Intelligent Price Alert System for Digital Assets - Cryptocurrencies

Sronglong Chhem
University of Derby
Derby
s.chhem1@unimail.derby.ac.uk

Ashiq Anjum
University of Derby
Derby
a.anjum@derby.ac.uk

Bilal Arshad
University of Derby
Derby
b.arshad@derby.ac.ukk

ABSTRACT

Cryptocurrency market is very volatile, trading prices for some tokens can experience a sudden spike up or downturn in a matter of minutes. As a result, traders are facing difficulty following with all the trading price movements unless they are monitoring them manually. Hence, we propose a real-time alert system for monitoring those trading prices, sending notifications to users if any target prices match or an anomaly occurs. We adopt a streaming platform as a backbone of our system. It can handle thousands of messages per second with low latency rate at an average of 19 seconds on our testing environment. Long-Short-Term-Memory (LSTM) model is used as an anomaly detector. We compare the impact of five different data normalisation approaches with LSTM model on Bitcoin price dataset. Result shows that decimal scaling produces only Mean Absolute Percentage Error (MAPE) of 8.4 per cent prediction error rate on daily price data, which is the best performance achieved compared to other observed methods. However, with one-minute price dataset, our model produces higher prediction error making it impractical to distinguish between normal and anomaly points of price movement.

KEYWORDS

Kafka, LSTM, Anomaly Detection, Real-time Monitoring System, Bitcoin

ACM Reference Format:
Sronglong Chhem, Ashiq Anjum, and Bilal Arshad. 2019. Intelligent Price Alert System for Digital Assets - Cryptocurrencies. In *IEEE/ACM 12th International Conference on Utility and Cloud Computing Companion (UCC '19 Companion), December 2–5, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3368235.3368874

1 INTRODUCTION

Unlike stock market trading, cryptocurrency trading platforms operate daily non-stop and combining with the volatility of the market itself, cryptocurrency traders and holders are facing the challenge of trying to keep track of all their asset values. For example, on Bitmax exchanges, there was an unexpected 80% drop of it token BTMX in just 31 minutes [1]. It would be impossible for them to monitor all the trading price actions manually and constantly. Therefore, a real-time alert system mechanism is required to help them manage risks. The system can send specific trading price alerts to users based on their preferences or automated alerts when current trading price meets the target threshold. To the best of our knowledge, only large exchanges such as Binance.com, Coinbase.com, and Upbit.com , equip their systems with an automatic mechanism. Moreover, only Coinbase provides an automatic alert to its users when Bitcoin and Ethereum price anomaly occurs within a twenty-four-hour window.

Although, most of the exchanges do not have such an automation system, they do provide an API and WebSocket endpoint for a third-party application to access their trading data. Therefore, an external system can utilise this data to build a real-time alert system that can independently operate from exchanges [27].

Such a system requires a framework that can handle a substantial amount of data in real-time and can seamlessly integrate that information into an existing system. Much like systems that incorporate data from High Performance Computing (HPC) and cloud platforms such as DIANA scheduler [3], Claren Web service [34], CMS workflow execution [15], Grid enabled analysis [33], Neuroimaging analysis [30] [31], Pervasive context provisioning [20], multi-objective optimization strategies [17] and Open-Flow [6].

Consequently, the purpose of this paper is to build a real-time cryptocurrency price alert platform from different exchanges. The hypothesis of this paper is how can machine learning be used to detect anomalies in cryptocurrencies trading prices?

To address the hypothesis, the following research questions need to be answered.

(1) How to build a real-time price alert streaming platform that can handle considerable amount of trading events, process them from various exchanges while querying users price alert targets to match with current prices and generate alerts with low latency?
(2) Does employing state-of-the-art deep learning LSTM model facilitate the automation process of detecting cryptocurrencies price movements without requiring users to set up specific price targets manually?
(3) Can the system handle large scale price comparison across multiple cryptocurrencies and their exchanges?

The research methodology employed to address the hypothesis and research questions is a combination of experimental and exploratory analysis [7]. Part of the problems will be analysed through literature review to find a suitable solution, and the other part will be analysed through experimentation.

