UNIVERSITY OF BARI
DEPARTMENT OF COMPUTER SCIENCE

Anomaly prediction in cryptocurrencies market: comparison between competitors

Student name: *Sergio Caputo*

Course: *Big Data* – Professor: *Michelangelo Ceci*

Contents

1. Introduction

Cryptocurrencies are digital currencies that exist on decentralized databases or ledgers and they use cryptography to secure transactions. They are not controlled by a central issuing or regulating authority in the way that conventional currencies are controlled by central banks,instead decentralized system to record transactions and issue new units are employed. There are several cryptocurrencies, and each has its own characteristics. Those with significant market capitalization are: Bitcoin, Ethereum, Litecoin, and Stellar.

Cryptocurrency trading can be very profitable given their volatility. Prices are subject to high volatility, in fact, like any other asset, the cryptocurrency market cannot continue to rise, and sometimes heavy price corrections take place. Trading in this market allows traders to make big profits due to large price fluctuations.

Cryptocurrencies have many more similarities to commodities, such as gold, than forex. Indeed:

- The performance of cryptocurrencies is not linked to the performance of a particular country's economy.
- Interest rates and monetary policies do not impact the value of cryptocurrencies.
- Investors prefer to own cryptocurrencies while waiting for them to increase in value and then convert them into traditional currencies.

Most crypto traders use technical analysis to make trading decisions and hold positions for anywhere from 30-minutes to several months in order to maximise profits. In the case of cryptocurrencies, their long-term value is based on scarcity and the fact that they have a finite or limited supply. As more people invest in a coin, or buy the coin to transact on a network, demand increases and the price rises. This means their value will not be eroded by inflation like other currencies or by dilution like stocks. In the short to medium term, the price of crypto assets is driven by sentiment about the future demand for a coin. This makes cryptocurrencies speculative investments.

In this case study, three different systems able to predict anomalous price variations of cryptocurrencies were compared to the one proposed by G.Nettis in his thesis work. For this work the most important cryptocurrencies were taken into account in order to analyze price movements useful to detect anomalies over time.

2. Anomaly Price Detection Systems in Crypto Market

In this section, all the systems considered for the comparison will be explored in depth.

2.1. Anomaly prediction in cryptocurrencies market data

The proposed system by G. Nettis in his thesis work was aimed at finding a way to accurately predict anomalous price variations in clusters of cryptocurrencies. Exploiting the correlation between different cryptocurrencies, the multi-target model was able to make better predictions considering the behavior of clusters of cryptos instead of making single predictions.

The cryptocurrencies used in this work were: Bitcoin (BTC), Bitshares (BTS), Dash (DASH), DigiByte (DGB), Dogecoin (DOGE), Ethereum (ETH), IOCoin (IOC), Litecoin (LTC), MaidSafecoin (MAID), Monacoin (MONA), Navcoin (NAV), Syscoin (SYS), Vertcoin (VTC), Counterparty (XCP), Stellar (XLM), Monero (XMR), Ripple (XRP).

2.1.1. Clustering phase

In order to group together cryptocurrencies correlated to each other, a clustering algorithm was applied to the retrieved dataset of crypto's price valuations. Then, given the clusters obtained, it was possible to build multi-target models able to predict the anomalies in a whole cluster of related cryptos.

The distance measure used to determine the separation degree of two cryptocurrencies was DTW (Dynamic time warping). Given two time series X with M points and a time series Y with N points, the idea was to find a matching between these two sequences which was called the 'warping path'.

A warping path is a correspondence between X and Y with K ordered tuples of the indices of X and Y so that (assuming 0-indexing) $W_1 = (0, 0)$, $W_K = (M - 1, N - 1)$, and $W_i - W_{i-1} \in \{(0,1),(1,0),(1,1)\}$.

In other words, matched points between time series must always stay still or advance by at most one along each, and at least one must move forward. Given a cost measure between the ith element X_i in the first time series and the jth element Y_j in the second time series, $C_{X,Y}$ (i, j), then an "optimal" or "exact" solution to the Dynamic Time Warping problem is a warping path W that minimizes the sum.

$$\sum_k C_{X,Y}(W_k^*(1)W_k^*(2))$$

We'll refer to an optimal path as W^* and the optimal cost as $D_{X,Y}$(M,N). In particular, if $D_{X,Y}$ (i, j) refers to the optimal cost of aligning the first i points of X to the first j points of Y , then the following recurrence holds for $i, j \geq 2$.

$$D_{X,Y}(i,j) = min \left\{ \begin{matrix} D_{X,Y}(i,j-1) & LEFT \\ D_{X,Y}(i-1,j) & UP \\ D_{X,Y}(i-1,j-1) & DIAG \end{matrix} \right\} + C_{X,Y}(i,j)$$

The following boundary conditions ensures that the warping path begins with the start points of both signals and terminates with their endpoint.

$$D_{X,Y}(0,j) = \sum_{k=0}^{j} C_{X,Y}(0,k)$$

$$D_{X,Y}(i,0) = \sum_{k=0}^{i} C_{X,Y}(k,0)$$

After filling in the first row and column by the boundary condition equation, it is possible to compute all values of $D_{X,Y}$ (i, j) from left to right, row by row. After processing all M N pairs of subsets in this fashion, $D_{X,Y}$ (M, N) contains the optimal cost. If we store a second matrix P (i, j) which remembers one of the three "backpointers" LEFT, RIGHT, and UP that realized the minimum cost at that cell, then we can "backtrace" by following these arrows back from (M, N) to (1, 1) to figure out the elements of an optimal W in between.

Once the cryptocurrencies distances were computed, the clusters were obtained using a k-medoids algorithm, a variant of K-means more robust to noises and outliers. Distance is computed considering the most centrally located object in the cluster instead of the mean point within it. Different configurations were experimented, indeed three values of k were used to have a comparison among the different multi-target models trained over 3, 5 and 7 clusters.

2.1.2. Multi-target LSTM network

In order to build models able to predict anomalies of multiple cryptos within the same cluster, a multi-target LSTM architecture was adopted. Taking into account all the cryptos in each cluster, they were merged together with all their features so as to obtain a dataset that was exploited to be given in input to the model with three parallel LSTM autoencoders. Three different tensors were obtained splitting the dataset with labeled sequences of 30 hours, one for each class (Stable, Up, Down). Every autoencoder was responsible for reconstructing a particular kind of row (Stable, Anomaly up, Anomaly down). After training the three autoencoders, the model was tested with a sequence of 30 hours fed to each autoencoder which reconstructed the sequence and computed the mean squared error between the input tensor and its reconstruction. The final prediction for the provided input sequence was delivered by the lowest MSE among the three autoencoders.

