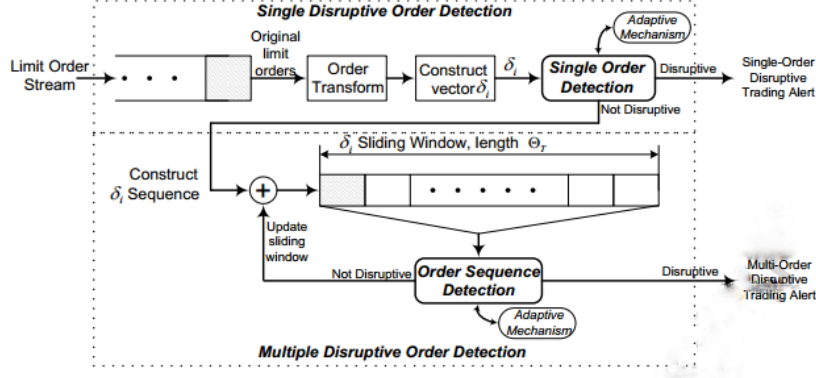


Figure 2.7 - Detecting system [7].

1. The order sequence is first input into the system and a vector containing the attributes are formulated in the input formulation engine.
2. The trades are then sent into the single order manipulation detection engine consisting of OCSVM for single order manipulation detection.
3. Then if an anomaly is detected then it can be alerted.
4. Then the orders are sent into the multi order detection engine consisting of HMM and there the trading are kept in a sliding window and checked for multi order manipulations scattered in time.

Input formulation

Financial data are considered as non-stationary time series. So, in this section time is taken as an index. So, we create a vector in the form of $\delta_i = [\delta_i^p, \delta_i^v, \delta_i^t, \text{DIRECT}_i]^T$. This vector is input into the OCSVM for single order manipulation detection

Let and represent the cancellation and execution time of a limit order, respectively; we can denote the lifecycle of an order as,

$$L_i^{\text{life}} = \begin{cases} L_i^t - L_i^{t,C}, & \text{if canceled} \\ L_i^t - L_i^{t,E}, & \text{if executed} \end{cases} \quad (2.1)$$

Where L_i^t is the submitted time of the order i, $L_i^{t,C}$ is cancel time of order i and $L_i^{t,E}$ is executed time of order i. The order execution and cancelation of a limit order are usually correlated with its volume. Therefore, the Volume Weighted Average Lifecycle (VWAL) of limit orders, in the prior period is calculated as,

$$L_\tau^{\text{VWAL}} = \frac{\sum_{i=1}^N L_i^{\text{life}} * L_i^v}{\sum_{i=1}^N L_i^v} \quad (2.2)$$

Where L_i^v is the volume of order i. Therefore, we convert a limit order to a three-dimensional impulse vector $[\delta_i^p, \delta_i^v, \delta_i^t, \text{DIRECT}_i]$. by the transformation approach as follows,

$$\delta_i^p = \begin{cases} \ln \frac{L_i^p}{B_i}, & \text{for buy order} \\ \ln \frac{L_i^p}{A_i}, & \text{for sell order} \end{cases} \quad (2.3)$$

$$\delta_i^v = \ln \frac{L_i^v}{v_\tau} \quad (2.4)$$

$$\delta_i^t = \frac{L_i^{life}}{L_\tau^{VWAL}} \quad (2.5)$$

Single-order detection

In single order manipulations, manipulators try to tune the quoted price or volume. These kinds of orders usually designed and placed in the market with no sequential relations. To detect these kinds of manipulations this modal uses historical data to recognize patterns. In anomaly detection area, one class support vector machine is the most frequently and ideal algorithm to provide a better classification. In this support vector machine, two support vectors will be generated based on the trading. Any trading that is separated from the support vectors will be identified as anomalies.

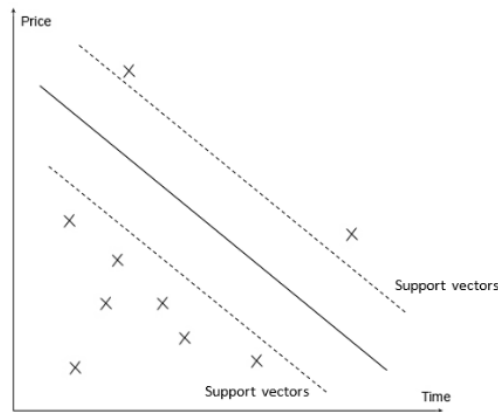


Figure 2.8 - Support vector machine to detect normal and abnormal trading [7].

Multi-order detection

Then the vectors of trading will be stored in a sliding window for processing for multi order manipulations detection. In the HMM model that is in that section will observe the vectors in the sliding window and check for multi order manipulations. From the OCSVM some trading that are part of a multi order trading can be recognized as a legitimate trading. They are put into the sliding window for the use of HMM. This is an advantage when a certain trading that are parts of multi order trading manipulations seem to be not disruptive when analyzed individually.

Disruptive trading can also be carried out by submitting multiple limit orders to the market as a sequence with no intention of execution. Such order sequences quick sweetening the market induces the market impact by successively aggressive quotes or volumes. To further identify the sequential set of orders the modal should capable of generating the observed temporal changes, also the probability of the occurrences of that observed temporal changes. So, to achieve that goal in this modal proposes a hidden Markov modal (HMM). Observable feature states and hidden mixture component states are used in the Markov modal. These concepts modeled on a standard Markov process on the assumption that each state is only depend on the previous states. One type of HMM can be used to identify anomalies. For that first the modal is trained with the normal activities and the cases tested against that modal.

So as discussed in the previous sections, here the detection of manipulative patterns can be framed as anomaly detection problem. That should be a system which captures new or unknown patterns and triggers and alert. To do that an improved version of HMM is needed that is explained next.

Multi-variable Gaussian mixture model

Financial data are generally considered to be inherently non-stationary, and so it is accepted that the statistical properties of such data, for example, the mean and variance, vary over time. To modal that varying data they use probability density function (PDF) of a variable is to approximate its (unknown) density using a Gaussian mixture model (GMM). Figure 10 shows the joint probability density functions(PDF) of δ_i^v (volume component of the vector) and δ_i^p (price component of the vector) for the MFST equity.

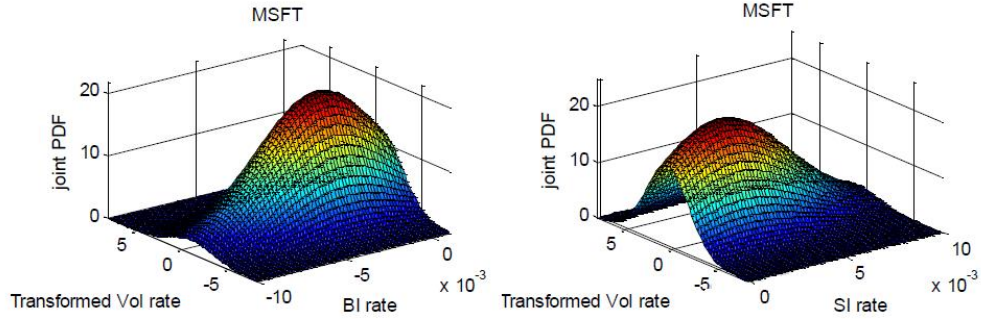


Figure 2.9 - Joint PDF of price vector component and volume vector component [7].

Then they use the traditional hybrid SMOTERUS approach that combines the widely used over-sampling algorithm SMOTE and under-sampling algorithm RUS together to achieve a reasonably balanced ratio in the dataset. In this technique the number of samples of the majority class is reduced to generate a balanced dataset. The majority class is balanced to the equal number of instances of minority class by randomly choosing instances from majority class. Therefore, the number of instances of majority and minority class are balanced.

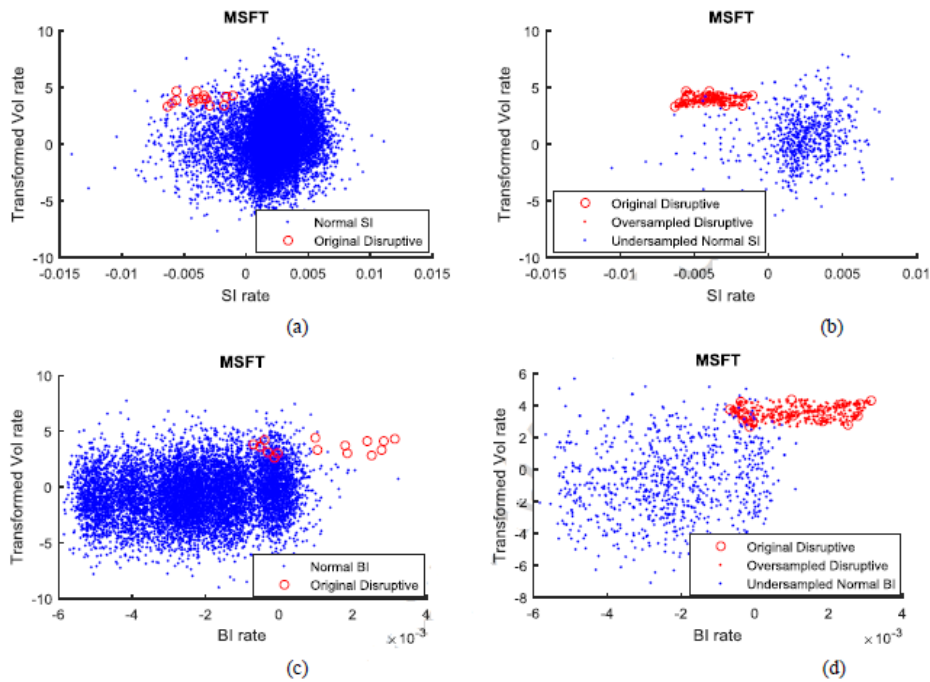


Figure 2.10 - SMOTERUS approach for MSFT data [7].

Figure 2.11 illustrates an example of the SMOTE-RUS approach on the MSFT stock dataset in a short time interval.

(a) buy side MSFT trading data from 9:00-12:00 am on 24 Jan 2013 with a 15:20000 ratio of disruptive (red circles) and normal (blue dots) trading data;

(b) data examples generated by the hybrid SMOTE-RUS approach on the data in (a) with a 0.95:1 ratio of disruptive (red circles) and normal examples;

(c) sell side MSFT trading data from 9:00-12:00 am on 24 Jan 2013 with a 15:20000 ratio of disruptive (red) and normal (blue dots) trading data;

(d) data examples generated by the hybrid SMOTE-RUS approach with a 0.96:1 ratio of minority (red circles and red dots) and majority (blue dots)

HMM-based Model

As discussed earlier they construct the joint PDF of normal and disruptive trading behaviors separately using GMM, and the two PDFs are represented as P_N (normal) and P_D (disruptive). We setup the probability thresholds for P_N (normal) and P_D (disruptive) following the heuristic method usually applied in a one-class support vector machine (OCSVM). As the thresholds are defined, the data are accepted as normal when $P_N(F_t) \geq 99\% * P_N^{\min}$. Similarly, for P_D , the threshold is set as 99% of the lowest cumulative probability of disruptive trading examples. The 1% outlying values are not simply taken as abnormalities but are used to generate the states for the HMM.

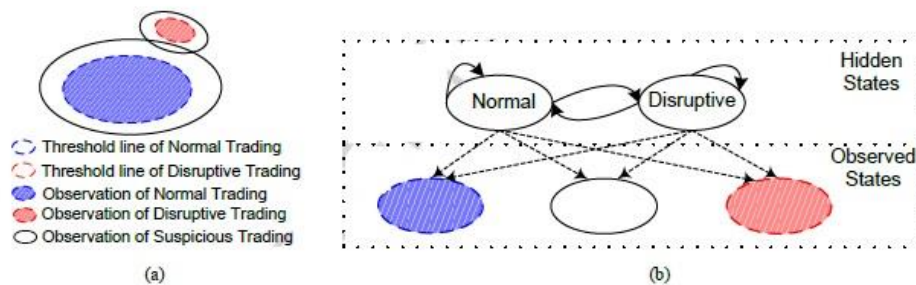


Figure 2.11 - (a) Example states of observed trading behaviors (b) structure of the designed HMM with disruptive states [7].

Figure 12 shows the model of the HMM that is used with this system. In that model blue colored part is the data that satisfies $P_N(F_t) \geq 99\% * P_N^{\min}$. red colored part is the where observed disruptive after the threshold is applied and the white area consist of the data that is not goes to either red or blue. So as the observed states we can calculate the probabilities.

The fundamental property of this design utilizes the inference features of HMM, which could answer two key questions: 1) what is the most likely sequence of hidden states to generate the observed sequence (what is the latent intention given a certain observed trading sequence), and 2) what is the probability of an observed sequence (how likely are we to observe a specific

trading sequence)? HMM provides answers to the two questions by determining the sequence of hidden states most likely generating the observations sequences and the occurrence probability of the observed sequence based on the transition and emission probability matrix constructed during the training process. Those answers can be used as a measure of disruptive trading behaviors.

Detection Algorithm

Single-order detection algorithm can be described as below,

1. Select a historical date and construct the training dataset of the transformed vector.
2. Train the OCSVM using the constructed training dataset.
3. Detect the most updated incoming vector using the trained OCSVM.
4. If OCSVM triggers a disruptive trading alert, the most updated vector is stored in the disruptive trading dataset.
5. If no disruptive trading alert is triggered, the training dataset is slid forward to include the most updated vector and is fed into the adaptive mechanism module for a model update check. If a model update is not needed, the algorithm flow goes to step 2.
6. If a model update is not needed, the algorithm flow goes to Step 3.

Multi-order detection algorithm can be explained as below,

1. Select a historical date and construct the training dataset of the transformed vector δ_i
2. Calculate and construct the training dataset of $\delta_i = [\delta_i^p, \delta_i^v]^T$
3. Collect all known disruptive trading records on the selected financial instrument and construct the dataset of the transformed disruptive vector $\delta_{i,disrupt}$
4. Calculate and construct the training dataset of $\delta_{i,disrupt} = [\delta_{i,disrupt}^p, \delta_{i,disrupt}^v]^T$
5. Apply SMOTETUS algorithm on the dataset of $\delta_{i,disrupt}$ and construct the over-sampled dataset of $\delta_{i,disrupt,over}$
6. Calculate the joint PDF of vectors δ_i and $\delta_{i,disrupt,over}$ using GMM. Set the corresponding thresholds for two joint PDFs.
7. Train the HMM using the constructed vectors and states.
8. For a constructed testing sequence θ_T , detect disruptive behaviors via the trained HMM model.
9. If HMM triggers a disruptive trading alert, the corresponding vectors are stored in the disruptive trading dataset, the algorithm flow goes to Step 5.

10. If no alert is triggered, the training dataset is slid forward to include the most updated vector and is fed into the adaptive mechanism module for model update check. If a model update is needed, the algorithm flow goes to Step 2.
11. If a model update is not needed, the testing sequence is slid forward to include the most updated server and the algorithm flow goes to Step 8.