UCC '19 Companion, December 2–5, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-7044-8/19/12…$15.00
https://doi.org/10.1145/3368235.3368874

2 LITERATURE REVIEW

2.1 Real-time Event Streaming Applications

Real-time event streaming mechanism could be implemented to help cryptocurrency traders managing risks in price volatility. Such a system has already been actively implemented in many existing domains where continuous adaptation of the application to the environmental conditions is critical such as fraud protection in financial services [21], logistics and asset management in manufacturing, telematics for automotive maintenance, and disease surveillance for public health [22], and agriculture sector [2].

Based on [8] Apache Kafka and RabbitMQ are the two popular open-source and commercial graded streaming systems (by Confluent Inc. and Pivotal) that have been widely integrated into enterprise companies. RabbitMQ outperforms Kafka in the most basic set up environment in terms of throughput. However, by increasing partition within the node, Kafka proves its scalability capability as it can significantly out perform RabbitMQ. As a result, the authors suggest the use case of Kafka for scalable ingestion system data-layer infrastructure, and stream processing due to its high throughput and scalability. Whereas request-response messaging, operational metrics tracking, and information-centric networking are better using Rabbit MQ.

Kafka has been adopted for data transmission between producer and consumer since it provides several out-of-the-box third-party integration support, including direct database connection to Cassandra and Hadoop, and streaming applications like Spark. D'Silve et al. [10] uses Kafka to handle message communication among IoT devices that need to send messages to each other over the cloud in real-time. In a study of license plate recognition, Kafka is responsible for receiving BlackBox videos, frame by frame, and sequentially deliver them to the node that performs recognition task [32]. A similar approach is implemented to support real-time anomaly detection in log data streams application [11].

At an enterprise level, each company has different reasons and purposes integrating Kafka into the production system. As an illustration, LinkedIn, the creator of Kafka itself, has been using it in their data centre to facilitate hundreds of gigabytes of log data and close to a billion messages per day generated by the front-end services [18].

2.2 Anomaly Detection

The anomaly detection in this paper refers to the monitoring of abnormal behaviour in the price movement of cryptocurrency trading price. This feature is crucial because it can inform the trader about price fluctuation in real-time so that they can make informed decisions of their trading strategies. One common approach for anomaly detection has been to build prediction models and use prediction errors to compute an anomaly score. A data point is flagged as an anomaly when the prediction error falls into a certain threshold [16].

The author of [13] uses Linear regression, Logistic regression, SVM, and Neural network to generate models for predicting Bitcoin price for one hour in advance. Several bitcoin network node features such as a number of transactions, new addresses, and the total number of bitcoin mined were used to build a supervised machine learning model. The authors expect that these features might affect its value exchange behaviour; additionally, they believe that the price movement is controlled by exchanges instead, as their model produces only 55% accuracy.

These regression algorithms normally treat each data sample individually, so they could not take advantage of the dependency information exhibited in time-series data. Consequently, LSTM can produce better prediction accuracy because of its ability to capture such dependency information.

Author of [4] employs LSTM to forecast the next day closing price of stock index. The authors introduce three levels of processes to produce the final prediction result. First, they use the wavelet transform to normalised input vector data before forwarding them to stacked autoencoder to extract the high-level features needed for LSTM predictor.

Phaladosailoed et al.'s [28] experiment proves that Deep Learning models such as LSTM and Gated Recurrent Unit (GRU) produce better prediction accuracy, then Theil-Sen Regression or Huber Regression. As GRU, an adjust version of LSTM, gives the best result of MSE at 0.0002 and R-Square (R2) at 99.2% based on a one-minute interval dataset of Bitcoin historical price data containing highly correlated features such as close, open, high, low, weight price, volume_BTC and volume_USD, and timestamp that can affect the prediction results.

The predicted values from the prediction algorithm are used to determine point anomaly. At Microsoft, the anomaly score is computed by configuring the LSTM model to produce output value range [0,1] to denote an anomaly point [29]. In Munir et al. [25] work, Mean Average Error (MAE) approach is used as an indicator to reduce an error between an actual and predicted value so that the LSTM network can learn to predict the normal behaviour of the time series. Finally, the predicted value is passed to the anomaly detector to measure the error between actual and predicted value by Euclidean distance formula (1):

$$error = \sqrt{(y_t - y'_t)^2} \tag{1}$$

where y_t, y'_t are actual and predicted value of time t respectively.