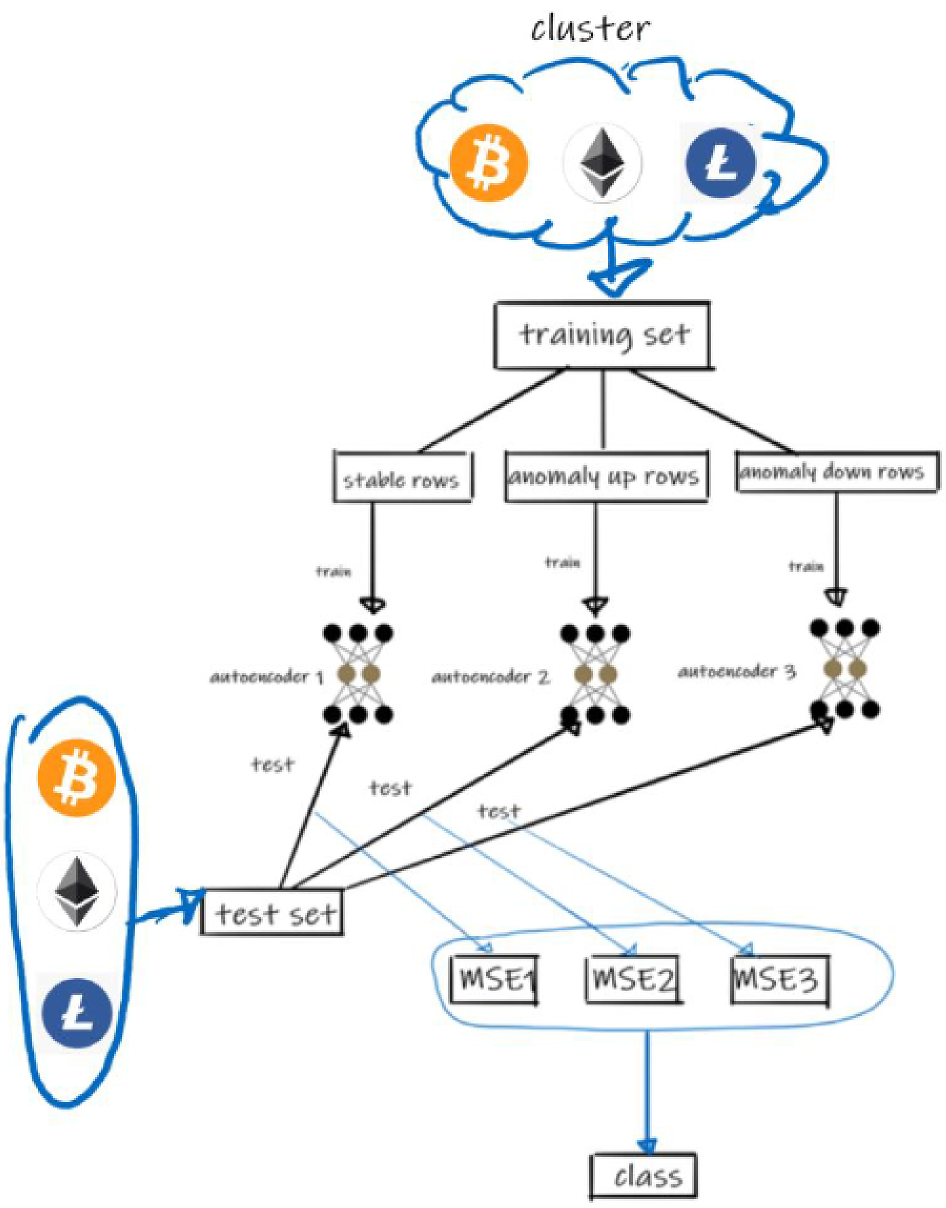


Figure 1: Multi-target LSTM Autoencoder

2.1.3. Dataset acquisition

Data for this system was collected scraping it from https://finance.yahoo.com/, price variations within a single hour were considered for the work purposes. The features available with the scraped data were:

- **Date**
- **Open Price**
- **Volume**
- **Highest price in the hour**
- **Lowest price in the hour**
- **Close price**
- **Adjusted closed price**

After data selection, also data cleaning and handling of missing values was performed.

2.1.4. Data construction

In this phase they labeled each sample in the dataset in order to define which items could be considered anomalies or not. For this scope, variations in the close price between two consequential hour were evaluated: variation of price above 1% was useful to label the previous hour as an upward anomaly, otherwise if the close price variation was below 1% then the previous hour was labeled as downward anomaly. When no variation was observed, items were labeled as stable.

Poor results were obtained labeling instances just considering price variations in one hour delta due to unbalanced dataset. This problem was solved applying the curve shifting technique of 4 hours. In this case, the samples that preceded any anomaly for 4 hours were labeled as anomalous. Thanks to this refinement the three classes were redistributed in a better way.

2.1.5. Integration with techinical analysis indicators

The dataset retrieved was integrated with different indicators commonly used in the cryptocurrencies markets.

- **SMA** (Simple Moving Average): average of the closing prices of the asset at different n periods. In Nettis'work this indicator was computed with different time steps: 5,12,13,14, 20, 21, 26, 30, 50, 100, 200 hours.

$$SMA = \frac{SUM(n_{closingprices})}{n}$$

- **EMA** (Exponential Moving Average): is a type of moving average that places a greater weight and significance on the most recent data points, it reacts more significantly to recent price changes than a SMA, which applies an equal weight to all observations in the period. The smoothing value used was 2 and EMA was computed with different time steps: 5, 12, 13, 14, 20, 21, 26, 30, 50, 100, 200 hours

$$EMA_{currenthour} = \left(Value_{currenthour} * \left(\frac{smoothing}{1 + hours} \right) \right)$$

$$+EMA_{prevhour} * \left(1 - \left(\frac{smoothing}{1 + hours} \right) \right)$$

- **MACD** (Moving Average Convergence Divergence): is a trend-following momentum indicator that shows the relationship between two moving averages of security's price. The MACD was calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA.

$$MACD = EMA(12, closeprice) - EMA(26, closeprice)$$

It is widely used for generating signals:

$$MACD_{signal} = EMA(9, MACD); MACD_{difference} = MACD - MACD_{signal}$$

- **RSI** (Relative Strenght Index): is a momentum indicator to measure the magnitude of recent price changes looking for overbought or oversold conditions in the price of the stocks or other assets. It can assume values between 0 and 100, if the RSI is greater than 80 then the asset is overbought, otherwise if it is lower than 20 an oversold situation is happening. This indicator is computed considering the average gain (U) of the upward closures in prices of the last n hours, the average loss (D) of the downward closures in prices of the last n hours. As for the previous indicators, RSI was computed for different time steps: 5,12,13,14, 20, 21, 26, 30, 50, 100, 200 hours.

$$RSI = \frac{U}{U + D} * 100$$

- **MOM** (Momentum): measures the rate of change in prices, it compares the actual price with the price of previous periods generating a curve that oscillates above and below a 0 line.

- **CMO** (Change Momentum Oscillator): sum of recent earnings and recent losses and then divides the result by the sum of all price movements in the same period. CMO oscillates in a range between -100 and + 100 and its base value is 0.

- **UO** (Ultimate Oscillator): measures the price momentum of an asset across multiple timeframes. By using the weighted average of three different timeframes the indicator has less volatility and fewer trade signals compared to other oscillators that rely on a single timeframe. Buy and sell signals are generated following divergences. The Ultimately Oscillator generates fewer divergence signals than other oscillators due to its multi-timeframe construction. The Ultimate Oscillator is a range-bound indicator with a value that fluctuates between 0 and 100. Similar to the Relative Strength Index (RSI), levels below 30 are deemed to be oversold, and levels above 70 are deemed to be overbought.

$$UO = \frac{[(A_7 * 4) + (A_{14} * 2) + A_{28}]}{7} * 100$$

where:

$$A_n = \sum_{p=1}^{n} \frac{BP}{TR}$$

is the average of n hours between Buying Pressure (BP) and True Range (TR)

$$BP = Close - Min(Low, PriorClose)$$

$$TR = Max(High, PriorClose) - Min(Low, PriorClose)$$

- **VWAP** (Volume-weighted average price): tells traders which is the average price for an asset during the day, it is based on both volume and price.

$$P_{vwap} = \frac{\sum_j P_j * Q_j}{\sum_j Q_j}$$

where:

P_{vwap} is the weighted moving average for Volume,

P_j is the price of the asset at the timestep j,

Q_j is the number of boughts assets at timestep j.

VWAP was computed with a period of 200 hours.

- **BBANDS** (Bollinger Bands): are used to analyze assets price volatility within a specific period. It is composed of 3 bands:
  - Middle Band (MB) which is the price average of the last n timestep:

$$MB = \sum \frac{n\_last\_close}{n}$$

- Upper band (UB) that adds 2 times the standard deviation:

$$UB = MB + (2 * \sigma)$$

- Lower band (UB) that subtracts 2 times the standard deviation:

$$LB = MB - (2 * \sigma)$$

Many traders believe the closer the prices move to the upper band, the more overbought the market, and the closer the prices move to the lower band, the more oversold the market.

- **STOCH** (Stochastic Oscillator): is a momentum indicator comparing a particular closing price of a security to a range of its prices over a certain period of time. The sensitivity of the oscillator to market movements is reducible by adjusting that time period or by taking a moving average of the result. It is used to generate overbought and oversold trading signals, utilizing a 0–100 bounded range of values.

  $$\%K = \left(\frac{C - L_n}{H_n - L_n} \right) * 100$$

  where:

  $$C \text{ is the most recent closing price,}$$
  $$L_n \text{ is the lowest price traded of the n previous trading sessions}$$
  $$H_n \text{ is the highest price traded of the n previous trading sessions}$$
  $$\%K \text{ is the current value of the stochastic indicator}$$

  STOCHF was computed with different timesteps: 3, 5, 14 days.

- **DPO** (Detrended Price Oscillator): strips out price trends in an effort to estimate the length of price cycles from peak to peak or trough to trough. The DPO is not a momentum indicator, It instead highlights peaks and troughs in price, which are used to estimate buy and sell points in line with the historical cycle.

2.1.6. Scaling

The obtained dataset was transformed normalizing the entries in order to make it exploitable by deep learning algorithms. The normalization strategy applied was the Min-Max Normalization, for which the range of features is scaled to range [-1,1].

2.1.7. Experiments

In order to evaluate the Multi-target LSTM model, the collected dataset was split into training and test set using a simple holdout strategy while preserving the hour's order. The model was trained on different time intervals to analyze performances over various periods:

- One month dataset: from 1st November 2021 to 1st December 2021
- Three months dataset: from 1st September 2021 to 1st December 2021
- Six months dataset: from 1st June 2021 to 1st December 2021
- Two years dataset: from 1st January 2020 to 1st December 2021

The test set was defined from 1st December 2021 to 1st January 2022.

Since hours of each day were considered, each interval started from the 00:00 of the first day until the 23:00 of the last day in the interval. Each of the previous time interval dataset (with the sole exception of the two years dataset) was used to train the k-medoid algorithm with 3 different k values (3, 5, 7). The resulting clusters obtained with the k-medoid were then used to train the Multi-target LSTM considering all the previous 4 datasets. In total, 36 different configurations were studied for the Multi-target model training.