2.3.2. Trading Networks

Firstly, an entire trading network [8] will be constructed for each stock. Each trader who bought or sold the stock enters the network as a node. A directed link is formed between two traders if they had transactions and the direction of the link is from the seller to the buyer. When a trader places an effective market order, it is possible that the order is executed by several orders on the limit order book submitted by different traders. In this case, the local network structure is a star-like graph with the central node acting as a source if the trader sells or a sink if the trader buys.

The constructed trading networks record the patterns of order execution in limit order book and of the flows of cash and stock shares among investors, which provides a potential opportunity to detect market manipulations of some traders and to further investigate their influences on the market's behaviors'. After scanning all the trading networks, it is easy to find that there are some motif patterns in contrast with the intuitions, which can be considered as evidence in favor of market manipulations of some investors.

Mainly there are three types of abnormal trading motifs in stock trading, namely self-loop, two node loop and two node multiple arcs

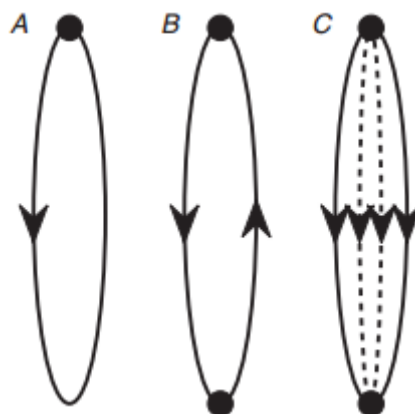


Figure 2.12 - Graphical illustration of abnormal trading motifs (A)Self-loop (B)Two-node loop (C)two-node multiple arcs [8].

Motif A in Figure 2.12 is a self-loop, containing a single trader who sells shares to himself. Such motifs are reminiscent of wash sales, which are improper transactions in which the buyer and seller are the same person such that there is no genuine change in ownership. If a trader utilizes the technique of wash sales, his trading behavior will be identified as motif A from the trading network. Alternatively, a motif A trader is possibly not a manipulator, but the probability is very low.

Motif B in Figure 2.12 is a two-node loop, in which two traders (or more precisely two stock accounts) exchange shares of the same stock. Motif B is the simplest structure embedded in the manipulation technique of stock pools. Stock pools can be identified as a collusive group of manipulators trading shares back and forth among themselves. Motif B can be observed not only in cliques and some circular trading sets but also in other pools.

Motif C in Figure 2.12 contains multiple (at least two) links with the same direction, which happens when one account repeatedly sells to or buys from the same counterparty. The occurrence of network structure of motif C is more common than motifs A and B. Motif C might correspond to a normal behavior when a trader buys or sells the same stock several times and encounters the same counterparty by coincidence. Alternatively, motif C might also appear between two traders within a same pool.

2.4. Unsupervised Detection

2.4.1. Time Series Contextual Anomaly Detection

Reduction-based Contextual Anomaly Detection (CAD) method for complex time series that are not described through deterministic models is presented in [9]. First, a subset of time series is selected based on the window size. Second, a centroid is calculated representing the expected behavior of time series of the group within the window. The centroid is used along with statistical features of each time series X_i (e.g., correlation of the time series with the centroid) to predict the value of the time series at time t (i.e., x_{it}). Table 2 describes the algorithm. This is a lazy approach, which uses the centroid along with other features of the time series for predicting the values of X_i :

$$X_{it}^M = \Psi(\Phi(X_i), c_t) + \epsilon \quad (2.6)$$

where X_{it}^M is the predicted values for the time series X_i at time t , $\Phi(X_i)$ is a function of time series features (e.g., the value of X_i At time stamp $t-1$, drift, autoregressive factor etc.), Ψ specifies the relationship of a given time series feature with the value of centroid at time t (i.e., c_t), and ϵ is the prediction error (i.e., $((X_{it} - X_{it}^M)^2)^{1/2}$). The centroid time series C is the expected

pattern (i.e., $E(X_1, X_2, \dots, X_n)$) which can be calculated by taking the mean or weighted mean of values of time series X_i at each timestamp t . Ψ is defined as the inner product of statistical features of each time series and its correlation with the centroid. Pearson correlation of each time series is used with the centroid to predict values of the time series because if the centroid correctly represents the pattern of time series in a group (i.e., industry sector), the correlation of individual time series with the centroid is an indicator of time series values.

Third, authors assign an anomaly score by taking the Euclidean distance of the predicted value and the actual value of the given time series (the threshold is defined by the standard deviation of each time series in the window). It has been shown that the Euclidean distance, although simple, outperforms many complicated distance measures and is competitive in the pool of distance measures for time series. Then continue this process by moving the window.

Authors have compared this with other competing time series analyzing approaches, KNN and Random walk and shown that the recall is improved from 7% to 33%. In the manipulation detection in stock trading, false negatives are costlier as missing a market manipulation period by predicting it to be normal. So, recall is making a larger impact than the precision. As a future work they are proposing a second phase would consist of weeding out some of the false positives by means of a classifier to improve the precision.

2.4.2. Mathematical Model

Pump-and-Dump Model

In pump-and-dump [10], a manipulator places buy orders, increasing the price and volume of an equity as shown in Figure 2.13. During this period, other investors think that the price is going up and join the buy orders. A manipulator makes profits by cancelling all remaining buy orders and executing sell orders at the higher prices than the price before the manipulation. Therefore, other investors who are not cautious about the fraudulent orders from the manipulator might have bought the equity at a higher price than the usual price.

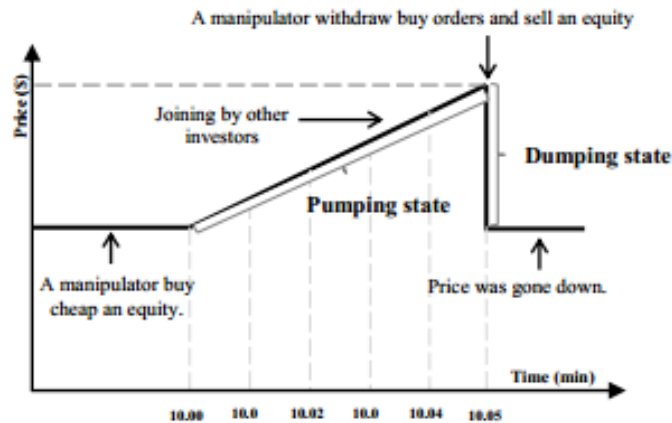


Figure 2.13 - Price variation in a Pump and Dump scenario [10].

In this model three conditions are defined to detect pump-and-dump as below: one condition for the pumping state, and two conditions for the dumping state. First two conditions in the dumping state are the amount of the order cancellations and the matched orders. For the first dumping condition we classify the activity as a dumping position when the amount of the cancellation and deletion of buys orders is more than a threshold (50% here) of the average volume of buy orders. In the second dumping condition, the difference between the highest price of sell orders and the lowest price of sell orders is more than a threshold (0.15% here). It is identified as a dumping state when both conditions are satisfied.

If a dumping state is detected with first two conditions, then it is verified with the pumping condition.

For the pumping condition, it is tested whether the price was going up before the previously detected dumping period. If such behavior is detected, the price rising activity is defined as the pumping state when the difference between the highest bid price that have been matched and the lowest bid price that have been matched at the starting of period is more than a threshold (0.2% here).

When the mentioned three conditions are satisfied, it is concluded that a pump-and-dump manipulation has taken place.

Spoof Trading Model

Spoof trading is one of the most popular techniques that a manipulator uses to make a profit. A manipulator starts placing large spoof ask or bid orders into the market to trick other traders that there are high demands for that equity. The manipulator has no intention for these spoof orders to be matched. The orders will be cancelled when they are about to be matched. These

orders are called passive orders. The volume of these passive sell or buy orders is usually large. Spoof orders can be implemented in two ways: the passive sell price is lower than the current ask price, or the passive buy price is higher than the current bid price. In the Figure 2.14(a), a manipulator intended to buy an equity at the price lower than the current ask price. He placed a large-volume order at a passive price, which is lower than the current ask price as shown in the dashed bar in the Figure 2.14(a). Then, other investors joined into this spoof orders in the Figure 2.14(b), and they expected that the current ask price will decrease. Afterwards, the manipulator withdrew the large spoof sell orders and bought all stocks of remaining sell orders from other investors who were not cautious about this manipulated price as shown in the Figure 2.14(c).

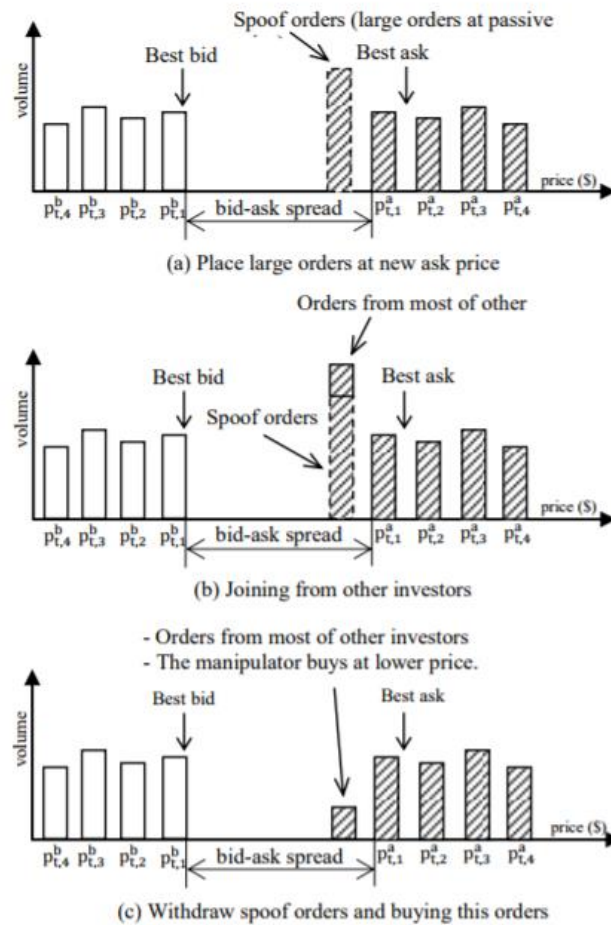


Figure 2.14 - Spoofing trading example [10].

Three conditions were defined for spoof trading: order cancellation has its price close to the current bid or ask price, high cancellation volume, and high volume matched at the last buy or sell order.

For the first condition to be satisfied, the absolute value of the difference between the price of cancellation sell orders and the current ask price should be lower than a threshold (0.5% here). For the second condition, the amount of the cancellation sell orders should be higher than a threshold, which is five times of the summation of matched orders since the starting point in this case. For the third condition, the amount of matched buy orders should be higher than a threshold which is 50% of the summation of matched orders since the starting point. When these three conditions are met, the set of events are treated as a spoof trading event. Then, it is used as a desired output in the training data for the neural network model.

The neural network model consisted of the input layer (25 nodes), the hidden layer (3 nodes), and the output layer (1 node). This model detected pump-and-dump and spoof trading events. The output of the neural network was a binary variable that indicates the probability of manipulation. 1 and 0 for manipulated points and non-manipulated points respectively.

2.4.3. Clustering

Clustering with Particle Swarm Optimization

In general, a cluster can have several cluster boundaries as shown in Figure 2.15. A generic algorithm can be applied here to find the most suitable cluster boundary among all possible cluster boundaries. Particle Swarm Optimization [11] which determines the optimum solution heuristically, could be applied in this kind of clustering approach provided the availability of an appropriate objective function to determine the optimum cluster boundary.

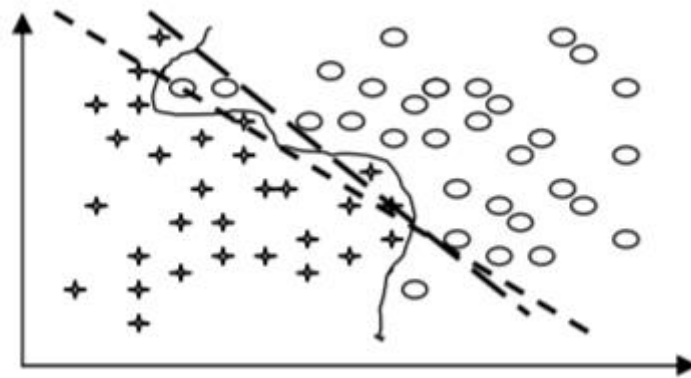


Figure 2.15 - Different cluster boundaries in same set of clusters [11].

An objective function is defined here for clustering with particle swarm optimization that works on the entries of U-matrix to locate the best cluster boundary. U-matrix is simply a matrix of distances between code vectors where lower values indicate code vectors are closer and higher values indicate code vectors are far apart. Logically, clusters are formed where data density is

relatively higher from nearby region so the values from the matrix are used to find the boundary with maximum average value so that optimum cluster boundary lies where density of code vectors starts decreasing. In some data sets it has been found that data that are far apart make a cluster of their own. To cater for this type of data distribution this method automatically determines if the region near cluster center has lower density, then the opposite is done i.e. find the boundary with minimum average value so that optimum cluster boundary is where density of code vectors starts increasing i.e. relative lower values in U-matrix. This proposed novel algorithm is called Expanding Cluster Boundary Algorithm which has been described below:

In some approaches Shrinking Cluster Boundary approach has been used which is entailed with few shortcomings that has been tried to overcome in this new approach of Expanding Cluster Boundary algorithm. Shrinking Cluster Boundary does not perform well if a proper initial cluster boundary is not specified manually whereby the Expanding Cluster Boundary algorithm is insensitive to initializations because it does not require the initial cluster boundary to be specified. This algorithm starts with automatic assignment of a unit square cluster boundary outside manually chosen cluster center anywhere in the region of the expected cluster using graphical U-matrix which grows until the optimum cluster boundary has found or the edge of the u-matrix has reached. There are some other existing methods too, to determine cluster centers automatically that utilizes K-means algorithm with minimal user interference. The clustering algorithm proceeds as follows:

1) Initialization: After specifying the cluster center, PSO automatically initializes a unit principal cluster boundary i.e. closest square boundary outside cluster center along with a set of neighboring boundaries. Additional boundaries help in better acquisition of dispersion of data near the boundary. The cluster centers are approximated as middle point of denser regions through visual inspection of U-matrix. Now for each iteration of PSO:

2) Adjust ranks: PSO generates particles' positions (sequence/array of numbers) and changes their positions by picking 2 indices and swapping them. These indices are called ranks. These sequences have constant size; they are utilized as cluster boundary whose size changes iteratively. The ranks are simply arranged corresponding to the size of expanded cluster boundary i.e. mathematically:

$$New Rank = ceil(\frac{rank * SOM Boundary Size}{PSO Sequence Size}) \quad (2.7)$$

3) Expand cluster boundary: Expand the cluster boundary in the U-matrix gradually away from cluster center by picking both ranks (indices) and moving them one row and/or column outwards.