3 PROPOSED APPROACH

Time-series input data is non-linear and highly dynamic and needs to undergo a normalisation process because model performance is dependent on the consistency of data. The difference between the value scale of each input data increases the time for the model to train and could also affect its performance [9]. The normalised data is then used to detect anomalies.

The anomaly detection algorithm proposed in this paper contains two main steps. First, the prediction model is built to predict the future price of a specific digital asset by learning the regular pattern from its historical price data. Second, anomaly detection is performed by computing anomaly scores from the prediction errors.

3.1 Data Normalisation

Data normalisation is the primary data pre-processing technique for reducing the range of each time-series input value to the same range commonly between 0 to 1 or, -1 to 1. The effectiveness of time series forecasting is heavily dependent on normalisation technique

used on data before feeding them into the network for training and prediction.

There are many data normalisation techniques, but only five of them are used in our experiment including Min-max Normalisation, Decimal Scaling Normalisation, Z-Score Normalisation, Median Normalisation and Tanh Estimators [5].

In this paper, all the above methods are used to normalise input data to find the best normalisation method on cryptocurrency dataset that can produce better forecasting accuracy. The best one is then selected for normalising real-time cryptocurrency data.

3.2 Prediction Model and Anomaly Detection

We propose the use of multi-layered LSTM, otherwise known as Stacked LSTM, to build and train our cryptocurrency price prediction model because having more hidden layers gives the network more depth to divide the processing tasks. Each layer processes different tasks in hierarchical order before passing its output to the next layer until the output layer is reached [16]. It is an optimisation technique for a network to require fewer neurons and run faster [26].

As shown in figure 1, the network layer consists of an input layer, three LSTM hidden layers, a dense layer and a recurrent output layer. The number of hidden layer and number of unit in each hidden layer varies according to experiments that show better performance on the datasets. All layers are fully connected; therefore, dropout is used to avoid overfitting. Moreover, since this is a regression model, we use linear activation and MSE as the loss function.

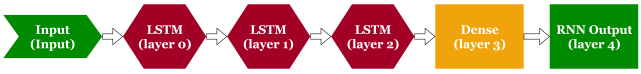

Figure 1: Proposed stacked LSTM architecture

In a real-time system, anomaly detection can happen only when the real input value becomes available. Hence, we propose a model to predict what would be the current price based on the historical price data and compare it with the actual current rate. We assume that by training model with historical price data, it can learn the normal patterns on each asset prices movement. Accordingly, when using it with new price data, it should produce higher error prediction rate on unfamiliar pattern regions.

To illustrate this, suppose that there is a collection of 31 timestamp data x_1,x_2,x_3,...,x_{30},x_{31},. However, the model accepts only 30 timestamp input vector x_1 to x_{30} and it will produce the y_{31}, which is the predicted value of x_{31}. The next input interval is x_2 to x_{31} to predict the y_{32}, respectively.

Each input vector x_n is a set of multiple individual asset trading indicators such as open, close, low, high price and trading volume. All of these raw feature values need to go through the normalisation process first to reduce their original value to the same range. It is important to note that other external factors such as political situation that may influence the price fluctuations are not considered in this research. We use all methods described in Data Normalisation section to normalise input data for comparison and select one of the suitable technique for the cryptocurrency price domain.

We propose the price input data to be a one-minute average computed from real-time trading data. Therefore, if there is something unusual, the system can send out an alert within a preferable timely manner instead of sending out multiple time alerts within a short range of time, for example, few second intervals. For the network to be able to learn to predict the normal behaviour of the time series more accurately, we use Absolute Error (MAE) (Equation (2)) to define the discrepancy between the actual value and the predicted value and try to reduce the error by modifying the LSTM training model.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|x_t - y_t| \tag{2}$$

Where n is the total number of the input sequence, x_t, y_t are the real value, and predicted value respectively.

Anomaly Score: The prediction model is trained only on data without any anomalies so that it learns the normal behavior of the time series. Then, the prediction errors produced by LSTM predictor are used as indicators of whether or not an alert should be sent out to a concerned party. Euclidean distance (Equation (1)) is used to determine the anomaly score. A high anomaly score indicates a significant anomaly at the given time t.

4 SYSTEM ARCHITECTURE.

We incorporate a real-time alert system concept to design a scalable cryptocurrency price alert system that will notify users through mobile push notification when one of these events occur:

(1) Pre-defined target prices match with current trading price.
(2) The price movement of target currencies matches the predefine anomaly threshold with the help of neural network algorithm.