K-medoid training		**Multi-target LSTM**
Dataset	**K**	**Dataset**
1 Month	3	FULL
1 Month	5	FULL
1 Month	7	FULL
1 Month	3	6 Months
1 Month	5	6 Months
1 Month	7	6 Months
1 Month	3	3 Months
1 Month	5	3 Months
1 Month	7	3 Months
1 Month	3	1 Month
1 Month	5	1 Month
1 Month	7	1 Month
3 Months	3	FULL
3 Months	5	FULL
3 Months	7	FULL
3 Months	3	6 Months
3 Months	5	6 Months
3 Months	7	6 Months
3 Months	3	3 Months
3 Months	5	3 Months
3 Months	7	3 Months
3 Months	3	1 Month
3 Months	5	1 Month
3 Months	7	1 Month
6 Months	3	FULL
6 Months	5	FULL
6 Months	7	FULL
6 Months	3	6 Months
6 Months	5	6 Months
6 Months	7	6 Months
6 Months	3	3 Months
6 Months	5	3 Months
6 Months	7	3 Months
6 Months	3	1 Month
6 Months	5	1 Month
6 Months	7	1 Month

2.1.8. Results

Training the LSTM with a time interval of only 1 or 3 months wasn't enough to let the model learn long term patterns in the market price variations, instead, training with a 6 months' time interval better dealt with the purpose of LSTM to learn both long and short term trends from data. For the k-medoid training, the best results were obtained with 1 month dataset and 5 clusters as k. F1 score was 0,397, Precision 0,460 and Recall 0,432 computed as average over all the crypto classifiers results.

2.2. Pump and Dumps in the Bitcoin Era: Real Time Detection of Cryptocurrency Market Manipulation

Pump and dump is a market manipulation fraud that consists in artificially inflating the price of an owned security and then selling it at a much higher price to other investors.

There are hundreds of cryptocurrencies, the market is not strictly regulated, and prices are easy to manipulate. These manipulations are performed by self-organized groups, they arrange the frauds out in the open on the Telegram instant messaging platform or Discord server, thus everyone can join the groups without prior authorization.

The approach presented in this work to detect fraud could work in real time, it was based on the abnormal growth observable in the market buy orders caused by investors that wanted to buy extremely quickly, whatever the order's price was. The pending orders for a cryptocurrency, like any other security, are listed in the order book for that cryptocurrency. The book is a sorted double list of sell (ask) and buy (bid) orders not yet filled. The asks are sorted from the lowest price to the highest, the bids are sorted from the highest to the lowest. The fastest way to buy on the market is through a buy market order. A buy market order looks up the order book and fills all the pending asks until the requested amount of currency is reached. Although a market order is completed almost instantly, the price difference between the first and the last ask needed to fill the order can be very high, especially in markets with low liquidity, and so the price can rise considerably.

2.2.1. Dataset

In order to create a pump and dump dataset for cryptocurrency market manipulation, 19 Telegram groups were analyzed. Only pump and dumps schemas carried out on Binance were investigated to create the dataset. Binance is the most popular exchange among all groups interested in market manipulations. Starting from the initial set of pumps and dumps, 900 days of trading were collected, about 104 pumps and dumps events with historical trading data for the 7 days before and after the event. The data retrieved from Binance was provided with indications about Volume, price, operation type (buy or sell), and the UNIX timestamps. Considering these, the final features set was defined with:

- **StdRushOrders** and **AvgRushOrders**: Moving standard deviation and average of volume of rush orders in each chunk of the moving window.
- **StdTrades**: Moving stardard deviation of the number of trades, both buy and sell (ask or bid).
- **StdVolumes** and **AvgVolumes**: Moving standard deviation and average of volume of trades in each chunk of the moving window.
- **StdPrice** and **AvgPrice**: Moving standard deviation and average of closing price.

- **AvgPriceMax** and **AvgPriceMin**: Moving average of maximal and minimum price in each chunk.
- **Gt**: Ground truth 0/1 value to indicate rush orders in a pump and dump schema.

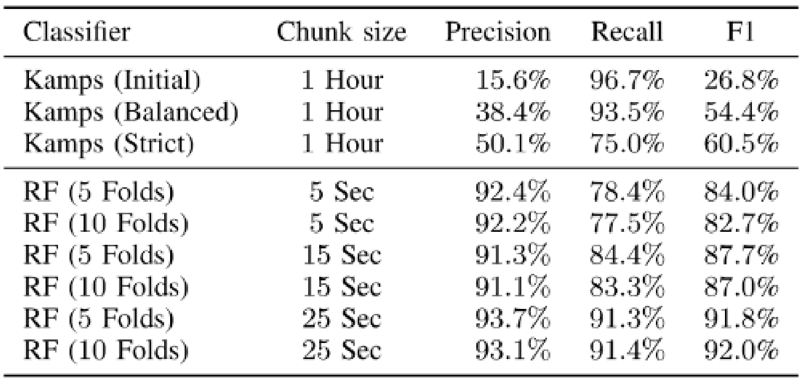
2.2.2. Experiments

Different configurations were used for the conducted experiments: data was split in chunks of s seconds and a moving window w in hours was defined (5, 15, 25 seconds for chunk size and 7 hours for window size). The algorithm used were Random Forest and Logistic Regression with 5 and 10 cross-validation since only 104 pump and dumps were available. The forest trained had 200 trees with a maximum depth of 4 for each tree.

2.2.3. Results

Since Random Forest performed slightly better than the ones based on the Logistic Regression in all the chunk size, it was the only one considered for comparison purposes. Classifier performances were compared to the ones obtained by the Kamps et al. detector. They simulate a real-time detector using as input candlesticks of 1 hour to detect pump and dumps. So, detection time can be up to 1 hour, with an expectation of 30 minutes. To detect pump and dumps their methodology exploit two anomaly thresholds one for transactions volume and the other for the coin price. They compute the values of the thresholds using average windows on the recent history of the candlestick under observation. Hence, if both the price and the volume are higher than the computed thresholds they mark the point as pump and dump event.

Comparing the two algorithms on the same dataset produced for this work, performances of the proposed detector were considerably both better (93.1% of precision and 91.4% recall against their 50.1% precision and 75.0% recall) and faster than Kamps one.



Classifier	Chunk size	Precision	Recall	F1
Kamps (Initial)	1 Hour	15.6%	96.7%	26.8%
Kamps (Balanced)	1 Hour	38.4%	93.5%	54.4%
Kamps (Strict)	1 Hour	50.1%	75.0%	60.5%
RF (5 Folds)	5 Sec	92.4%	78.4%	84.0%
RF (10 Folds)	5 Sec	92.2%	77.5%	82.7%
RF (5 Folds)	15 Sec	91.3%	84.4%	87.7%
RF (10 Folds)	15 Sec	91.1%	83.3%	87.0%
RF (5 Folds)	25 Sec	93.7%	91.3%	91.8%
RF (10 Folds)	25 Sec	93.1%	91.4%	92.0%

2.3. ELM-AD: Extreme Learning Machine Framework for Price and Volume Anomaly Detection in Stock Markets

Extreme Learning Machine (ELM) is a machine learning algorithm that is a single hidden layer feedforward neural network. This neural model has a good generalization performance with extreme learning speeds, as it doesn't have an iterative tuning method during training. In place of the regular, gradient-based propagation, it uses Moore-Penrose generalized inverse to optimize its weights during the training phase. ELMs can be applied to solve both classification and regression problems after sufficient hidden neurons and great trainable features in the chosen dataset.

ELM-AD, was a proposal for detecting price and volume manipulation in daily stock market data presented using the Extreme Learning Machine algorithm.

2.3.1. Dataset

In this paper, they explored the market manipulation cases released by The Securities and Exchange Board of India (SEBI). SEBI released the details of 16 companies whose stocks were manipulated. For every particular stock, there was a pre investigation period, investigation period and post investigation period. The day to day data of the above mentioned three periods for the 16 stocks were combined to form the dataset. For the period of manipulation, each instance was labeled 1, as well as pre and post manipulation, each instance was labelled 0. Each instance in the dataset had 20 attributes Open, High, Low and Close prices (OHLC) for that day, Weighted Average Price (WAP), number of trades that went through, number of shares that were traded, total turnover, deliverable quantity, percentage of deliverable quantity to traded quantity, spread between High-Low prices and Open-Close prices, rate of change of each of the OHLC prices and the Simple Moving.

2.3.2. System's structure

ELM model had an input layer, just one hidden layer and an output layer. The input layer had nodes N corresponding to the features in the input matrix X. There were M hidden nodes whose number was determined using hyperparameter tuning. The output layer had L nodes corresponding to the number of output classes. The weight matrix w and the bias matrix b were assigned random initial weights and biases, and the output weight matrix was calculated during training.