4) Calculate average weight of cluster boundaries (fitness function): Since the initial principal boundary is extracted from U-matrix, each weight unit is the distance between clusters or neighborhood distance. Additionally, some neighboring boundaries are calculated; this improves the search for cluster boundaries. If a matrix entry is promising to be good cluster boundary, then we change the weight to make it closer to the maximum value of the matrix according to its quality and if the matrix entry is not promising then we change the weight towards the minimum value of the matrix. The higher the average weight the better the cluster boundary. For example, if matrix entry has smaller weight *unit_weight* near the boundary then it is promising, and it is changed to: $unit_weight = max_val - unit_weight$ where, *max_val* is the maximum value of the U-matrix and *unit_weight* is the weight of a single entry of U-matrix which is part of a cluster boundary. The fitness function is simply the average weight of cluster boundaries. The objective function of clustering algorithm uses these values of U-matrix to find maximum average weight through PSO. If the weight unit of an outer boundary, i.e. a boundary that is further away from the corresponding cluster center than the principal boundary, is higher than the weight unit of the principal boundary, then it is added to the weight since it is likely to be a cluster boundary. If weight units for the principal boundary is lower, then again, the corresponding weights are increased. In some cases, the weights can also decrease; for example, if the weights for some outer boundary are lower than the corresponding values for the principle boundary, then the outer boundary is in fact not a boundary; instead, it belongs to the interior of another cluster. Similarly, inner boundaries (interior part of cluster) whose weights are higher than the corresponding values of the principal boundary in fact do not belong to the interior of the cluster; rather it is part of the cluster boundary. Hence the fitness function is simply the average weight of the cluster boundaries. Each particle of a swarm consists of the value of this fitness function which is simply the average weight of cluster boundaries. Each particle of a swarm consists of the value of this fitness function which is simply the average weight of cluster boundaries.

5) Solution: Repeated application Steps 2 to 4 results in a maximum average weight, which corresponds to the best cluster boundary. The best boundary is the one which has highest average value of distances between points and their neighbors.

3. Design

This chapter describes the design of the proposed stock market manipulation detection tool. The proposed tool consists of a client web portal and a server.

We analyze the order events by time based windows or order event based windows. Window can be considered as chunk of order messages for a particular equity. Mainly three approaches have used to detect the manipulation in stock trading. Manipulator will slightly increase the price for an equity. We have used price gap visualizer because, if we get the price difference for all the windows, it will be easier to visualize the manipulated frame in a different view. Traders are using different events in market. If a trader uses different number of events in a window the randomness will be changed, and it can be detect using entropy. If a window has major changes compare to other windows, when the windows are clustered anomalous windows can be identified using the proposed clustering approach.

Section 3.1 describes the workflow diagram which explains how the communication is happening between components. Section 3.2 describes the layered architecture diagram and all the components briefly.

3.1.Workflow Diagram

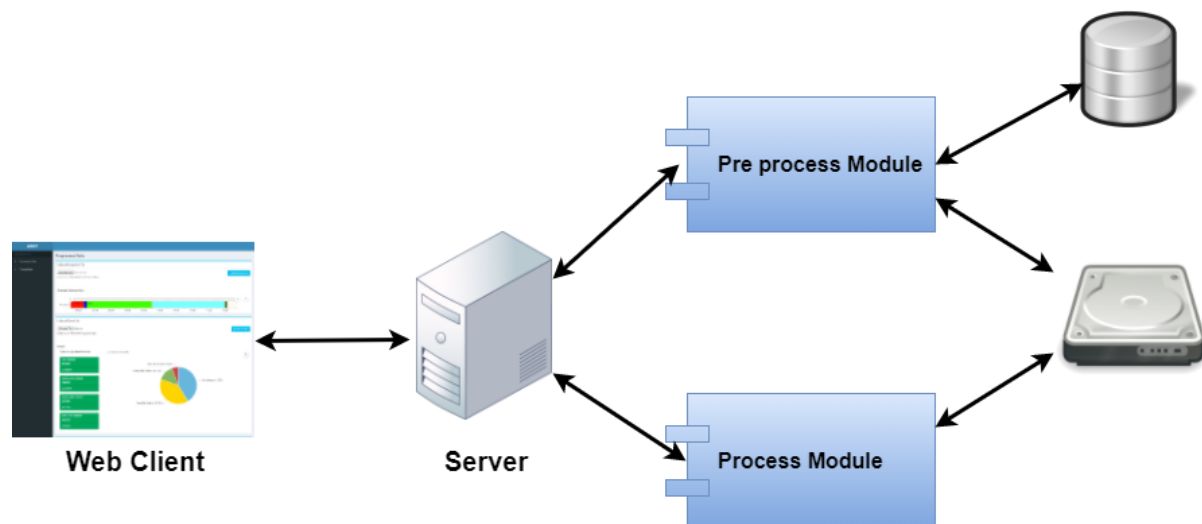


Figure 3.1 - Workflow diagram.

Figure 3.1 illustrates the workflow diagram of the proposed manipulation detection tool. We assume that user will uploaded the stock market trading data to the tool as two files, namely session data file (i.e., all stock trading sessions) and stock trading data file (all stock trading order data) to the server. After uploading those files, user will be shown the statistics of the

files. Then the user has to select options needed to process the files and after processing the user will be redirected to another view by displaying the most suspected window as a result of the processing (either by order or by time) with a detailed description.

3.2.Layered Architecture

Figure 3.2 shows the high-level architecture of the proposed system.

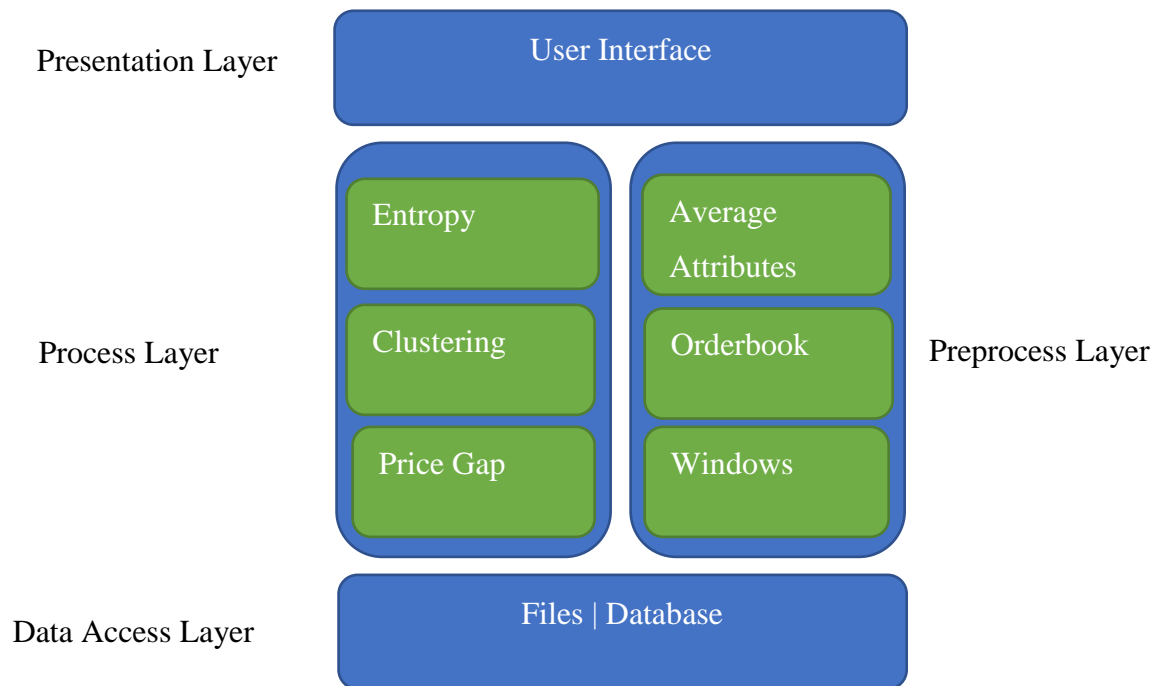


Figure 3.2 - High level architecture design.

3.2.1. Presentation Layer

Presentation Layer of the system handles all the communication with the user. It uses a REST API to communication with the server and provides functionalities to upload files, select analysis options, view statistics, etc. Use REST API decouples the client and the server connections. Hence, with the initial URI, the client does not require the routing information, client can have generic ‘listener’ interface for notifications, etc. The presentation layer also includes input validation and guide the user from making false operations on the application.

3.2.2. Pre-Processing Layer

Preprocessing layer reads trading data, check whether the dataset is in given format and inform user about basic statistics related to the dataset such as no of buy and sell orders. Moreover, preprocessing phase extracts relevant features from the dataset which are used to develop the model in the processing stage. Different machine learning models need different features and transformations to detect the manipulation time frames. Furthermore, when we follow feature

extraction techniques (eg.. probability value of order events and normalization) and prepare the data, sometimes some algorithms can deliver better results without the preprocessing. But for the model used in this application requires the outcomes of a preprocessing stage.

Therefore, by adding many different preprocessing techniques to this data, we exercise a handful of algorithms on each view of the dataset. This helped us to feed the better preprocessed data to the machine learning models in the system.

Windows

For analyzing the data, we basically divide the dataset into windows because windows allow us to detect particular manipulation within each area and we assume that all windows should have the same properties (eg.. execution type average, price volume average, etc..).

We are using two types of windows to process. They are,

1. Event based windows
2. Time based windows

In event based windows, we specify what is the number of orders to be included inside a window. So that the data is divided into chunks of the desired amount and in each pre-processing technique different types of values are calculated for that window.

In each type of pre-processing we are going to calculate a value for a window.

$$Probability_{event\ type} = \frac{number\ of\ event\ type}{number\ of\ all\ events} \quad (3.1)$$

$$Average_{executed_price} = \frac{Total\ executed\ price}{Total\ order\ count} \quad (3.2)$$

$$Average_{executed_volume} = \frac{Total\ executed\ volume}{Total\ order\ count} \quad (3.3)$$

Equation 3.1 will calculate the average events (new orders, amend orders, cancelled orders and executed orders). Equation 3.2 will calculate the average executed price for a window and Equation 3.3 will calculate the average executed volume for a window. The reason for calculating executed quantities for windows is that we are assuming that manipulators are directly focusing on executed quantities rather than amending and cancelling.

$$\mu = \frac{\sum_{i=1}^N x_i}{N} \quad (3.4)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \quad (3.5)$$

$$X_{normalized} = \frac{X - \mu}{\sigma} \quad (3.6)$$

Equation 3.5 and 3.6 have used for calculating mean and standard deviation for a window. Then normalized value can be calculated from equation 3.7 [12].

$$\Delta P_{time} = P_{t+1} - P_t \quad (3.7)$$

Equation 3.4 is used to calculate the price gap between order messages. For processing phase, only the values calculated for each window is considered. Then the whole window acts like a value point among the others.

Orderbook

Orderbook module is used to generate the model of the actual orderbook according to the order messages data file. This order book lists the number of shares being bid or offered at each price point, or market depth and this will help to identify how the market behaves in a particular time/order frame. Orderbook simulation process is done in every time frame if it is requested by the user.

3.2.3. Processing Layer

This phase consists with different machine learning algorithms and visualization components and preprocessed data will be filtered to this phase. After analyzing preprocessed data, the anomaly time/order window will be displayed with relative parties.

Clustering

K-means clustering algorithm with a dynamic k value was used in the data analytic module of the tool. Anomalous clusters were identified according to an anomaly score.

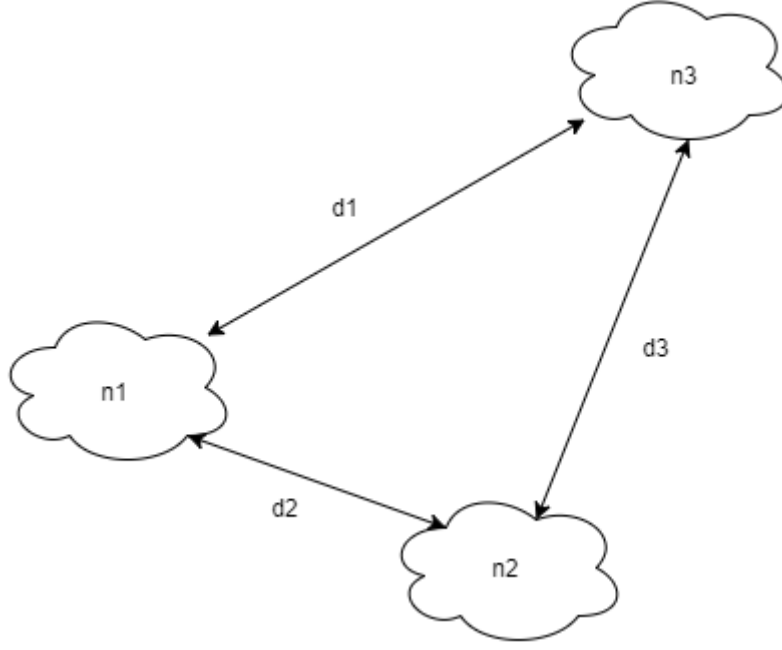


Figure 3.3 - Anomaly Score.

If the number of data points in each cluster are $n1$, $n2$ and $n3$ and distances between them are $d1$, $d2$, $d3$, anomaly score of $n1$ cluster is,

$$Distance\ Score = \frac{d1 + d2}{2} \quad (3.8)$$

$$Anomaly\ Score = \frac{Distance\ Score}{No\ of\ data\ points} \quad (3.9)$$

$$Anomaly\ Score = \frac{d1 + d2/2}{n1} \quad (3.10)$$

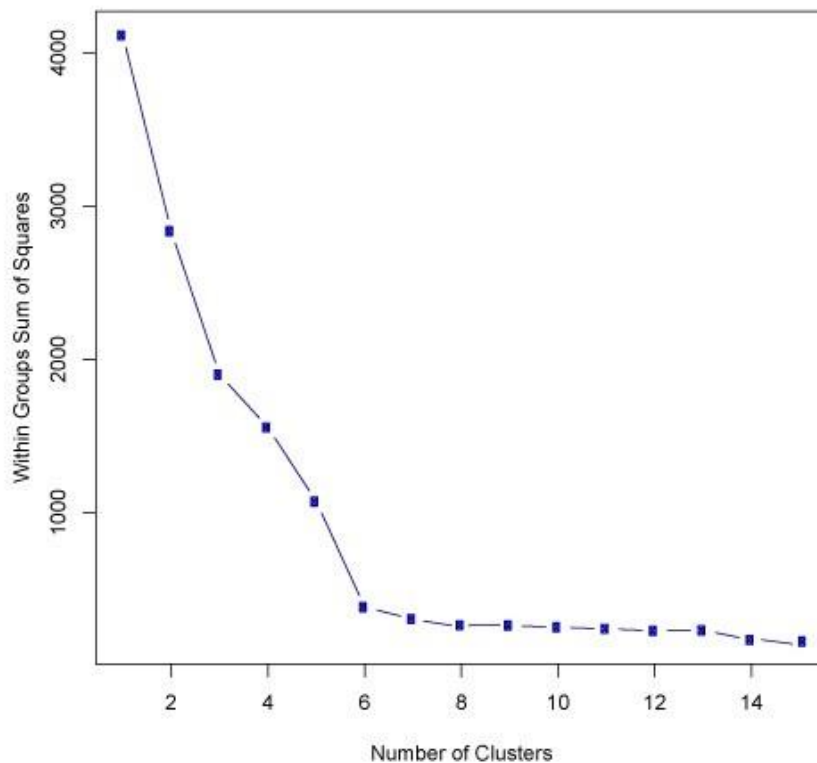


Figure 3.4 - Elbow method to generate dynamic K value.