Our real-time price stream system is built using Kafka and its Streams API since it is a distributed streaming platform capable of publishing and subscribing to a huge amount of record streams. Therefore, by following the design principle of Kafka [12], we integrate all necessary Kafka components into our proposed system as shown in figure 2.

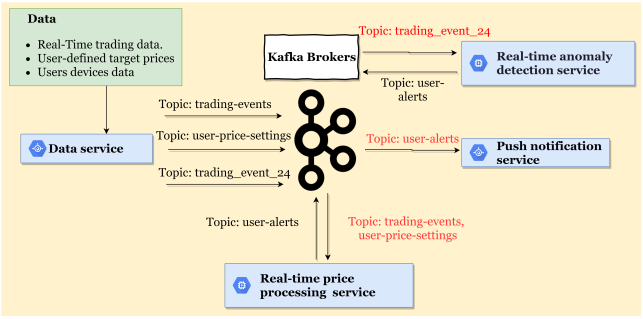

Figure 2: Real-time price alert streaming system in detail.

- **Data service:** Relevant data are extracted from different sources, including real-time trading data via exchanges, user-defined target prices, users device data via mobile application. WebSocket client is built to communicate with exchanges' API to get trading data and send them to Kafka brokers for price processing in real-time. Exchange name, asset name,

price and event time are forwarded to broker topic: trading_event. And price indicators such as open, close, low, high and volume of each digital asset are sent to topic trading_event_24.

- **Real-time price process service:** It processes incoming trading data by calculating an average of each asset price in the 10-second interval. Meanwhile, data from the userprice-settings topic are processed from a collection of key-value pair, where the key is exchange and asset name and value is the array price defined by users for the specific asset. They are then stored in Kafka Streams state-stores. Finally, the existing local state store is searched to get the list of push notification tokens of users who set target price similar (exact match or price difference less than 0.2 per cent) to the current average price of each asset by using the following formula:

$$\alpha = \frac{|price1 - price2|}{(price1 + price2)/2} * 100(where 0 \leq a \leq 0.2)$$

  - $price1$: *current price*
  - $price2$: *user set price*
  - α : *difference in percentage between the two*
- **Real-time price process service:** It consumes real-time price data, applies the normalisation method on it and forwards it to the LSTM model for prediction. The anomaly scores are calculated based on the prediction result to determine anomaly points in real-time.
- **Push notification service:** It is responsible for subscribing to user_alerts topic and process all incoming messages by sending FCM [24], a remote notification service provided by Google Firebase platform, to user devices.

5 EXPERIMENTAL ENVIRONMENT

5.1 Data Collection

In this proposed system, there are four primary types of data to be collected, namely: user device token, user price preferences data, real-time trading data, and historical trading data.

Device tokens are collected from mobile devices using an android app, while price preferences data is collected artificially by using the app to automatically select the trading assets and target prices instead of human interaction.

The real-time trading price information is collected from Binance exchange via WebSocket application to collect data and test our system. They are divided into real-time and one-second trading price data, which length are between 115 to 264 bytes, with detail statistics such as price, open, high, low and volume

Historical trading price data are needed as well for training machine learning algorithm, so daily and per-minute Bitcoin (BTC) price data from Gemini exchange are collected from an online resource named cryptodatadownload [1].

BTCUSDT daily dataset: There are seven features including date, symbol, open, close, low, high and the volume in each dataset. The BTC dataset consists of 1413 rows of recorded data daily from October 2015 to August 2019. The lowest and highest point during that period is suitable for selecting an appropriate data normalisation method. However, due to the inconsistency of volume feature in both dataset, it is excluded from our proposed input features. As a a result, only four features related to digital asset trading such low, open, high, close are normalised and fed to the network.