The first step in training ELM model, was calculating the forward activation matrix at the hidden layer using equation:

$$H = g(w * X + b)$$

where, g is the activation function, w is the weight matrix, X is the input matrix and b is the bias. Output target T could be calculated with:

$$T = H\beta$$

where *H*, is the forward activation value that acts as the input to the hidden layer, and β is the output weight matrix.

H can be represented as:

$$H = \begin{bmatrix} g\left(w_1 * x_1 + b_1\right) & \cdots & g\left(w_M * x_1 + b_M\right) \\ \vdots & \cdots & \vdots \\ g\left(w_1 * x_N + b_1\right) & \cdots & g\left(w_M * x_N + b_M\right) \end{bmatrix}_{NxM}$$

β can be represented as:

$$\beta = \begin{bmatrix} \beta_1^1 & \cdots & \beta_1^L \\ \vdots & \cdots & \vdots \\ \beta_M^1 & \cdots & \beta_M^L \end{bmatrix}_{MxL}$$

and *T* can be represented as:

$$T = \begin{bmatrix} T_1^1 & \ldots & T_1^L \\ \vdots & \ldots & \vdots \\ T_N^1 & \ldots & T_N^L \end{bmatrix}_{NxL}$$

where, N is the number of features, M is the number of hidden nodes and L is the number of output nodes.

The output weight matrix β can be derived by calculating the Moore–Penrose generalized inverse of matrix *H*.

$$\beta = H^T T$$

The Beta matrix is generated such that the cost is minimum as per equation:

$$Cost = \min_{\beta} |H\beta - T|$$

2.3.3. Results

Increasing the number of trainable nodes allowed the neural model to learn the features from the dataset for better weight optimization, for both price and volume manipulation detection models. The increasing accuracy took a drop with the number of hidden nodes at the last step using 1024 units in the hidden layer. The model was chosen with 512 hidden nodes giving a testing accuracy of 0.83 and 0.84, respectively for price and volume manipulation models. Comparing the results to other machine learning models for price and volume manipulation, ELM-AD showed better results having fast convergence, good generalization ability and fast-training speeds.

2.4. Detection of Market Manipulation using Ensemble Neural Networks

This work proposed a framework to identify stock price manipulation for the three types of manipulations: price manipulation, volume manipulation and trade reversal. An Ensemble Neural Network was created to detect whether or not manipulation occurred. The proposed Ensemble Neural Network was compared with various supervised learning algorithms such as KNN, Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Support Vector Machine and Naive Bayes Classifier. Usually, the stock market data is classified into two types, i.e. daily trade and tick data. The Ensemble Neural Network is tested on a daily trade dataset collected from the Bombay Stock Exchange (BSE).

2.4.1. Dataset

The Ensemble Neural Networks model was trained on daily stock market data, collected from the Bombay Stock Exchange (BSE) for 16 companies that were investigated for market manipulation by the Securities and Exchange Board of India (SEBI). It was the same dataset used for the previous work proposed above. Each instance had 20 attributes. For the period of manipulation, they were labelled 1, as well as pre and post manipulation, each instance was labelled 0.

2.4.2. Neural Network

For this paper, they created a hybrid stacking Ensemble Neural Network to predict whether a particular stock had been manipulated or not. Stacking, also known as stacked generalization, is an ensemble technique that is capable of combining the output of various classification models. Multiple models are combined and each model contributes equally to the ensemble regardless of the performance of a single model. Each model is trained on its own weights and then they are combined through a meta-classifier, in this way the obtained final model has better results in predictions with much higher accuracy.

Each sub-model had 20 input nodes in the input layer, each corresponding to one attribute and the activation function used was the Rectified Linear Unit. The hidden layer consisted of 6 layers, each provided with 20 nodes and ReLU as activation function. In the end, the output layer consisted of one node with the sigmoid activation function and its output depicted two binary values, 0 or 1. If manipulation occurred, it was 1, otherwise 0. The loss function used by the sub-models was the Binary Cross Entropy.

Creating an ensemble of these artificial neural networks, it was possible to achieve a boost in performance and accuracy of the system. A total of five sub-models were created, these sub-models which are Neural Networks acted as a meta learner. The sub-models could be embedded into a more extensive multi-headed ensembled Artificial Neural Network; this Ensemble Neural Network had the capabilities to learn how to combine the outputs of every sub-model efficiently. This allowed the stacked gen-

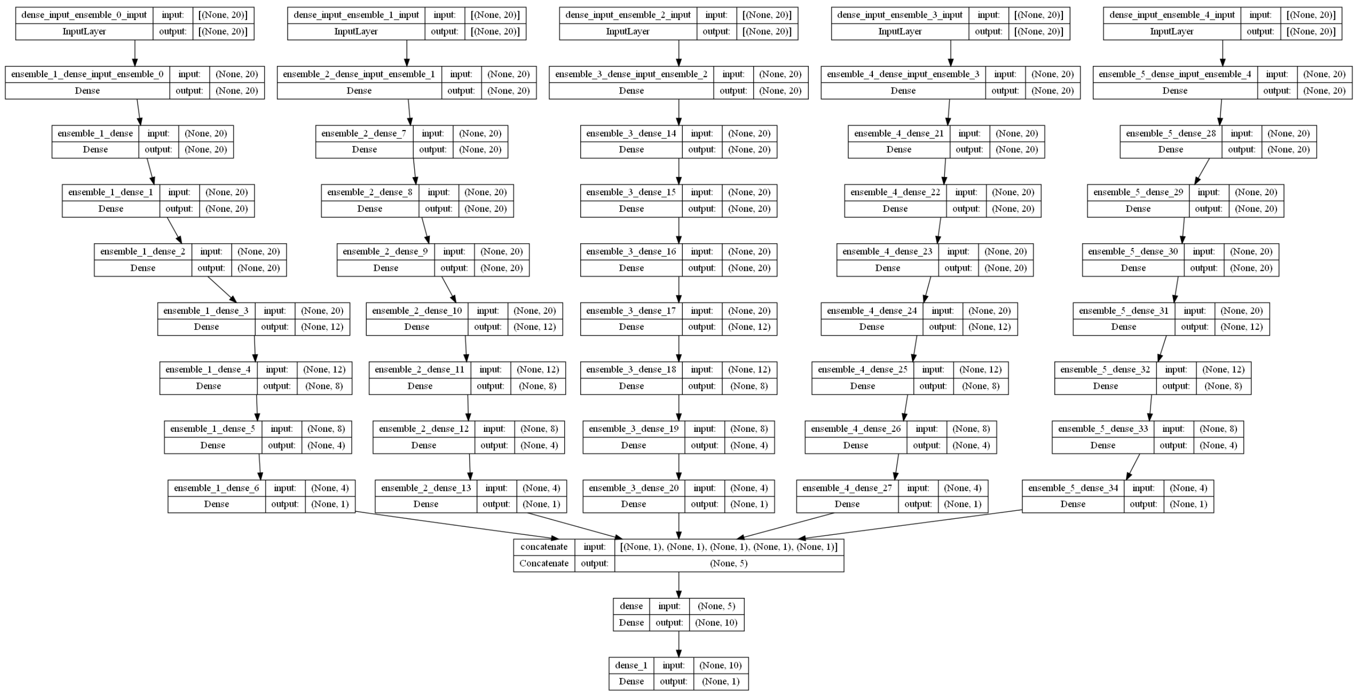

dense_input_ensemble_0_input
dense_input_ensemble_1_input
dense_input_ensemble_2_input
dense_input_ensemble_3_input
dense_input_ensemble_4_input
concatenate
Concatenate
dense
dense_1

Figure 2: Entire Ensemble Network with sub-models

eralization to be treated as a single large model. The output of every Neural Network sub-model was used as the input to the stacking ensemble model; all the outputs of the sub-models were merged into a single vector which acted as the input for the Ensemble Neural Network. Its hidden layer was made of 10 nodes with ReLU, while the output layer had one node with the sigmoid function. Also for this network the loss function was the Binary Cross Entropy.

2.4.3. Results

In order to analyze model performance with different configurations, two different training approaches were used:

- Sub-models layers set as non-trainable: all sub-models trained first and optimized individually. Once this process was done, the output layer of each model was concatenated to a single vector that was used for training in the level 1 Neural Network after optimization of level 0 Neural Network (5 sub-models).
- Sub-models layers set as trainable: level 0 and 1 ensemble models were trained together.