Figure 3.4 shows how the usage of elbow method to calculate the dynamic K value. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k , and for each value of k calculate the sum of squared errors (SSE).

Then, plot a line chart of the SSE for each value of k . If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k (the SSE is 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster). So, our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k .

Entropy

Entropy, as it relates to machine learning, is a measure of the randomness in the data being processed. The higher the entropy, the harder it is to draw any conclusions from that information. For an example, imagine the situation of flipping a coin which is an action that

provides information that is random. For a coin that has no affinity for 'heads' or 'tails', the outcome of any number of tosses is difficult to predict. Why? Because there is no relationship between flipping, and the outcome. This is the essence of entropy. The entropy can explicitly be written as,

$$H(X) = \sum_{i=1}^n P(x_i) I(x_i) = - \sum_{i=1}^n P(x_i) \log_b P(x_i) \quad (3.11)$$

Relation between the entropy concept and our problem lies with the trading patterns that are in the providing data. Here, order execution types are considered. If the data has these order execution types randomly distributed over the whole dataset, we can identify an evenly valued entropy value set. But if any time window has a weighted arrangement of a certain execution type, we have that time window with a more biased entropy value. That kind of a result is obtained by applying the entropy concept in the processing layer. Figure 3.5 shows the entropy variation graph.

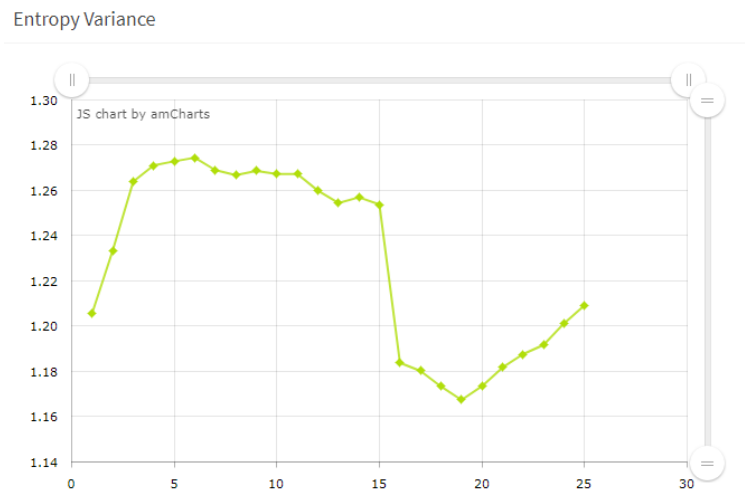


Figure 3.5 - Information entropy of execution types.

Price Gap Analysis

Consecutive price gaps will be calculated in this phase. After normalizing those values, the plot will be displayed in the user interfaces. It would be easy to differentiate anomaly price gap window and other windows.

3.2.4. Data Access Layer

The main advantage of keeping a separate data access layer is to keep the code we use to pull data from your data store (files and database) separate from business logic (Pre-processing layer and processing layer) and presentation (User interface) code. This way, it is easy to

change data stores and no need to rewriting the whole thing. In this system, data access layer is used to access the files (.csv format) and database(MySQL).

Data Files

Mainly, there are two data files. Session file where all the session states are included in a day and a data file where all the trading are included. If a user needs to detect manipulation, he needs to upload both files to the server.

Database

Database is used to track the uploaded file names, output file names and file accessing times. Main benefit of this part is to upload the files which are used recently.

4. Implementation

This chapter gives a detailed description about implementation of our system. Section 4.1 describes the tools, languages and technologies used. Section 4.2 describes the implementation details of each component in the system.

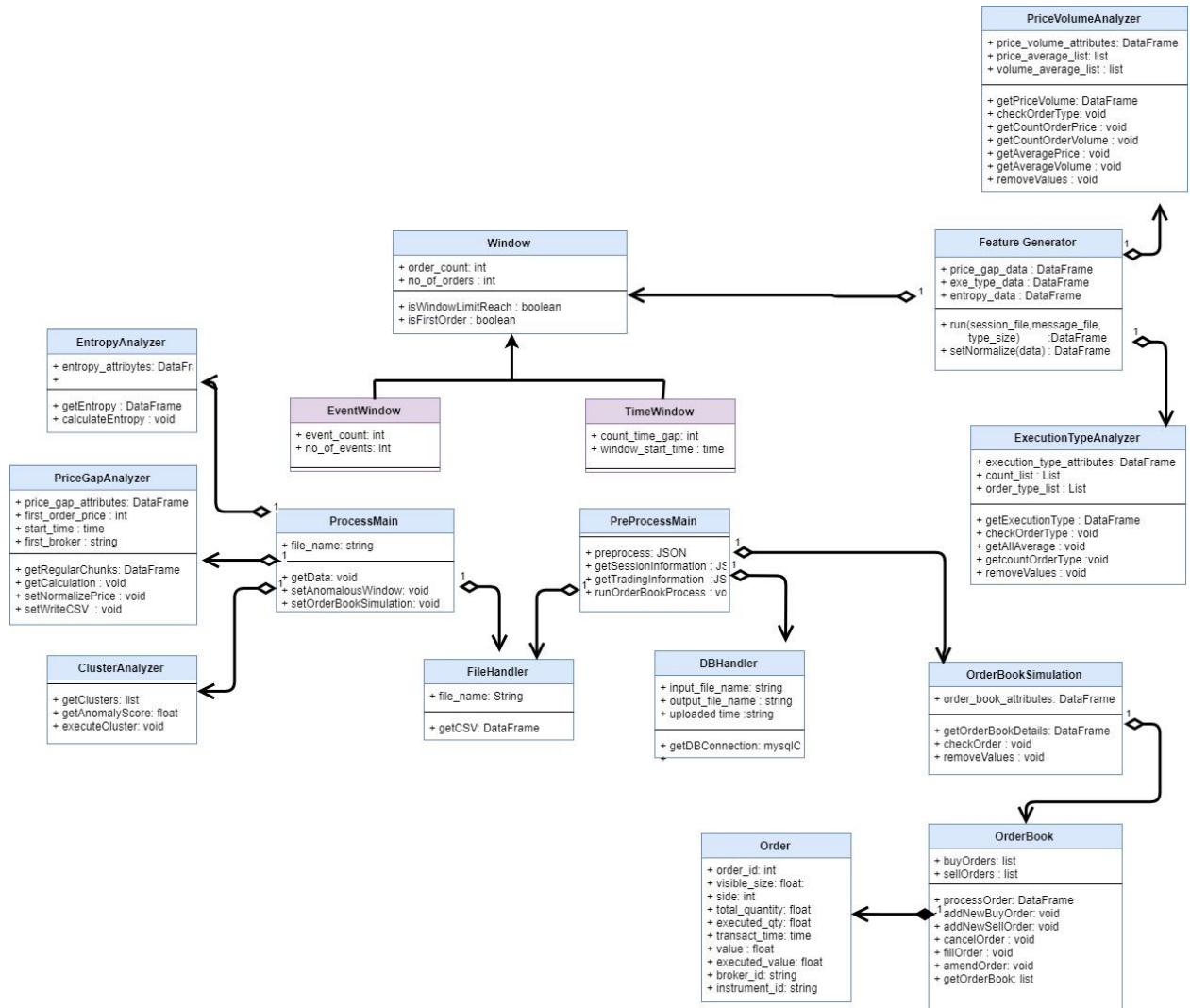


Figure 4.1 - The class diagram of the system.

4.1.Tools and Technologies

Our system is developed having two main sections, frontend and the backend. The backend or the server is written entirely in Python3 using the Flask framework [13]. This enabled us to define and call REST API endpoints and communicate through a client. The client or the frontend is written in Javascript using AngularJS framework. Third party solutions like Pandas, Amcharts, etc. are used to create functionalities in the project. We used the PyCharm IDE by JetBrains to organize, edit the codebase. To keep the file's local history, we have used MySQL

as a database service. For version controlling we used Git and Github. Our project is hosted at Github[14]. It is easier to maintain the versions and do collaborative work using Git services.

4.2.Preprocessing Module

Window

As we discussed previous chapter window is a chunk which consists with orders or time. In time-based windows, we can specify the time in minutes. Then the data is divided into chunks based on the time. By looking at the *transact_time* attribute of the Order, it will be allocated to a window. Not like event windows there can be time windows which does not have an order inside it.

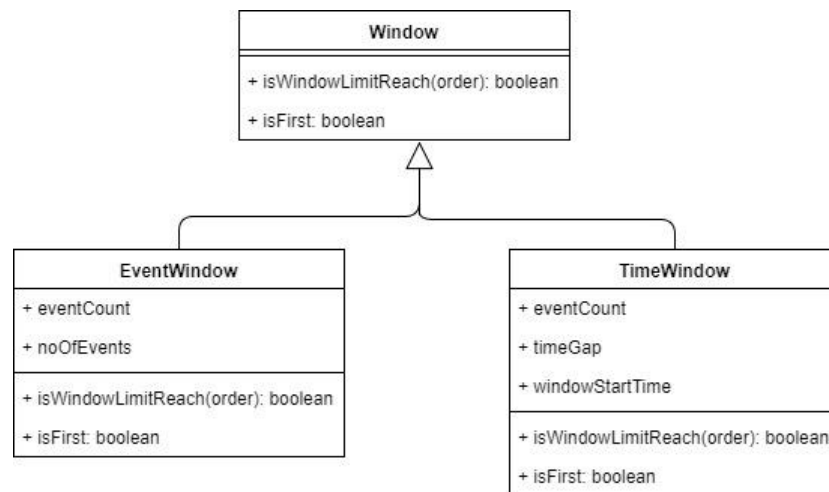


Figure 4.2 - Window design.

For each type of pre-processing we are parsing a window object with it. So, all the pre-processing work happens using that window object.

Orderbook

Orderbook is the central entity of all the entities of the project. Orderbook has the record of the buy and sell amounts of equity amounts. By looking at the orderbook at a given time, we can get a snapshot of the market situation at that time. So, we decided to simulate the orderbook as a part of the data processing.

First, we have a model class for an Order. An order is an entry at the Orderbook where the transactions are recorded. For every order message that is in the orderbook message, we create an order instance and pass it to the orderbook simulation to process the order.



Figure 4.3 - Orderbook design.

According to the *execution_type* attribute there are four main types,

1. New orders - A new buy or sell order that a trader has put in to the orderbook for future processing. Denoted by the "0" value in the program.
2. Cancel orders - Deletion of an order that was in the orderbook. Denoted by the "4" value in the program.
3. Amend orders – Changing an order in the orderbook. Denoted by the "5" value in the program.
4. Fill orders – Execution happened for an order that was in the orderbook by matching with another order. Denoted by the "15" value in the program.

Volume – volume is expressed as three fields. Each order has quantity values has a different meaning to that. *total_qty* shows the total volume of the order. *visible_size* shows the amount of the volume that is visible to the orderbook. *executed_qty* is the volume that has been get executed when a fill order happened.

Side – Side denotes whether the order is a buy order or a sell order. "1" denotes a buy order and "-1" denotes a sell order.

Value – value is the price of the order that has been chosen by the trader. Since the market orders are placed without denoting a price this fields comes empty. Executed value is the actual price that the order gets executed with another order.

For every order that fed in to the orderbook is been processed by its type. If it is a new order, then it gets processed and the price point of that order put in to the corresponding ArrayList either buy or sell as shown figure 4.4.

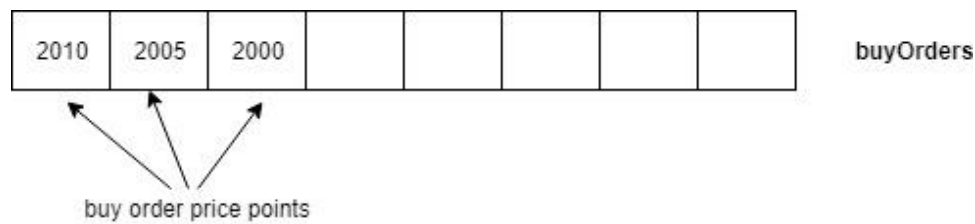


Figure 4.4 - Buy orders price points.

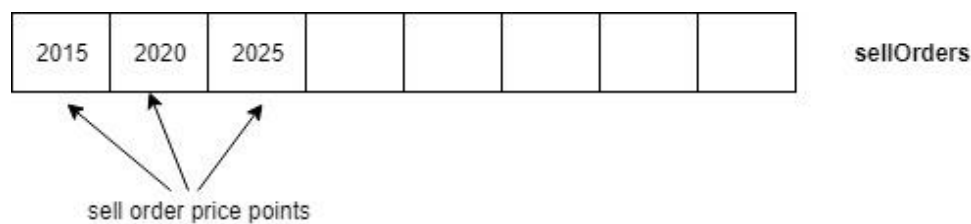


Figure 4.5 - Sell orders price points.

These price points are the details which actually shown in the orderbook simulation. Buy order list is reverse sorted so that the largest buy order come to the front. Sell order list is sorted so that the smallest order come to the front. If an order came with a price point that is already there in the corresponding list, nothing needs to happen in the above two lists. That details kept in a dictionary data structure. It keeps the price point and the list of order ids that of the orders corresponds to that price point.

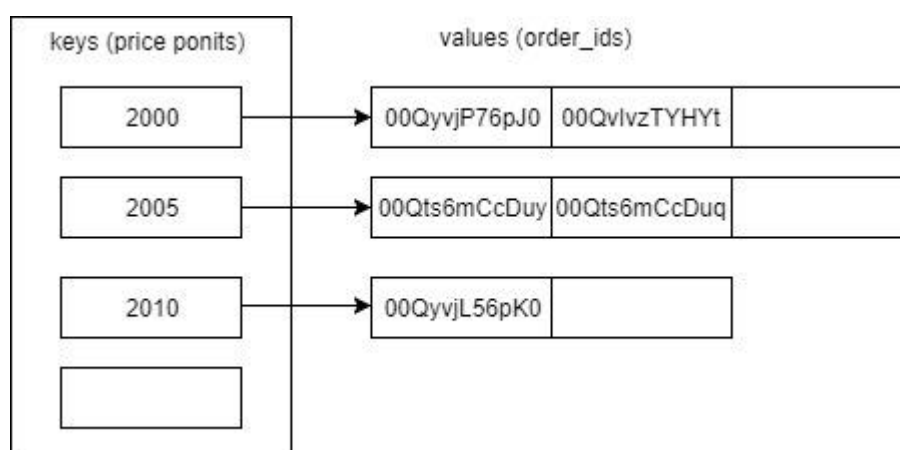


Figure 4.6 - Dictionary structure of order details.

When the details needed for a specific *order_id* it fetched from a data frame that maintains to keep track of all the details.

4.3.Processing Module

4.3.1. Clustering Module

K-means clustering

The idea of this algorithm is to find groups in the data, where the number of groups is represented by the variable K , then separate most different groups or anomalous groups as fraudulent behaviors. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K -means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster)

Rather than defining groups beforehand, dynamic k-means clustering allowed us to find and analyze the groups that have formed organically. The following section describes how the number of groups can be determined.

Therefore, in our approach, the market data of a single day were divided into frames of user's choice and normalized mean values of different features during these time intervals were used as data.

A vector combining all these features was used as a data point. These vectors were clustered using the clustering algorithm.

Dynamic K value

The K-means algorithm is somewhat naive as it clusters the data into k clusters given the k is determined early and provided at the start, even if k is not the right number of clusters to use. Therefore, when using k-means clustering, we had to use some way to determine whether they are using the right number of clusters.

The method used to find the number of clusters is the *elbow method* as described in section 3.2.3.

Anomaly Score

After clustering the market data, we had to differentiate anomalous clusters from normal clusters. For this we used an Anomaly score which we calculated as follows.

First, we selected one cluster and calculated the mean of the distances between the centroid of that cluster and centroids between every other cluster. Then that mean distance value is divided by the number of elements in the cluster. The idea behind this is, for a cluster to be anomalous the data points in that cluster should differ from other data points i.e. they should be distant from other data points. To take this factor into account we had to take a distance measure into anomaly score calculation. Also, manipulative behaviors do not appear in stock market data in huge numbers. There may be few fraudulent behaviors among millions of normal transactions. Therefore, the amount of data points in a cluster should also be considered when calculating the anomaly score.

Here, for clusters which are far away from the rest of the data set, distance score value is high. When the distance score is high, and the no of data points is low, those type of clusters get the highest anomaly scores.

Entropy Module

As the second data analytic module we analyzed the Information Entropy of several features of the stock market data. Generally, entropy refers to disorder or uncertainty. Information entropy is the average amount of information produced by a probabilistic stochastic source of data. The measure of information entropy of data value is the negative logarithm of the probability mass function for the value. Therefore, when the data source has a lower-probability value (i.e., when a low-probability event occurs), the event carries more information than when the source data has a higher-probability value. The amount of information of each event becomes a random variable whose expected value is the information entropy.

Information entropy of four execution types and two sides of orderbook were calculated. Four execution types: new orders, amend orders, fill orders and cancel orders gave the most accurate results when the information entropy values were used for the analysis. As shown in the Figure 3.5, during the time frame of the manipulation the information entropy of execution types have decreased rapidly. Meaning that the trading behavior in that time frame is different compared to the other time frames.

When most the manipulations take place only one or two types of execution types are executed for the buildup of the manipulation. For example, in pump and dump manipulation type mostly new orders or amend orders are used to create an artificial demand for an equity. Then the information entropy of execution types in that time interval decreases rapidly. This is the reason

for the drastic change in entropy in the Figure 3.5 during the manipulated time frames. Apart from the lowest entropy value, we can also see several local minima and their occurrences.

Price Gap Analysis

Some manipulation types like Momentum Ignition scenario, manipulators try to increase the price of the equity gradually. Then normal traders will see it like a high trading equity. Then they will try to trade on those equities. After visualizing price gap graph with respect to time, manipulated time frame or order frame shows the significant difference compare to other windows. Therefore, this method can be used as a visualization approach in this tool.

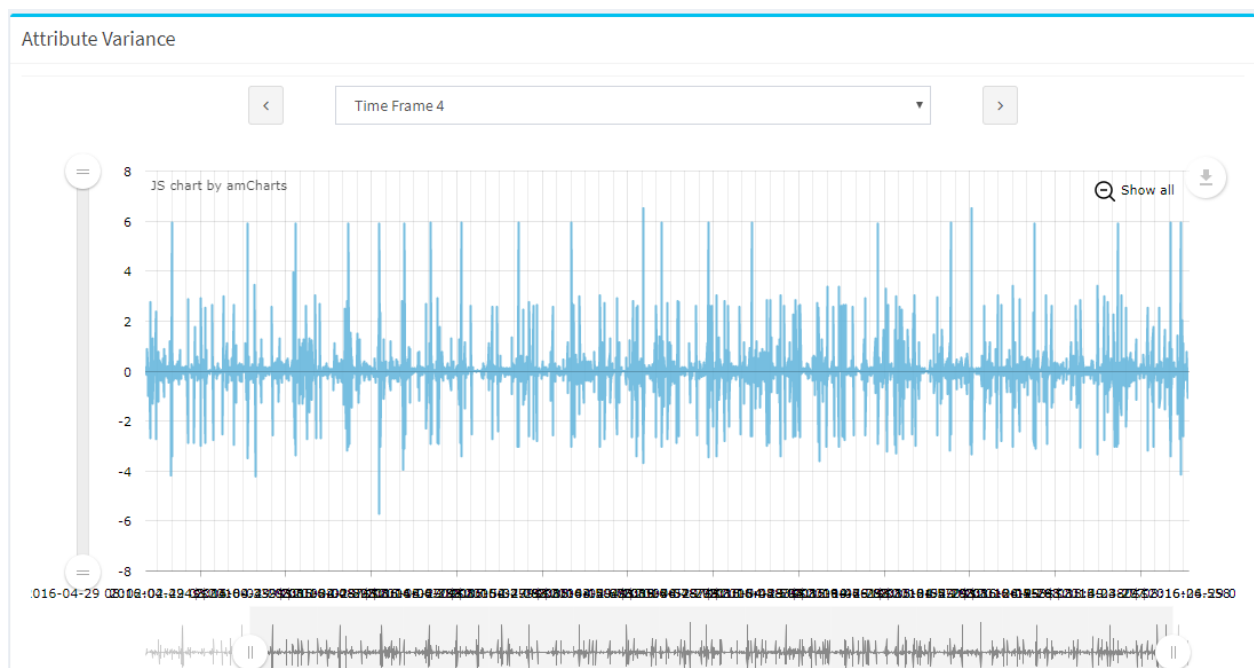


Figure 4.7 - Price gap window without manipulations.

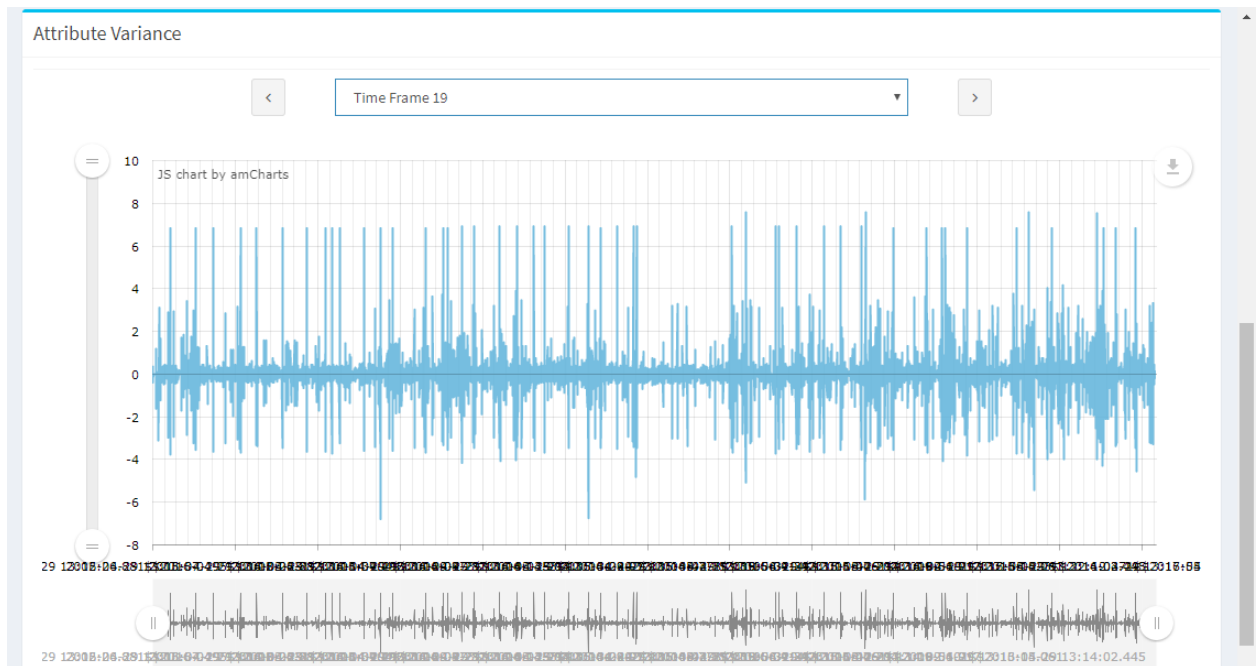


Figure 4.8 - Price gap window with a manipulation.

4.4. Dashboard Implementation

This dashboard application has functionalities from uploading the necessary files to the backend server and getting processed results out of the system. There are several visualizations to get a visual feedback about the data, even for the statistics of the uploaded files. All the visualizations can be saved in one of the selectable file formats.

4.4.1. Session File Uploader

Every time user must choose the corresponding session file for the corresponding day. The tool allows the user to upload session details of the corresponding day before uploading the trading data. A typical session file can be defined as follows.

Transaction Time	Session Name
2016-04-29 06:00:00.015	Pre-Trading
2016-04-29 06:50:00.004	Pre-Trading
2016-04-29 06:50:00.004	Opening Auction Call
2016-04-29 07:00:11.102	Opening Auction Call
2016-04-29 07:00:11.102	Regular Trading
2016-04-29 11:00:00.005	Regular Trading
2016-04-29 11:00:00.005	Periodic Auction Call 1
2016-04-29 11:02:16.102	Periodic Auction Call 1
2016-04-29 11:02:16.102	Regular Trading 1
2016-04-29 15:30:00.006	Regular Trading 1
2016-04-29 15:30:00.006	Closing Auction Call
2016-04-29 15:35:13.102	Closing Auction Call
2016-04-29 15:35:13.102	Closing Price Publication
2016-04-29 15:35:14.060	Closing Price Publication
2016-04-29 15:35:14.060	Closing Price Cross
2016-04-29 15:40:00.102	Closing Price Cross
2016-04-29 15:40:00.102	Post Close
2016-04-29 16:15:02.186	Post Close

Table 4.1 - Session table details.

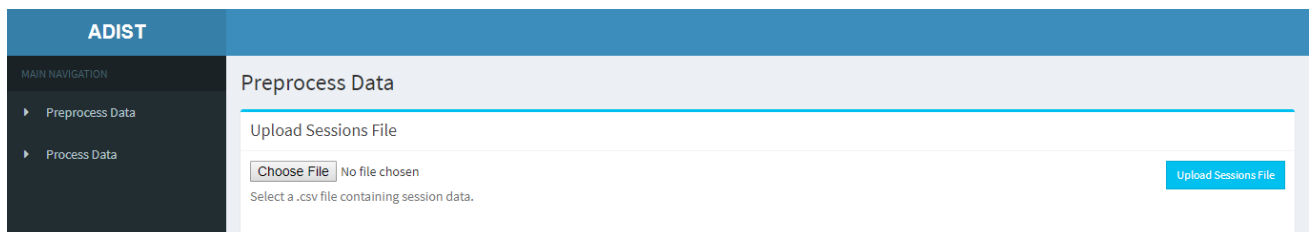


Figure 4.9 - Session file uploader.

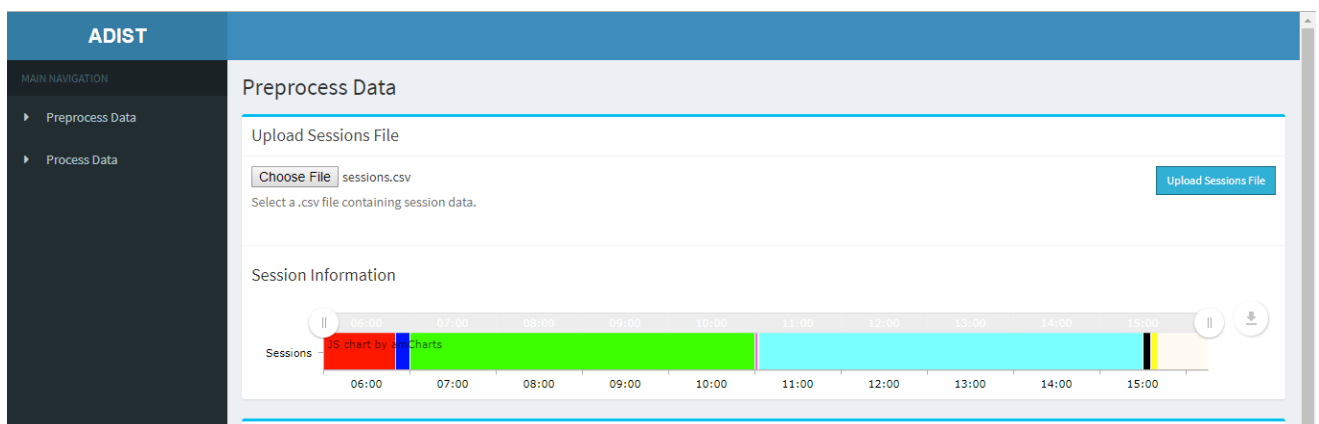


Figure 4.10 - Session details displayed by the tool.

It is easy for users to view spreading of trading times in a day. After viewing the Gantt chart, he can decide the size of the time window or order window. This lets the user to get an idea about the distribution of each session though the day's total trading time.

4.4.2. Trading Data File Uploader

Figure 4.11 - Trading data file uploader.

The user has to upload the trading data file next. This step processes the data initially to get a graphical view to the user. It visualizes the number of new orders, cancelled orders, amended orders and executed orders both in pie chart and as a quantity. This graphical view shows how many new orders got executed, amended or deleted during the regular trading time of the day.

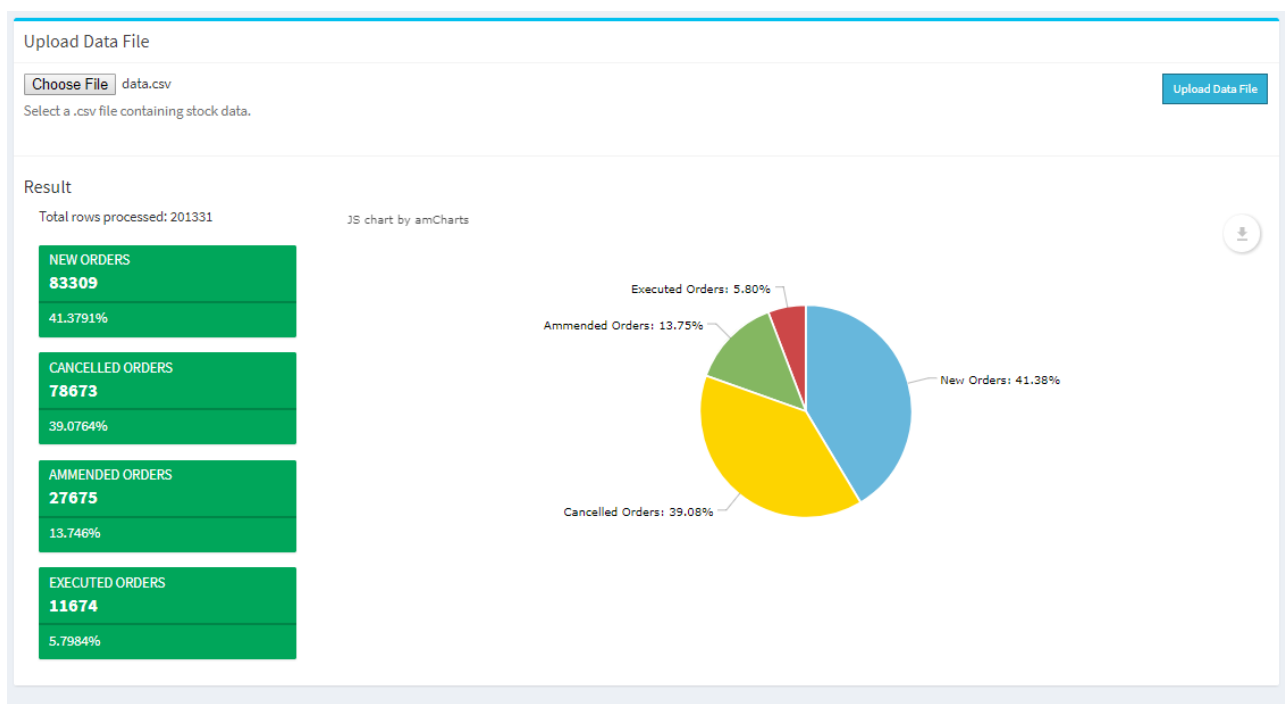


Figure 4.12 - Trading data statistics shown in the tool.