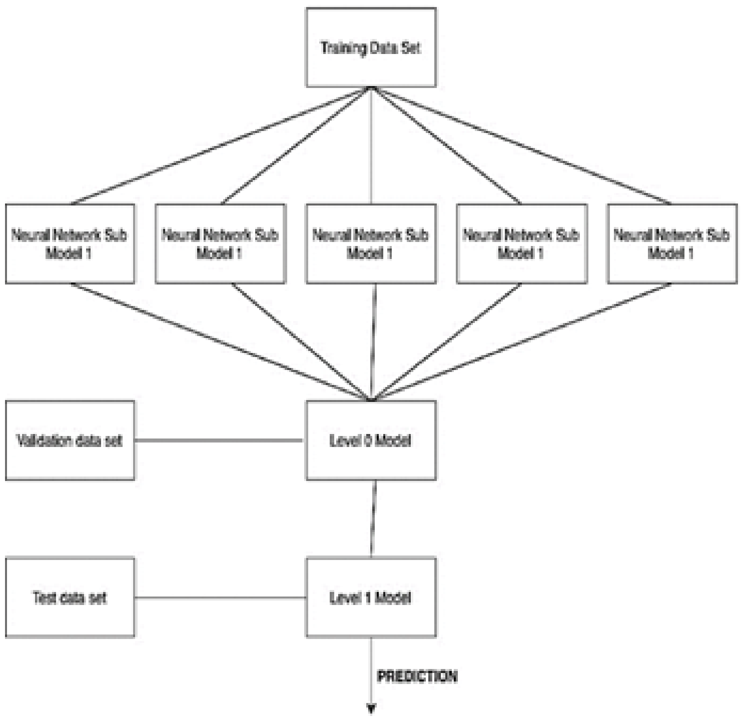

Training Data Set
Neural Network Sub Model 1
Neural Network Sub Model 1
Neural Network Sub Model 1
Neural Network Sub Model 1
Neural Network Sub Model 1
Validation data set
Level 0 Model
Test data set
Level 1 Model
PREDICTION

Figure 3: Stacking Architecture

The stacked model without trainable layers had the highest accuracy of 96%, whereas the model with trainable layers showed an accuracy of 91%. Both the models outperformed all the other supervised learning models such as K Nearest Neighbor (KNN), Logistic Regression, Naive Bayes Classifier, Support Vector Machine (SVM), Quadratic Discriminant Analysis and Linear Discriminant Analysis.

3. Comparison between systems

Since we were performing a comparison between different systems and the one proposed by Nettis, the same collected dataset introduced above was used for the other three systems that will be discussed in the next sections of this report. The original training and test set was common to all 3 systems.

- One month dataset: from 1st November 2021 to 1st December 2021
- Three months dataset: from 1st September 2021 to 1st December 2021
- Six months dataset: from 1st June 2021 to 1st December 2021
- Two years dataset: from 1st January 2020 to 1st December 2021

The test set was defined from 1st December 2021 to 1st January 2022.

Considering the systems proposed in these papers, our reference dataset was adapted in order to evaluate the three approaches performance by exploiting Nettis' dataset.

3.1. Pump and Dumps in the Bitcoin Era: Real Time Detection of Cryptocurrency Market Manipulation

3.1.1. Features adaptation

Taking into account the features set of this system, the original dataset was adapted to be usable by the Pump and Dump proposed approach. The features used and computed were:

- Date: date and hour of the crypto's recorded information
- Std_Rush_Orders and Avg_Rush_Orders: Moving standard deviation and average of volume of rush orders in each chunk of the moving window
- Std_Trades: Moving standard deviation of the number of trades, both buy and sell
- Std_Volumes and Avg_Volumes: Moving standard deviation and average of volume of trades in each chunk of the moving window
- Std_Price and Avg_Price: Moving standard deviation and average of closing price
- Avg_Price_Max and Avg_Price_Min: Moving average of maximal and minimum price in each chunk
- Hour_Sin: sine of the hour of logged information
- Hour_Cos: cosine of the hour of logged information
- Gt: ground truth 0/1 value to indicate rush orders in a pump and dump schema

In order to be aligned with the Nettis' results, chunks size was chosen equal to 4 hours since in Nettis' work curve shifting of 4 hours was applied in construction phase to obtain better and more reliable results. Whereas, the window size for rolling operations was set to 30, it was considered to have a comparison with the system of Nettis since he used sequences of 30 hours as input for the LSTM autoencoders.

The computation of the labels in the ground truth was performed with an algorithm of majority voting, every chunk of entries grouped with a time frequency of 4 hours was labeled with a single label that corresponded to the most frequent label in the chunk. The number of labels was increased to 3 to consider up anomalies, down anomalies and stable situations.

3.1.2. Changes performed

Differently from the original features set used in the Pump and Dump paper, we had to exclude the moving standard deviation and the average of volume of rush orders in each chunk of the moving window. This choice was dictated by the absence of rush orders, in other words our dataset was not provided of multiple entries associated to same timestamps, and so these two measures could not be computed. In our dataset each entry was related to a single hour.

Following the approach of Pump and Dump system, the new obtained dataset was fed to the classifiers Random Forest and Logistic Regression. More specifically, here we considered a Multinomial Logistic Regression, an extension to the logistic regression model that predicts a multinomial probability (i.e. more than two classes) for each input example to support multi-class classification problems. For Multinomial Logistic Regression, LBFGS solver was used with a regularization strength of C equals to 1.

3.2. ELM-AD: Extreme Learning Machine Framework for Price and Volume Anomaly Detection in Stock Markets

With respect to the ELM-AD structure and features, our dataset was conformed in order to be tested with the proposed methodology.

3.2.1. Feature adaption

The features used and computed were:

- Date: date and hour of the crypto's recorded information
- Open Price: price of the index at the start of the day
- High Price: highest price of the index in the day
- Low Price: lowest price of the index in the day
- Close Price: price of the index at the end of the day
- WAP (Weighted Average Price): average price of the index over a day
- No. of Shares: number of stocks traded in a day
- No. of Trades: number of transactions placed in a day
- Total Turnover: price of shares * number of shares traded
- Deliverable Quantity: quantity of stocks traded that have been transferred from one account to another
- % Deliverable Quantity to Traded Quantity: ratio of Number of Shares that have been transferred to the Total Number of Trades that have been placed
- Spread High-Low: Difference between High-Low Price
- Spread Close-Open: Difference between Close-Open Price
- Ch(t) Open Price: Rate of Change of Open Price between two successive days
- SMA Open Price: Average Rate of Change of Open Price for n days
- Ch(t) High Price: Rate of Change of High Price between two successive days
- SMA High Price: Average Rate of Change of High Price for n days
- Ch(t) Low Price: Rate of Change of Low Price between two successive days
- SMA Low Price: Average Rate of Change of Low Price for n days
- Ch(t) Close Price: Rate of Change of Close Price between two successive days
- SMA Close Price: Average Rate of Change of Close Price for n days

- Label: 1 for manipulation period and 0 for pre and post manipulation

The dataset provided for this system contained day to day trading data, for this reason we had to perform some changes in order to consider the Nettis' dataset based on hour by hour dataset:

- Open Price, High Price, Low Price, Close Price, Spread High-Low, Spread Close-Open: these were considered for every hour
- WAP: it was computed over a set of 24 entries (one day)
- SMA Open Price, SMA High Price, SMA Low Price, SMA Close Price: these were computed over 30 entries (as Nettis 30 hours sequences fed to the LSTM autoencoders)
- Ch(t) Open Price, Ch(t) High Price, Ch(t) Low Price, Ch(t) Close Price: these were computed over 2 successive hours

Also in this case, the number of labels was changed to 3 to consider up anomalies, down anomalies and stable situations.

3.2.2. *Changes performed*

Unlike the original set of features used in ELM-AD, the features set for our dataset was not provided of enough information to compute or deduct features like: No. of Shares, No. of Trades, Total Turnover, Deliverable Quantity and % Deliverable Quantity to Traded Quantity. For this reason two approaches were used to train the ELM-AD model:

- All features dataset: the whole set of features used in this paper was maintained, but the previous 5 features were set to 0 for all the entries.
- Reduced features dataset: the incalculable set of 5 features was removed. Volume feature available in the original dataset was added to the new features set.

For the case analyzed here, the ELM-AD structure was modified in order to be suitable for our 3 labels dataset. The Sigmoid activation function was modified to the Softmax since we moved from a binary classification model to a multi-class classification model.

3.2.3. *Pre-processing*

Complying with the decisions made by the authors of this paper, our dataset was normalized using Min-Max Scaler. It helps to reduce values of data in a range [0,1], this can improve the convergence speed during the learning phase.

$$X = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3.3. Detection of Market Manipulation using Ensemble Neural Networks

The available dataset constructed by Nettis was adapted to be suitable for the Ensemble Neural Network.

3.3.1. Feature adaption

Since the original dataset and feature set for this model were the same used in ELM-AD, no additional changes with respect to the previous section were made.

The only aspect that was not changed from the proposed system, was related to the number of labels in the dataset. In Nettis'dataset there were three labels to indicate up anomalies, down anomalies and stable situations; for this work the number of labels in their dataset was fixed to two: 1 for manipulation period and 0 for pre and post manipulation.

According to these two labels they designed sub-models and the actual stacked neural network with just one node in the output layer with the sigmoid activation function to perform a binary classification. Considering the proposed model configuration designed to be a binary classifier, we decided to not modify the structure of the network in order to avoid an inconsistent comparison for a system not designed to be a multi class classifier for three labels. Classification among two classes is an easier task than classification with three, so the system could easily reach better performance than Nettis'one.