User can decide whether to select order chunks or time-based chunks. If the number of trading orders are much lower, then user must choose order by option. If he selects order by time, then non-manipulated time frames will be detected as manipulated ones because number of orders in a particular time frame is a lesser value. Also, the results can be vary depending on the window size that user chooses. Then there is an opportunity to test the data on two different time windows and compare the results.

4.4.3. Window Size Chooser

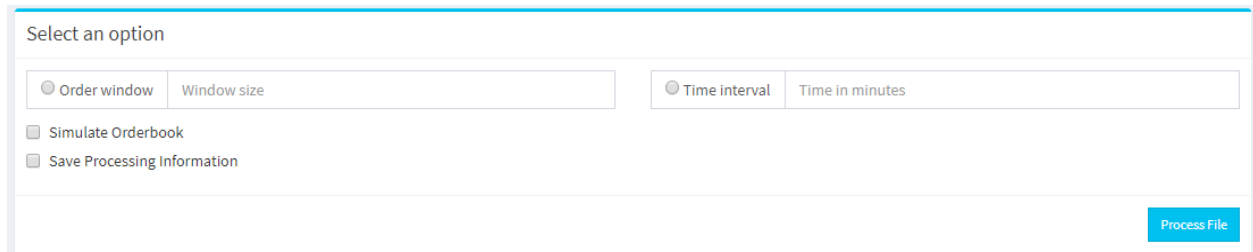
The screenshot shows a web interface titled "Select an option". It contains two main sections. The first section has two radio buttons: "Order window" (selected) and "Time interval". Each radio button is followed by a text input field; the first field is labeled "Window size" and the second is labeled "Time in minutes". The second section contains two checkboxes: "Simulate Orderbook" and "Save Processing Information". A blue "Process File" button is located in the bottom right corner of the form.

Figure 4.13 - Window size input fields and other options.

This window allows users to select either an order-based window or time-based window. Apart from that it allows to select orderbook simulation. Assume a user needs to verify whether the recognized manipulation could be happening with given time/order frame orderbook scenario. Then he/she needs to select this option and do the process. With this the user can visualize the orderbook in the next window. Since the processed files will be cleaned after each process, user must select save processing information option to keep his processed data for future. His first operation costs a lot of processing time and the second operation costs space for generated files. Hence both the operations are added as optional check boxes to be selected if needed.

The results generated for the same data set could be vary depending on what option was chosen for the time interval. So, there is a comparability between the results generated by selecting the two options.

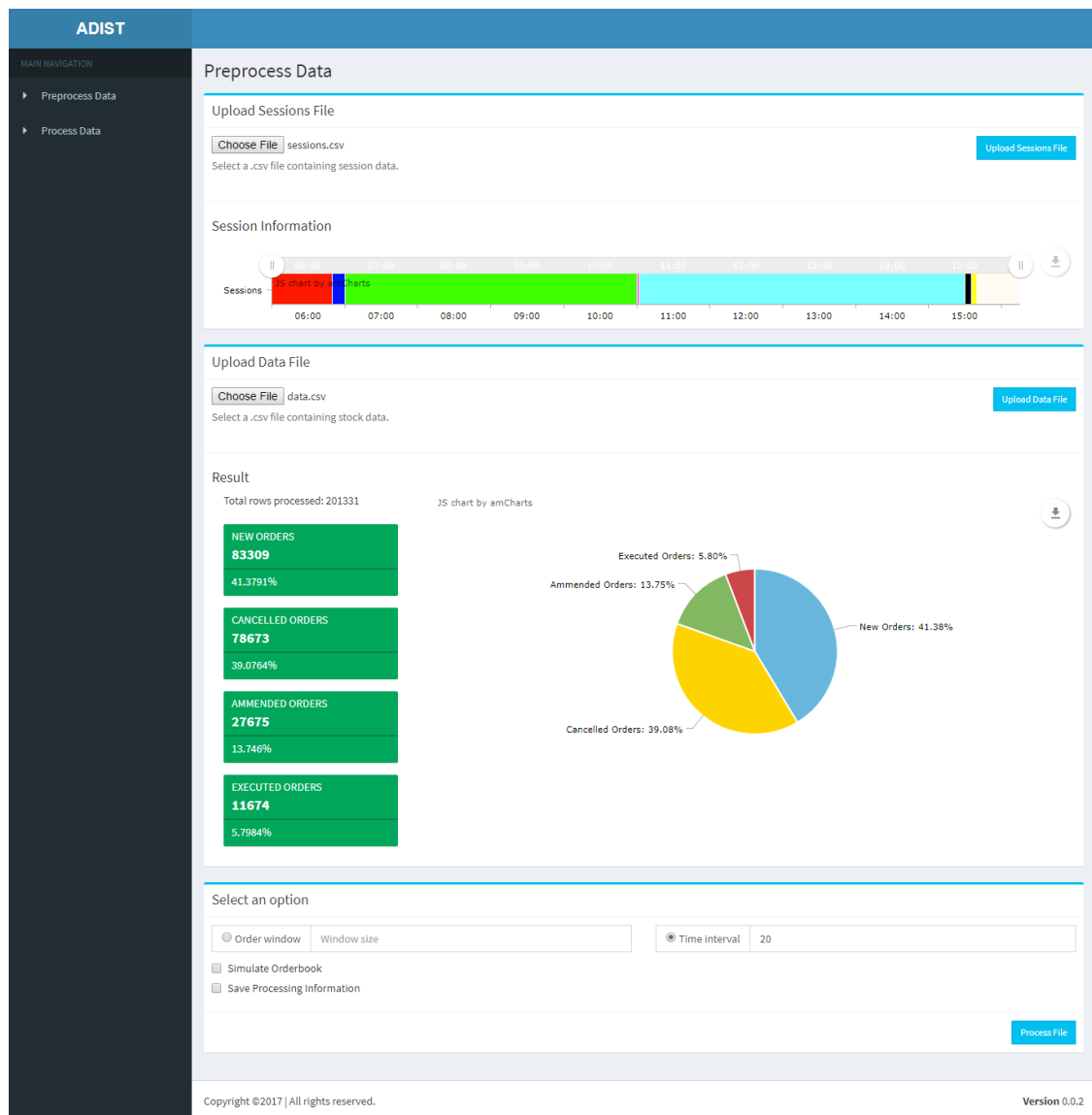


Figure 4.14 - Tool after the end of pre-processing.

4.4.4. Entropy Display

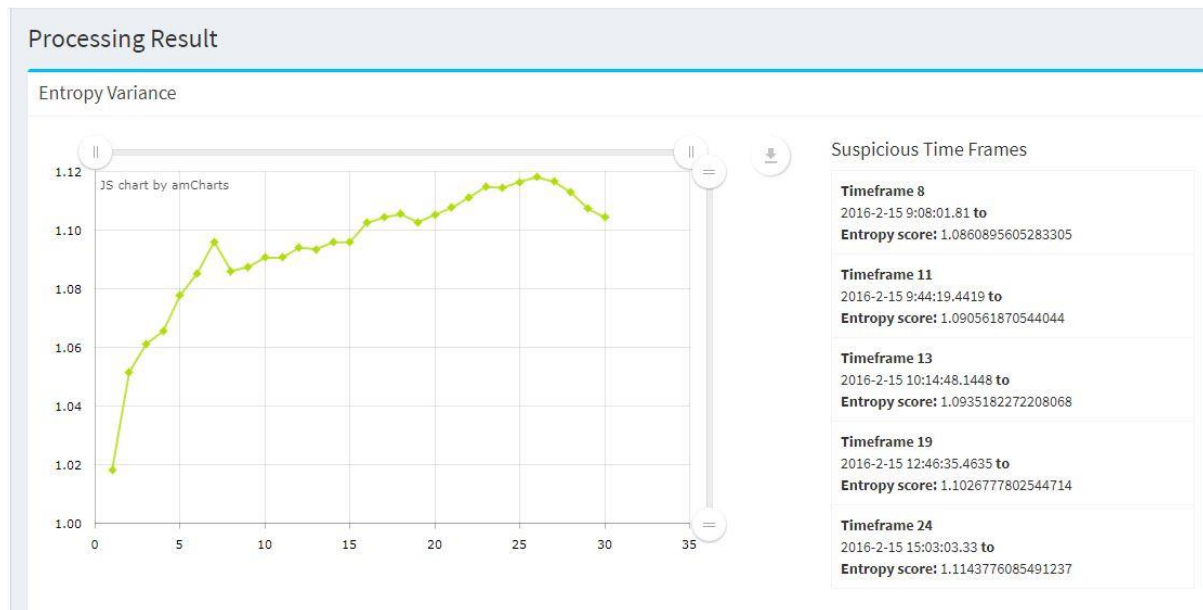


Figure 4.15 - Entropy variation displayed in the tool.

This is in the second page that user is redirected with the processed data and results. This line graph shows the entropy variation between the different windows and the values. On hover, the user can read the different entropy values and the local minimums of the graph are shown on the right side of the line graph. These local minimums are the suspicious time frames in the given data set resulted by different patterns of trading related to surrounding time frames.

4.4.5. Price Gap Variation Display

Price gap variation display is the line graph showing the price gap variation in each time or order frames. There is a navigation between the time frames to visualize any price gap variation in the given data set. In the tool the user can zoom in and out to specially focus any point on an axis.

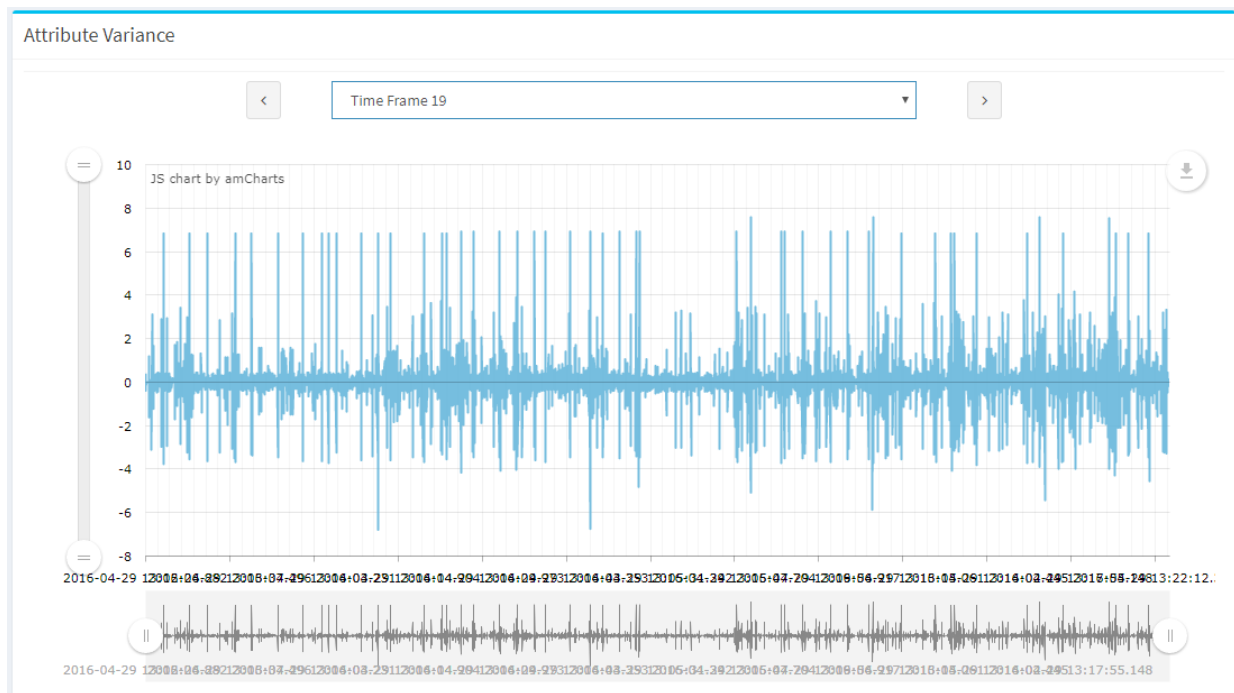


Figure 4.16 - Price gap variation display in the tool.

4.4.6. Clustering Display

As mentioned in section 4.3.1 the clustering module has developed. Highest anomaly score points will be displayed as suspicions frames. Figure 4.17 suggests that highest anomaly score is 19.06801 and there are four points in that cluster. Timeframe 7, 17, 18 and 19 are the most suspicious frames suggested by the cluster result.

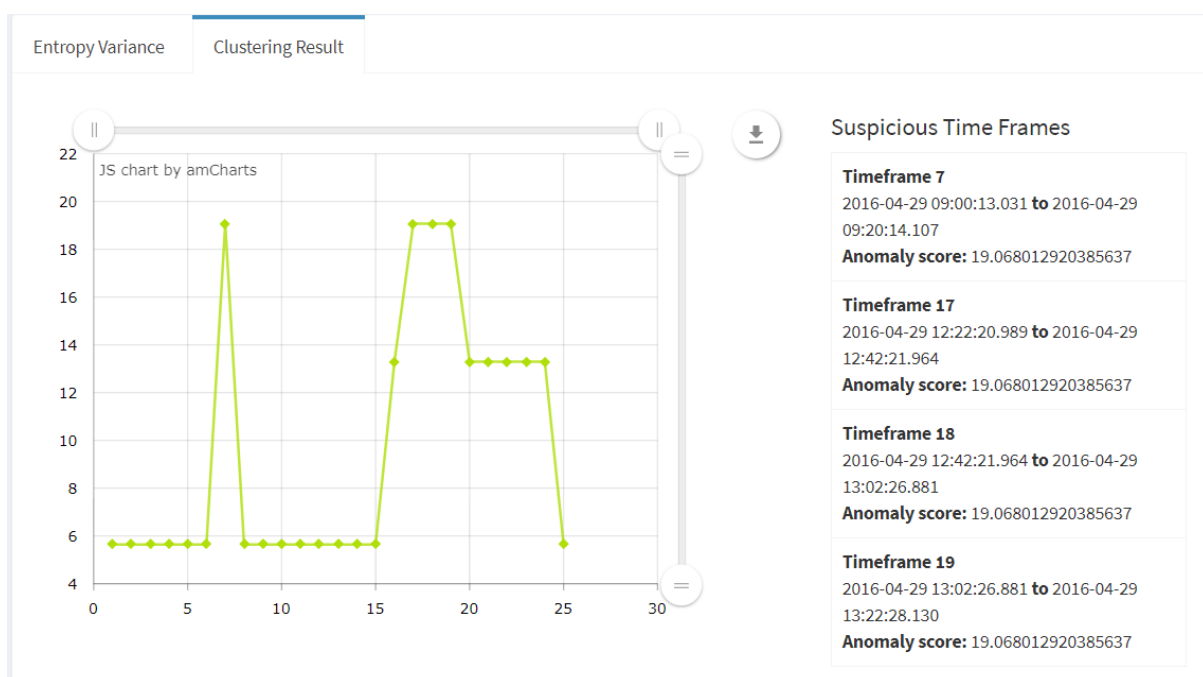
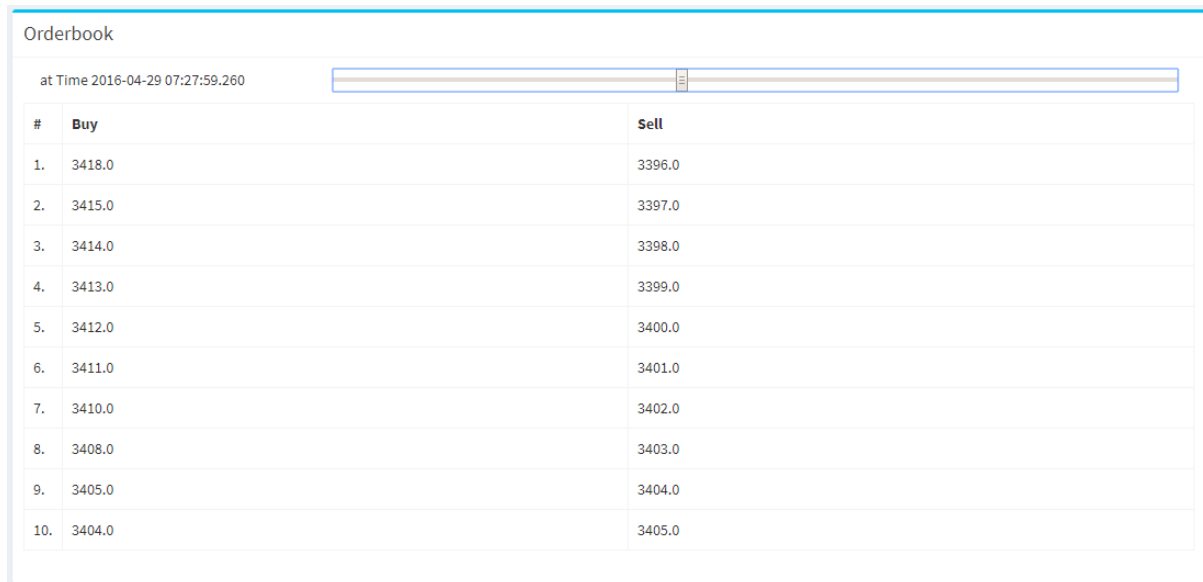


Figure 4.17 - Clustering result visualization.

4.4.7. Orderbook Simulation Display

As describes in section 4.2 dynamic orderbook has integrated to the system which shows the highest ten buy order prices and sell order prices. When a user selects a time frame, order book simulation for that will be display. There is a sliding bar as shown in Figure 4.18 to slide the orderbook in 30 seconds time gaps.



#	Buy	Sell
1.	3418.0	3396.0
2.	3415.0	3397.0
3.	3414.0	3398.0
4.	3413.0	3399.0
5.	3412.0	3400.0
6.	3411.0	3401.0
7.	3410.0	3402.0
8.	3408.0	3403.0
9.	3405.0	3404.0
10.	3404.0	3405.0

Figure 4.18 - Orderbook Simulation.

4.4.8. Frame Summary Display

When a user clicks on a point in Entropy display chart (Figure 4.15) the Figure 4.19 display will be shown. It shows the percentage of order types in that frame and a table which consists of broker identification codes in descending order of order count to make it easy for the user to identify who has done the trades in a high frequency manner.

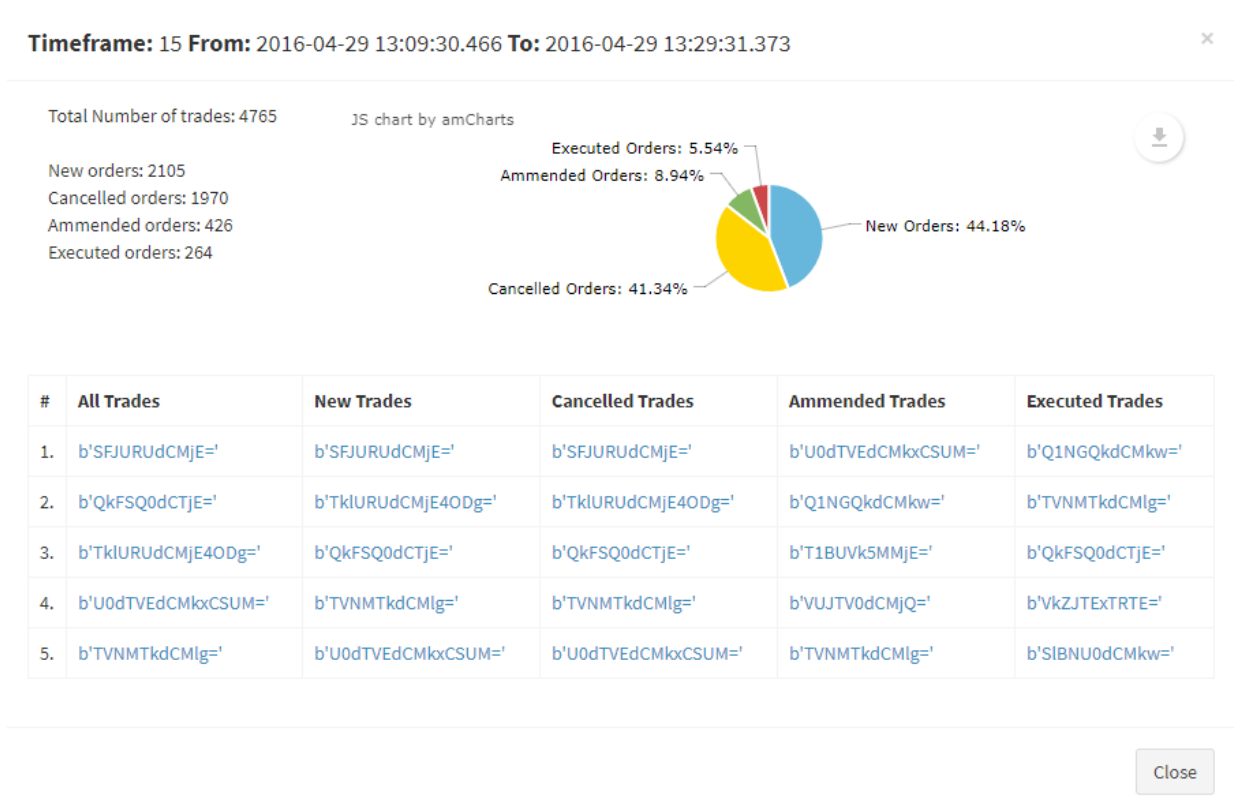


Figure 4.19 - Frame summary.

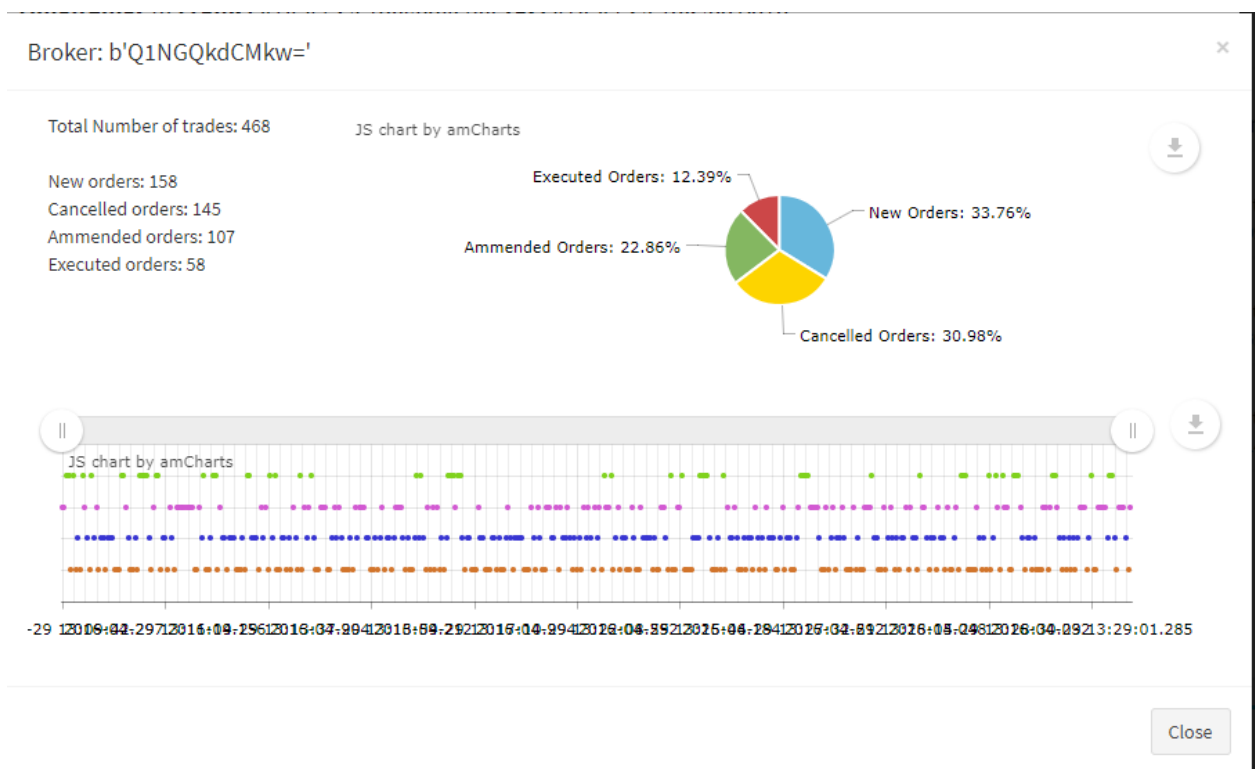


Figure 4.20 - Broker summary.

After clicking on a broker ID in Figure 4.19, Figure 4.20 will be display. It describes how the broker trade on that selected frame.



Figure 4.21 - After generating the processing result.

5. Performance Analysis

5.1.Data Preparation

Trading data set and session data set are the two main datasets we have used for data analysis. To complete the performance analysis, we had to create a synthetic dataset where we add manipulated frames to the normal trading data set.

We had datasets with Momentum Ignition and Layering manipulation behaviors.

Trading dataset consists with twelve columns. They are,

1. Instrument ID – Identification code of the trading equity.
2. Broker ID – Identification code of the broker who involves with the particular trading.
3. Executed value – Executed price value when buy and sell orders are matched.
4. Value – Price of a new order
5. Transaction time – Time of the order has placed (yyyy-mm-dd hh:mm:ss.ms)
6. Execution type – Type of the order placed (either new, cancel, amend or fill)
7. Order quantity – Volume of the order
8. Executed quantity – Volume of the order which has executed
9. Total quantity – Total volume of the order
10. Side – Whether the order is buy or sell
11. Visible size – Remaining order size
12. Order ID – Identification code of the order

Table 4.1 is an example for session file and it has following columns,

1. Transaction time – Session start or end time
2. Session name – Name of the start or ending session

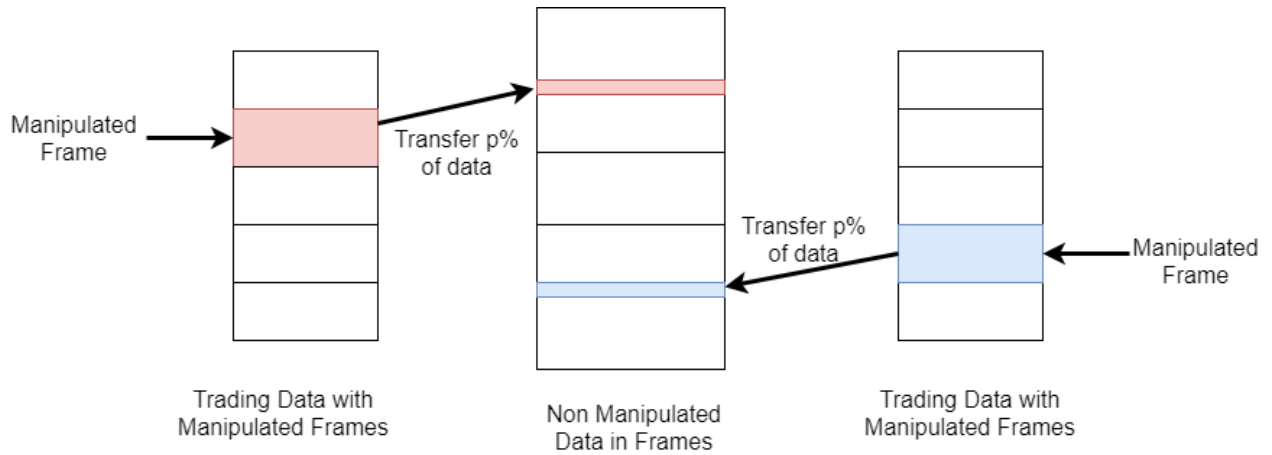


Figure 5.1 - Synthetic Data Creation.

As in Figure 5.1 shows, we have a non-manipulated trading data set and it is possible to add manipulated time frames as follows.

1. Identify different trading datasets which are having different manipulations.
2. Divide those dataset into equal sized frames.
3. Identify the frame which has the manipulation.
4. Take $p\%$ of trading messages in a random way from the manipulated frame.
5. Identify a random frame from non-manipulated data frame and replace step 4 data (except transaction time) in a random manner.
6. Repeat step 1 to step 5 until getting the synthetic data set.

5.2.Performance Measurement

Tool Performance

No of Orders	Time Window(minutes)			Event Window (order count)		
	10	15	20	1000	3000	5000
201332	28.35	28.37	33.04	22.54	24.81	31.37
134672	20.43	20.33	24.61	17.85	19.48	21.51
47345	7.61	8.96	10.21	5.31	7.38	8.21

Table 5.1 - Tool performance analysis.

Different trading data has different number of orders. Therefore, the system will respond to those inputs with different ways as shown in Table 5.1

Accuracy Measurement

All = 30	Classified: NO	Classified: YES	
Actual: NO	TN = 19	FP = 1	20
Actual: YES	FN = 3	TP = 7	10
	22	8	30

Table 5.2 - Confusion matrix for Entropy.

All = 30	Classified: NO	Classified: YES	
Actual: NO	TN = 20	FP = 0	20
Actual: YES	FN = 4	TP = 6	10
	24	6	30

Table 5.3 - Confusion matrix for Clustering.