In any case, just to analyze this system's performance, Nettis' dataset with three labels was adapted and labels cardinality was lowered to two: stable situations were labeled to 0, up and down anomalies were considered as manipulations labeled to 1.

3.3.2. Changes performed

For the same motivations previously provided for ELM-AD, also for this system two approaches based on different features set were applied.

3.3.3. Pre-processing

As for ELM-AD, data was normalized with the Min-Max Scaler.

4. Results

Considering systems trained on equal datasets, the results of the various models were compared with those produced by the configurations used by Nettis that obtained greater improvement between LSTM with clustering and LSTM without clustering.

4.1. One month training

Results obtained with 1 month training compared with Nettis best configuration for 1 month dataset (k means training with 6 months and k = 7).

Precision

Crypto	Nettis System 1m, k=7, 6m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,367	0,636	0,183	0,361	0,332
BTS	0,397	0,365	0,34	0,309	0,317
DASH	0,539	0,66	0,231	0,315	0,32
DGB	0,365	0,529	0,277	0,347	0,336
DOGE	0,391	0,482	0,34	0,333	0,328
ETH	0,396	0,587	0,313	0,317	0,34
IOC	0,396	0,411	0,366	0,348	0,343
LTC	0,483	0,434	0,289	0,337	0,327
MAID	0,243	0,507	0,281	0,321	0,339
MONA	0,332	0,54	0,32	0,331	0,346
NAV	0,479	0,341	0,204	0,348	0,351
SYS	0,272	0,395	0,183	0,325	0,311
VTC	0,056	0,29	0,125	0,349	0,335
XCP	0,287	0,555	0,31	0,351	0,339
XLM	0,278	0,311	0,363	0,283	0,305
XMR	0,337	0,466	0,425	0,318	0,339
XRP	0,395	0,519	0,396	0,324	0,319

Recall

Crypto	Nettis System 1m, k=7, 6m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,365	0,499	0,33	0,367	0,33
BTS	0,352	0,389	0,335	0,303	0,315
DASH	0,489	0,428	0,322	0,314	0,317
DGB	0,366	0,394	0,334	0,349	0,334
DOGE	0,394	0,479	0,341	0,333	0,327
ETH	0,387	0,493	0,338	0,328	0,352
IOC	0,384	0,386	0,339	0,351	0,345
LTC	0,361	0,438	0,396	0,341	0,332
MAID	0,252	0,52	0,349	0,331	0,349
MONA	0,352	0,54	0,316	0,331	0,345
NAV	0,476	0,429	0,316	0,344	0,35
SYS	0,330	0,387	0,33	0,296	0,289
VTC	0,333	0,328	0,333	0,309	0,276
XCP	0,383	0,401	0,337	0,352	0,343
XLM	0,303	0,315	0,363	0,283	0,306
XMR	0,300	0,463	0,434	0,326	0,347
XRP	0,397	0,503	0,351	0,327	0,32

F1

Crypto	Nettis System 1m, k=7, 6m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,261	0,513	0,235	0,359	0,328
BTS	0,355	0,356	0,237	0,3	0,311
DASH	0,440	0,373	0,254	0,312	0,316
DGB	0,360	0,367	0,302	0,342	0,327
DOGE	0,314	0,478	0,339	0,331	0,321
ETH	0,328	0,497	0,219	0,318	0,341
IOC	0,366	0,378	0,314	0,349	0,343
LTC	0,349	0,378	0,334	0,338	0,329
MAID	0,221	0,501	0,31	0,312	0,333
MONA	0,331	0,538	0,274	0,314	0,332
NAV	0,477	0,363	0,177	0,344	0,348
SYS	0,178	0,379	0,235	0,297	0,29
VTC	0,096	0,195	0,182	0,27	0,256
XCP	0,227	0,378	0,262	0,314	0,297
XLM	0,214	0,265	0,309	0,283	0,3
XMR	0,284	0,454	0,42	0,301	0,321
XRP	0,388	0,486	0,248	0,323	0,318

4.2. Three months training

Results obtained with 3 months training compared with Nettis best configuration for 3 months dataset (k means training with 3 months and k = 3).

Precision

Crypto	**Nettis System 3m, k=3, 3m**	**Pump and Dump**		**ELM-AD**	
		Random Forest	**Multinomial Logistic Regression**	**All features**	**Reduced features**
BTC	0,382	0,543	0,518	0,315	0,315
BTS	0,430	0,388	0,173	0,358	0,352
DASH	0,446	0,686	0,145	0,225	0,342
DGB	0,399	0,673	0,251	0,326	0,263
DOGE	0,500	0,457	0,432	0,25	0,303
ETH	0,450	0,527	0,291	0,298	0,268
IOC	0,400	0,454	0,246	0,264	0,328
LTC	0,459	0,49	0,266	0,342	0,347
MAID	0,285	0,496	0,287	0,371	0,296
MONA	0,345	0,596	0,493	0,291	0,273
NAV	0,415	0,339	0,247	0,349	0,329
SYS	0,342	0,328	0,35	0,314	0,43
VTC	0,350	0,29	0,46	0,343	0,348
XCP	0,338	0,324	0,532	0,368	0,349
XLM	0,434	0,403	0,253	0,355	0,375
XMR	0,430	0,444	0,271	0,308	0,323
XRP	0,423	0,514	0,348	0,336	0,374

Recall

Crypto	**Nettis System 3m, k=3, 3m**	**Pump and Dump**		**ELM-AD**	
		Random Forest	**Multinomial Logistic Regression**	**All features**	**Reduced features**
BTC	0,289	0,448	0,34	0,334	0,327
BTS	0,472	0,407	0,333	0,335	0,319
DASH	0,423	0,469	0,329	0,315	0,327
DGB	0,384	0,4	0,332	0,324	0,32
DOGE	0,485	0,459	0,336	0,296	0,297
ETH	0,362	0,501	0,352	0,33	0,315
IOC	0,325	0,416	0,31	0,315	0,32
LTC	0,465	0,463	0,363	0,347	0,352
MAID	0,309	0,517	0,346	0,326	0,318
MONA	0,360	0,476	0,342	0,332	0,327
NAV	0,423	0,446	0,323	0,331	0,336
SYS	0,444	0,333	0,334	0,331	0,334
VTC	0,359	0,326	0,342	0,308	0,3
XCP	0,312	0,35	0,34	0,315	0,319
XLM	0,430	0,393	0,34	0,347	0,351
XMR	0,394	0,434	0,332	0,342	0,341
XRP	0,409	0,514	0,336	0,324	0,333

F1

Crypto	Nettis System 3m, k=3, 3m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,212	0,447	0,251	0,238	0,232
BTS	0,422	0,371	0,228	0,196	0,188
DASH	0,386	0,414	0,201	0,17	0,181
DGB	0,379	0,369	0,232	0,186	0,184
DOGE	0,465	0,448	0,233	0,157	0,167
ETH	0,273	0,504	0,286	0,166	0,159
IOC	0,265	0,404	0,185	0,175	0,178
LTC	0,461	0,406	0,304	0,176	0,178
MAID	0,202	0,485	0,312	0,225	0,203
MONA	0,340	0,467	0,232	0,244	0,236
NAV	0,416	0,382	0,277	0,212	0,211
SYS	0,264	0,315	0,244	0,237	0,245
VTC	0,313	0,199	0,201	0,193	0,188
XCP	0,297	0,295	0,263	0,162	0,158
XLM	0,431	0,38	0,29	0,177	0,187
XMR	0,394	0,429	0,226	0,222	0,23
XRP	0,408	0,512	0,328	0,199	0,219

4.3. Six months training

Results obtained with 6 months training compared with Nettis best configuration for 6 months dataset (k means training with 1 months and k = 5).