	Precision	Recall	F Measure
Entropy	87.5%	70%	77.78%
Clustering	100%	60%	75%

Table 5.4 - Accuracy measurements.

$$Precision = \frac{TP}{TP + FP} \quad (5.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (5.2)$$

$$F\ Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5.3)$$

We have identified 30 frames and injected manipulated frames using the method describes in section 5.1. We have injected 50% of a manipulated time frame randomly to a non-manipulated time frame. Results of those data is given in the tables 5.2, 5.3 and 5.4.

Figure 5.2 shows the results obtained by testing real stock market data with the application. The real manipulation inside the data has identified manually and the momentum ignition behavior starts at 1.08pm. This dataset was processed by dividing it into timeframes of 20 minutes. The dataset is thus divided in to twenty-four timeframes.

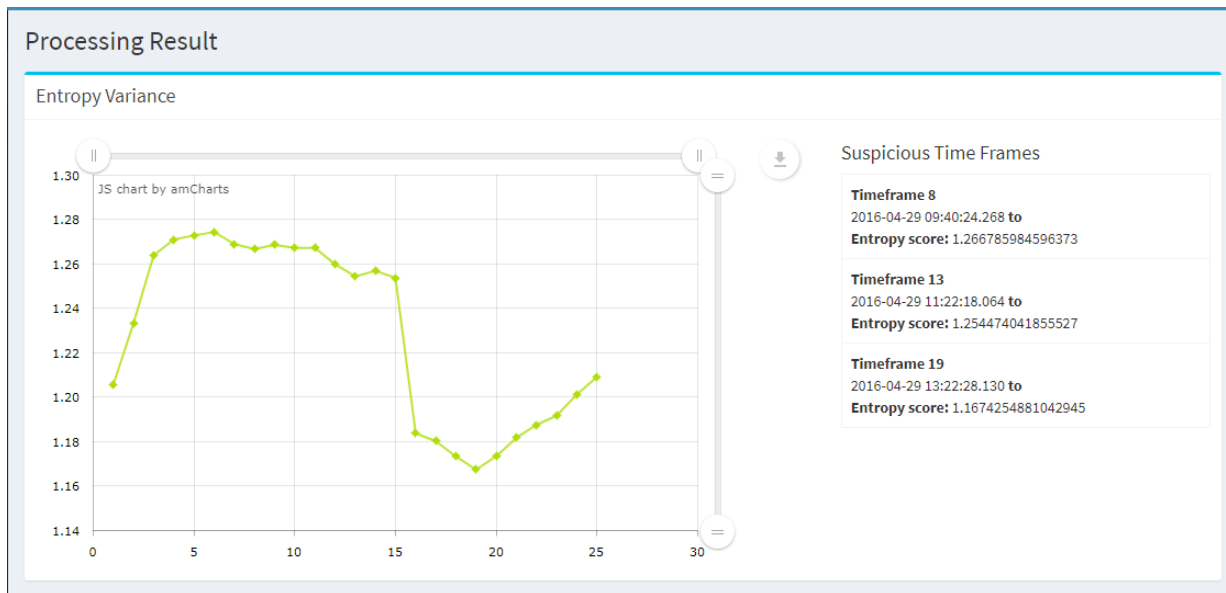


Figure 5.2 - Entropy variation of the selected dataset.

This is the entropy value graph after the processing. We can see the significant drop of the entropy value at the 19th timeframe which starts at 1.00pm and ends at 1.20pm. After analyzing the results furthermore, we can see the cause of the low entropy value in the price gap distribution chart.

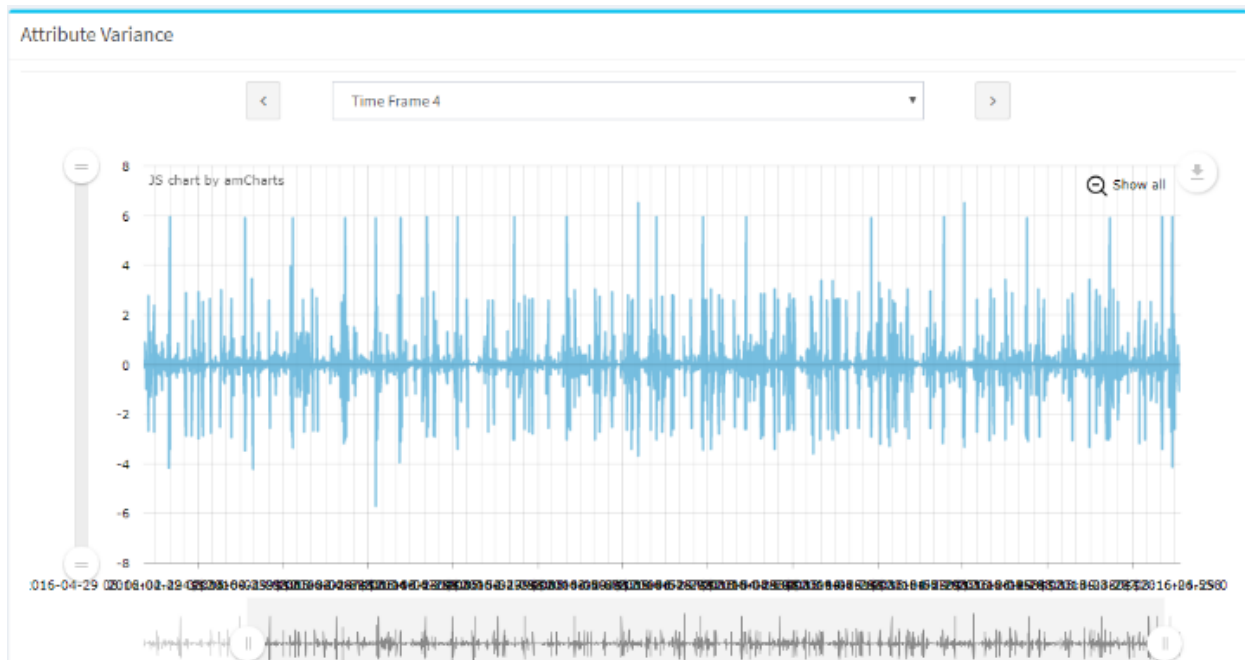


Figure 5.3 - Price gap variation of a normal time frame (Time frame 4).

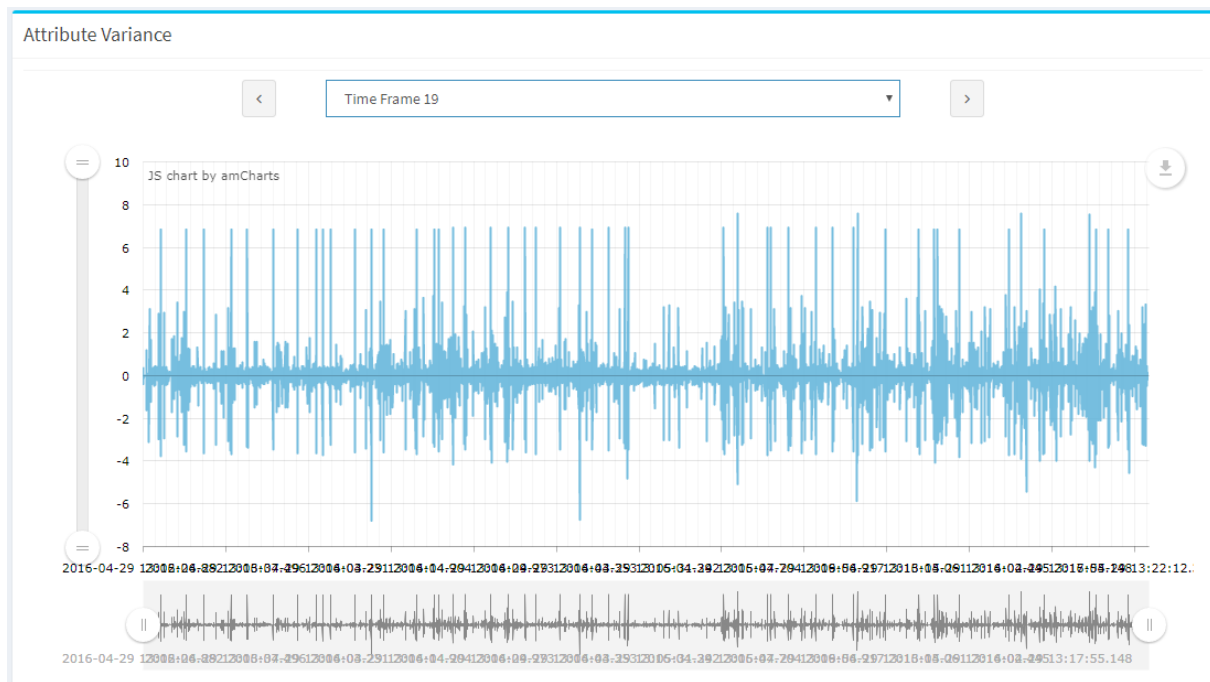


Figure 5.4 - Price gap variation of the manipulated time frame (Time frame 19).

Here we can see the difference between a normal timeframe and the manipulated timeframe. In a normal timeframe there is a uniform randomness in the price gap variation. But in the 19th timeframe we can see the significant difference in the price gap distribution. This is caused by the momentum ignition behavior at 1.08pm. With the price gap distribution chart, we can see the starting time of the behavior and the time indicated by the graph are the same.

Like this the tool identifies which timeframe has the most anomalous behavior related to the trades in the timeframe and points out that to the user.

6. Summary

6.1. Conclusion

When we look back at the incidents in the stock trading history, we can see collisions between the traders, benefitting people illegally by using market manipulations. These manipulators have gone undetected or the manipulation is not even known by anyone because there is no progressive way of detecting the manipulative behaviors in the stock markets. As the project outcome we are introducing an end-of-the-day stock market data processing technique using machine learning that can point out the exact time interval that the manipulation has taken place.

When tested on a synthetic dataset, the Entropy method was more accurate than the Clustering method. The precision was 100% in Entropy method when tested on real stock market data. All the timeframes which were detected with the Entropy method were real manipulation scenarios.

Using the proposed solution, suspicious time intervals in the stock trading can be pointed out easily to be further investigated. It is time effective because there is no need to go through the whole dataset which can be very large at sometimes. Also thinking about the financial benefits, huge losses that can be taken place because of the market manipulations can be averted. So, both the resource management benefits and the financial benefits taken through this project is valuable for the stock market domain.

6.2. Challenges

First and the most important challenge was finalizing a methodology for the solution. There were so many supervised methods to detect anomalies in a given dataset. Even for time series. But we had to use an unsupervised machine learning method because there was no labeled stock market data and manipulations were very infrequent. and the occurrence of a stock manipulation taking place was very rare.

Also, we could not use market watch data used by our supervising company because that was specific to the company and we had to design and simulate the orderbook model by ourselves. Running the simulation was costly regarding both time and the processing power.

The lack of the open source projects that can help the implementation in this kind of applications was another problem. Therefore, we had to create the structure of the project from scratch.

6.3.Future Work

There are several future works that we have identified as improvements to our research outcome and the analysis tool. As an application that runs machine learning mechanisms the performance and the accuracy of the application can be increased. For this we have to optimize the underlying algorithms and test the application against many datasets. Also, the solution can be extended to be adaptive to real time manipulation detection, where for now, it is only an end-of-the-day process of stock market data. We can improve this to match more manipulation types by adding more processing models to the system. Then we will be able to detect any anomalous behavior in any given dataset.

The data was used for processing after a preprocessing stage. There we extracted certain properties from the dataset. The more we analyze extracted properties, the more we can get results from the processing stages. So as an improvement we can extract more properties and try them in processing module to get better at producing accurate results.

There is only the time window information provided by the tool when a manipulation is detected. The tool can be improved to backtrack to the origin of the manipulation. That information might be the original trading's order ID, or the trader information, or other related trading data. This tool can only detect the several manipulation types. Hence this tool can be improve to detect other manipulation types too.

7. References

- [1] “Momentum Ignition: Arson for Financial Markets » Neurensic,” Neurensic, 10-Nov-2016. [Online]. Available: <http://neurensic.com/momentum-ignition-arson-financial-markets/>. [Accessed: 11-Nov-2017].
- [2] “Unstructured Data in a Big Data Environment,” dummies. [Online]. Available: <http://www.dummies.com/programming/big-data/engineering/unstructured-data-in-a-big-data-environment/>. [Accessed: 11-Nov-2017].
- [3] “Market Manipulation Examples & Cases | More Market Manipulation Examples | Girard Gibbs LLP,” & Cases | More Market Manipulation Examples | Girard Gibbs LLP. [Online]. Available: <https://www.girardgibbs.com/securities-fraud/stock/market-manipulation/examples/>. [Accessed: 11-Nov-2017].
- [4] C. Pirrong, “The economics of commodity market manipulation: A survey,” *Journal of Commodity Markets*, vol. 5, pp. 1–17, 2017.
- [5] ““Momentum Ignition" - The Market's Parasitic 'Stop Hunt' Phenomenon Explained,” ZeroHedge, 06-Jan-2013. [Online]. Available: <http://www.zerohedge.com/news/2012-12-14/momentum-ignition-markets-parasitic-stop-hunt-phenomenon-explained>. [Accessed: 16-Nov-2017].
- [6] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T. McGinnity, “A Hidden Markov Model with Abnormal States for Detecting Stock Price Manipulation,” 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013.
- [7] Zhai, Jia & Cao, Yi & Yao, Yuan & Ding, Xuemei & Li, Yuhua. (2016). Computational intelligent hybrid model for detecting disruptive trading activity. *Decision Support Systems*.
- [8] Z.-Q. Jiang, W.-J. Xie, X. Xiong, W. Zhang, Y.-J. Zhang, and W.-X. Zhou, “Trading networks, abnormal motifs and stock manipulation,” *Quantitative Finance Letters*, vol. 1, no. 1, pp. 1–8, 2013.
- [9] K. Golmohammadi and O. R. Zaiane, “Time series contextual anomaly detection for detecting market manipulation in stock market,” 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2015.

- [10] T. Leangarun, P. Tangamchit, and S. Thajchayapong, "Stock Price Manipulation Detection Based on Mathematical Models," *International Journal of Trade, Economics and Finance*, vol. 7, no. 3, pp. 81–88, 2016.
- [11] L. E. B. D. Silva and J. A. F. Costa, "Clustering of the self-organizing map using particle swarm optimization and validity indices," 2014 International Joint Conference on Neural Networks (IJCNN), 2014.
- [12] "Normalization (statistics)," Wikipedia, 22-Oct-2017. [Online]. Available: [https://en.wikipedia.org/wiki/Normalization_\(statistics\)](https://en.wikipedia.org/wiki/Normalization_(statistics)). [Accessed: 16-Nov-2017].
- [13] "Welcome," Welcome | Flask (A Python Microframework). [Online]. Available: <http://flask.pocoo.org/>. [Accessed: 11-Nov-2017].
- [14] SupunArunoda, "SupunArunoda/Final_Year_Project," GitHub, 16-Nov-2017. [Online]. Available: https://github.com/SupunArunoda/Final_Year_Project. [Accessed: 16-Nov-2017].