Precision

Crypto	Nettis System 6m, k=5, 1m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,431	0,592	0,852	0,231	0,268
BTS	0,478	0,359	0,34	0,105	0,356
DASH	0,541	0,315	0,145	0,355	0,463
DGB	0,416	0,339	0,669	0,126	0,122
DOGE	0,525	0,427	0,49	0,093	0,094
ETH	0,469	0,484	0,425	0,362	0,186
IOC	0,442	0,359	0,258	0,287	0,327
LTC	0,478	0,383	0,307	0,112	0,345
MAID	0,385	0,481	0,307	0,302	0,269
MONA	0,400	0,507	0,159	0,209	0,374
NAV	0,499	0,328	0,286	0,574	0,212
SYS	0,560	0,323	0,183	0,179	0,512
VTC	0,319	0,342	0,364	0,456	0,346
XCP	0,544	0,385	0,532	0,08	0,107
XLM	0,410	0,448	0,269	0,42	0,087
XMR	0,474	0,315	0,16	0,449	0,16
XRP	0,451	0,506	0,394	0,268	0,225

Recall

Crypto	Nettis System 6m, k=5, 1m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,438	0,468	0,349	0,328	0,326
BTS	0,457	0,405	0,335	0,33	0,333
DASH	0,533	0,414	0,329	0,331	0,336
DGB	0,418	0,376	0,353	0,331	0,324
DOGE	0,523	0,438	0,374	0,319	0,322
ETH	0,462	0,476	0,336	0,333	0,331
IOC	0,375	0,33	0,324	0,321	0,326
LTC	0,345	0,417	0,43	0,335	0,34
MAID	0,376	0,487	0,334	0,326	0,324
MONA	0,421	0,455	0,333	0,334	0,335
NAV	0,468	0,434	0,377	0,335	0,332
SYS	0,478	0,327	0,33	0,332	0,333
VTC	0,365	0,349	0,339	0,315	0,318
XCP	0,368	0,378	0,34	0,322	0,317
XLM	0,391	0,39	0,339	0,336	0,333
XMR	0,478	0,404	0,333	0,338	0,339
XRP	0,446	0,492	0,349	0,338	0,334

F1

Crypto	Nettis System 6m, k=5, 1m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,383	0,483	0,27	0,123	0,124
BTS	0,422	0,374	0,237	0,16	0,166
DASH	0,534	0,357	0,201	0,165	0,171
DGB	0,378	0,331	0,272	0,166	0,163
DOGE	0,507	0,398	0,319	0,145	0,146
ETH	0,445	0,475	0,265	0,117	0,112
IOC	0,336	0,291	0,202	0,15	0,15
LTC	0,290	0,364	0,355	0,136	0,144
MAID	0,354	0,481	0,214	0,182	0,179
MONA	0,395	0,453	0,215	0,125	0,131
NAV	0,459	0,373	0,32	0,192	0,188
SYS	0,374	0,325	0,235	0,233	0,234
VTC	0,254	0,313	0,21	0,178	0,181
XCP	0,319	0,338	0,263	0,128	0,131
XLM	0,387	0,364	0,236	0,144	0,138
XMR	0,475	0,354	0,217	0,165	0,161
XRP	0,441	0,49	0,248	0,172	0,168

4.4. Two years training

Results obtained with 2 years training compared with Nettis best configuration for 2 years dataset (k means training with 1 months and k = 5).

Precision

Crypto	Nettis System 2y, k=5, 1m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,431	0,546	0,852	0,065	0,065
BTS	0,478	0,451	0,34	0,105	0,105
DASH	0,541	0,49	0,145	0,104	0,104
DGB	0,416	0,66	0,669	0,106	0,106
DOGE	0,525	0,556	0,49	0,096	0,095
ETH	0,469	0,536	0,425	0,064	0,064
IOC	0,442	0,198	0,258	0,084	0,084
LTC	0,478	0,497	0,307	0,08	0,08
MAID	0,385	0,494	0,307	0,114	0,114
MONA	0,400	0,578	0,159	0,07	0,403
NAV	0,499	0,36	0,286	0,129	0,129
SYS	0,560	0,367	0,183	0,177	0,177
VTC	0,319	0,33	0,364	0,124	0,124
XCP	0,544	0,455	0,532	0,146	0,082
XLM	0,410	0,395	0,269	0,086	0,086
XMR	0,474	0,492	0,16	0,096	0,096
XRP	0,451	0,603	0,394	0,313	0,211

Recall

Crypto	Nettis System 2y, k=5, 1m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,438	0,444	0,349	0,333	0,333
BTS	0,457	0,442	0,335	0,333	0,332
DASH	0,533	0,486	0,329	0,33	0,332
DGB	0,418	0,408	0,353	0,332	0,333
DOGE	0,523	0,516	0,374	0,333	0,332
ETH	0,462	0,492	0,336	0,331	0,331
IOC	0,375	0,315	0,324	0,333	0,333
LTC	0,345	0,472	0,43	0,333	0,333
MAID	0,376	0,506	0,334	0,333	0,333
MONA	0,421	0,528	0,333	0,333	0,334
NAV	0,468	0,474	0,377	0,333	0,333
SYS	0,478	0,371	0,33	0,333	0,333
VTC	0,365	0,334	0,339	0,333	0,332
XCP	0,368	0,463	0,34	0,322	0,332
XLM	0,391	0,392	0,339	0,333	0,333
XMR	0,478	0,46	0,333	0,333	0,333
XRP	0,446	0,558	0,349	0,34	0,333

F1

Crypto	Nettis System 2y, k=5, 1m	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
BTC	0,383	0,443	0,27	0,109	0,109
BTS	0,422	0,441	0,237	0,16	0,16
DASH	0,534	0,455	0,201	0,158	0,158
DGB	0,378	0,378	0,272	0,161	0,161
DOGE	0,507	0,49	0,319	0,149	0,148
ETH	0,445	0,495	0,265	0,107	0,107
IOC	0,336	0,228	0,202	0,134	0,134
LTC	0,290	0,448	0,355	0,129	0,129
MAID	0,354	0,483	0,214	0,17	0,17
MONA	0,395	0,534	0,215	0,115	0,117
NAV	0,459	0,407	0,32	0,186	0,187
SYS	0,374	0,362	0,235	0,232	0,232
VTC	0,254	0,267	0,21	0,181	0,18
XCP	0,319	0,407	0,263	0,139	0,131
XLM	0,387	0,39	0,236	0,137	0,137
XMR	0,475	0,442	0,217	0,149	0,149
XRP	0,441	0,55	0,248	0,172	0,161

4.5. Considerations

Confronting the different results obtained with the best configurations tested by Nettis and those obtained from the systems discussed in this paper, the best models could be determined depending on the dataset used. In the following summary the general situation can be observed easily with measurement averages:

	Nettis System	Pump and Dump		ELM-AD	
		Random Forest	Multinomial Logistic Regression	All features	Reduced features
	1m, k=7, 6m	**One month**			
Precision	0,354	*0,472*	0,291	0,330	0,331
Recall	0,366	*0,435*	0,345	0,329	0,328
F1	0,305	*0,406*	0,274	0,318	0,318
	3m, k=3, 3m	**Three months**			
Precision	0,402	*0,468*	0,327	0,318	0,330
Recall	0,391	*0,432*	0,337	0,327	0,326
F1	0,349	*0,402*	0,253	0,196	0,197
	6m, k=5, 1m	**Six months**			
Precision	*0,460*	0,405	0,361	0,271	0,262
Recall	*0,432*	0,414	0,347	0,330	0,330
F1	*0,397*	0,386	0,252	0,158	0,158
	2y, k=5, 1m	**Two years**			
Precision	0,460	*0,471*	0,361	0,115	0,125
Recall	0,432	*0,451*	0,347	0,332	0,333
F1	0,397	*0,425*	0,252	0,152	0,151

Looking at the above table, Nettis best configuration is actually the best model trained on six months training set. For the other three datasets, Random Forest performed always in a better way with respect to the compared ELM-AD and Nettis settings.

In none of the configurations, ELM-AD model was able to perform better than the others. This underscores how although the model had more complex and technical features than Pump and Dump, these were ineffective compared to a less specific feature set used on a Random forest based model.

4.5.1. Results Ensemble Neural Network

The results obtained with Ensemble Neural Network system were not considered in the comparison with the others for the reasons explained in the previous section. As expected, the proposed binary classifier reached better results when compared to Nettis, Pump and Dump and ELM-AD systems. This was possible because of the inherent ease of the problem solved by binary classification, rather than classification on three classes performed by the other competitors.

	Nettis System	**Ensemble Neural Network**	
		All features	**Reduced features**
	1m, k=7, 6m	**One month**	
Precision	0,354	*0,384*	0,359
Recall	0,366	*0,504*	0,500
F1	0,305	*0,425*	0,414
	3m, k=3, 3m	**Three months**	
Precision	0,402	0,402	*0,403*
Recall	0,391	*0,504*	0,503
F1	0,349	*0,428*	0,427
	6m, k=5, 1m	**Six months**	
Precision	*0,460*	0,379	0,402
Recall	0,432	0,506	*0,511*
F1	0,397	0,428	*0,445*
	2y, k=5, 1m	**Two years**	
Precision	0,460	*0,532*	0,514
Recall	0,432	*0,539*	0,536
F1	0,397	*0,508*	0,506

Crypto	**Nettis System 1m, k=7, 6m**			**Ensemble Neural Network**					
				All features			**Reduced features**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
BTC	0,367	0,365	0,261	0,281	0,500	0,360	0,281	0,500	0,360
BTS	0,397	0,352	0,355	0,409	0,500	0,450	0,409	0,500	0,450
DASH	0,539	0,489	0,440	0,369	0,500	0,425	0,369	0,500	0,425
DGB	0,365	0,366	0,360	0,404	0,500	0,447	0,404	0,500	0,447
DOGE	0,391	0,394	0,314	0,348	0,500	0,411	0,348	0,500	0,411
ETH	0,396	0,387	0,328	0,644	0,570	0,483	0,225	0,500	0,310
IOC	0,396	0,384	0,366	0,299	0,500	0,374	0,299	0,500	0,374
LTC	0,483	0,361	0,349	0,354	0,500	0,415	0,354	0,500	0,415
MAID	0,243	0,252	0,221	0,378	0,500	0,430	0,378	0,500	0,430
MONA	0,332	0,352	0,331	0,252	0,500	0,335	0,252	0,500	0,335
NAV	0,479	0,476	0,477	0,363	0,500	0,421	0,363	0,500	0,421
SYS	0,272	0,330	0,178	0,484	0,500	0,492	0,484	0,500	0,492
VTC	0,056	0,333	0,096	0,477	0,500	0,488	0,477	0,500	0,488
XCP	0,287	0,383	0,227	0,406	0,500	0,448	0,406	0,500	0,448
XLM	0,278	0,303	0,214	0,352	0,500	0,413	0,352	0,500	0,413
XMR	0,337	0,300	0,284	0,374	0,500	0,428	0,374	0,500	0,428
XRP	0,395	0,397	0,388	0,331	0,500	0,398	0,331	0,500	0,398

Table 1: Comparison with Nettis best model trained with one month training set

Crypto	**Nettis System 3m, k=3, 3m**			**Ensemble Neural Network**					
				All features			**Reduced features**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
BTC	0,382	0,289	0,212	0,630	0,527	0,438	0,581	0,534	0,477
BTS	0,430	0,472	0,422	0,409	0,500	0,450	0,409	0,500	0,450
DASH	0,446	0,423	0,386	0,369	0,500	0,425	0,369	0,500	0,425
DGB	0,399	0,384	0,379	0,404	0,500	0,447	0,404	0,500	0,447
DOGE	0,500	0,485	0,465	0,348	0,500	0,411	0,348	0,500	0,411
ETH	0,450	0,362	0,273	0,275	0,500	0,355	0,275	0,500	0,355
IOC	0,400	0,325	0,265	0,299	0,500	0,374	0,299	0,500	0,374
LTC	0,459	0,465	0,461	0,354	0,500	0,415	0,354	0,500	0,415
MAID	0,285	0,309	0,202	0,378	0,500	0,430	0,378	0,500	0,430
MONA	0,345	0,360	0,340	0,585	0,533	0,447	0,644	0,517	0,379
NAV	0,415	0,423	0,416	0,363	0,500	0,421	0,363	0,500	0,421
SYS	0,342	0,444	0,264	0,484	0,500	0,492	0,484	0,500	0,492
VTC	0,350	0,359	0,313	0,477	0,500	0,488	0,477	0,500	0,488
XCP	0,338	0,312	0,297	0,406	0,500	0,448	0,406	0,500	0,448
XLM	0,434	0,430	0,431	0,352	0,500	0,413	0,352	0,500	0,413
XMR	0,430	0,394	0,394	0,374	0,500	0,428	0,374	0,500	0,428
XRP	0,423	0,409	0,408	0,331	0,500	0,398	0,331	0,500	0,398

Table 2: Comparison with Nettis best model trained with three months training set

Crypto	**Nettis System 6m, k=5, 1m**			**Ensemble Neural Network**					
				All features			**Reduced features**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
BTC	0,431	0,438	0,383	0,219	0,500	0,304	0,576	0,557	0,541
BTS	0,478	0,457	0,422	0,409	0,500	0,450	0,409	0,500	0,450
DASH	0,541	0,533	0,534	0,369	0,500	0,425	0,369	0,500	0,425
DGB	0,416	0,418	0,378	0,404	0,500	0,447	0,404	0,500	0,447
DOGE	0,525	0,523	0,507	0,348	0,500	0,411	0,348	0,500	0,411
ETH	0,469	0,462	0,445	0,275	0,500	0,355	0,275	0,500	0,355
IOC	0,442	0,375	0,336	0,299	0,500	0,374	0,299	0,500	0,374
LTC	0,478	0,345	0,290	0,354	0,500	0,415	0,354	0,500	0,415
MAID	0,385	0,376	0,354	0,378	0,500	0,430	0,378	0,500	0,430
MONA	0,400	0,421	0,395	0,609	0,597	0,585	0,629	0,627	0,626
NAV	0,499	0,468	0,459	0,363	0,500	0,421	0,363	0,500	0,421
SYS	0,560	0,478	0,374	0,484	0,500	0,492	0,484	0,500	0,492
VTC	0,319	0,365	0,254	0,477	0,500	0,488	0,477	0,500	0,488
XCP	0,544	0,368	0,319	0,406	0,500	0,448	0,406	0,500	0,448
XLM	0,410	0,391	0,387	0,352	0,500	0,413	0,352	0,500	0,413
XMR	0,474	0,478	0,475	0,374	0,500	0,428	0,374	0,500	0,428
XRP	0,451	0,446	0,441	0,331	0,500	0,398	0,331	0,500	0,398

Table 3: Comparison with Nettis best model trained with six months training set

Crypto	**Nettis System 2y k=5, 1m**			**Ensemble Neural Network**					
				All features			**Reduced features**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
BTC	0,431	0,438	0,383	0,559	0,534	0,497	0,583	0,554	0,529
BTS	0,478	0,457	0,422	0,490	0,493	0,486	0,503	0,502	0,493
DASH	0,541	0,533	0,534	0,565	0,577	0,565	0,543	0,550	0,543
DGB	0,416	0,418	0,378	0,905	0,507	0,461	0,628	0,510	0,473
DOGE	0,525	0,523	0,507	0,619	0,640	0,607	0,606	0,624	0,601
ETH	0,469	0,462	0,445	0,595	0,596	0,593	0,614	0,611	0,599
IOC	0,442	0,375	0,336	0,299	0,500	0,374	0,299	0,500	0,374
LTC	0,478	0,345	0,290	0,504	0,504	0,504	0,507	0,506	0,506
MAID	0,385	0,376	0,354	0,378	0,500	0,430	0,378	0,500	0,430
MONA	0,400	0,421	0,395	0,636	0,636	0,636	0,605	0,604	0,604
NAV	0,499	0,468	0,459	0,363	0,500	0,421	0,363	0,500	0,421
SYS	0,560	0,478	0,374	0,484	0,500	0,492	0,484	0,500	0,492
VTC	0,319	0,365	0,254	0,477	0,500	0,488	0,477	0,500	0,488
XCP	0,544	0,368	0,319	0,500	0,501	0,438	0,471	0,457	0,367
XLM	0,410	0,391	0,387	0,526	0,531	0,521	0,533	0,537	0,533
XMR	0,474	0,478	0,475	0,511	0,512	0,510	0,520	0,521	0,520
XRP	0,451	0,446	0,441	0,625	0,640	0,620	0,625	0,638	0,623

Table 4: Comparison with Nettis best model trained with two years training set

5. Conclusions and future works

In this work three different systems able to predict anomalies in the cryptocurrencies market were analyzed and compared to the Nettis LSTM Multi-target model. It was based on a cryptocurrencies clustering phase with k-medoid algorithm, clusters obtained were fed to three LSTM autoencoders able to reconstruct a specific type of row (stable, anomaly up, anomaly down). The one with the lowest MSE among the three identified the predicted class.
With the aim of having a comparative analysis between the various competitors, Nettis datasets based on different time intervals (1 month, 3 months, 6 months and 2 years) were employed to train the systems: Pump and Dump, ELM-AD and Ensemble Neural Network.
The Pump and Dump approach was focused on innovation related to the use of non-technical features rather than on a proposed model architecture, in fact, they used Random Forest and Logistic Regression models.
ELM-AD model used a feedforward neural network with a single hidden layer, without an iterative tuning during the training, Moore-Penrose generalized inverse was used to optimize weights.
Finally, Ensemble Neural Network was created stacking multiple sub-models. For each time dataset, five sub-models were trained for each crypto and then these were combined and used to construct the final ensemble model. This system was proposed just because its approach differed from the other two competitors, but it could not be included in the comparison analysis due to its binary classification nature.
Different configurations for each competitor were studied and compared with Nettis best models results. The best models among the different ones proposed by Nettis, were considered taking into account the models that obtained in his work greater improvement between LSTM with clustering and LSTM without clustering for each time interval dataset. The studies presented here highlighted how Nettis best model among the ones he studied, based on 6 months training for Multi-target LSTM and 1 month training for k-medoid clustering with k=5 got the best performances with respect to the other competitors. The Pump and Dump approach with Random Forest obtained the highest performances for the other three datasets, instead, ELM-AD results were quite far from the ones reached by Nettis and Pump and Dump competitors. This empathized how a simpler features set used on Random Forest model produced better results against the more complex and technical features chosen for ELM-AD.
For future works, in order to make comparable the performances obtained by Ensemble Neural Network, it could be interesting to modify the proposed stacked neural network so that multi-class classification could be applied.