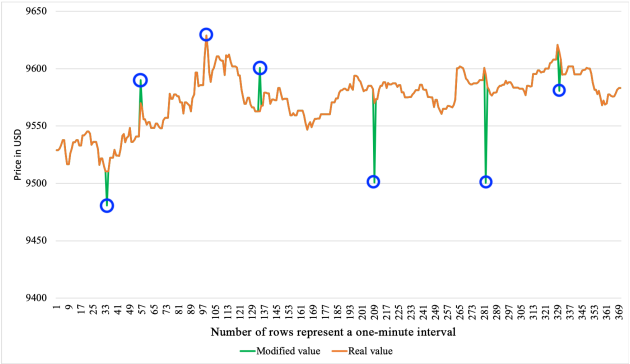

Figure 3: Anomaly point manually created to test the model

BTCUSDT minute dataset: BTCUSDT minute interval dataset contain 40325 rows and represents one minute of bitcoin price in August 2019. The price range during this month is minimum 9339.36 USD and a maximum of 12321.01, which is around 24.2% fluctuation. This data is split to 80% training and 20% testing dataset while the training data is then split another 20% from validation dataset.

Due to the lack of dataset containing labeled anomaly points for cryptocurrency, we decided to artificially create a few anomaly points out of the original dataset with the assumption that a few percentages of price movement make it look like an anomaly point. We select 367 minutes of bitcoin during August that already contain some few noticeable up and downtrend, and we artificially slightly increase or decrease those value, as shown in Figure 3. The blue dots indicate the artificially created anomaly points. The original dataset contain the minimum price of 9510.59 while 9480.59 in the artificial dataset. However, the maximum of 9629.07 USD remains unchanged.

5.2 Price Prediction Model

We used the LSTM model provided by Deeplearing4J [2] and adopt the implementation from Karim et al. [19]. The LSTM parameter setup is adapted from Liu et al. [23], which can be seen in table 1. Each algorithm was trained with 200 epoch and tested on a Bitcoin daily test dataset.

6 RESULTS AND ANALYSIS

6.1 Anomaly Detection

We want to choose a normalisation approach that responds well to our selected machine learning algorithm. Therefore, we make the comparison experiment on five types of data normalisation methods with our LSTM model, and the prediction accuracy is measured by Mean Average Error (MAE), MAPE, and Root Mean Square Error (RMSE).

[1] Available at: http://www.cryptodatadownload.com/data/northamerican/

[2] Available at: https://deeplearning4j.org/docs/latest/deeplearning4j-nn-recurrent

Name	Value
Learning rate iterations	A dam with initial of 0.001
Hidden layer 1 nodes (LSTM)	256
Hidden layer 2 nodes	256
Hidden layer 3 nodes	256
Dense layer nodes	32
Output layer node	1
Truncated BPTT length	30

Table 1: LSTM Parameters in this study

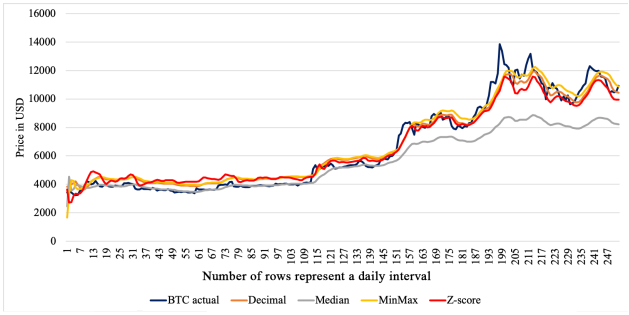

Figure 4: One-day-ahead prediction result of Bitcoin price four different data normalisation algorithm

The results can be seen in Figure 4. Among the five normalisation methods, Tanh Estimator produces the poorest result. According to table 2, while the other methods compete with each other, decimal scaling performs the best by producing the least error of percentage (around MAPE 8 per cent error.)

Consequently, will choose decimal scaling normaliser as a normalisation method for our anomaly detection. The Control chart in figure 5 shows the prediction accuracy of the algorithm on BTC daily data set where Upper Control Limit (ULC) value is 574.70 and Lower Control Limit (LCL) is -672.74.

The BTCUSDT minute-interval historical dataset contains a massive amount of data, therefore, requires a considerable amount of computing resource and time. Due to the constraint of space, we only train and test our model for the duration of 10-days from 20th to 30th of August. The reduced dataset contains 14405 records. Early stopping is used to find the best iteration for the dataset. As the nature of the dataset itself, the one minute ahead prediction result produces six per cent prediction error. This error gap is considered a significant error margin because the MAE is 698 USD difference. Therefore, we could not proceed to test the anomaly score because the average prediction already produced a large margin of error.

graphicx

6.2 Price Alert System Pipeline

Kafka is widely well-known for its ability to handle high volumes of data with high throughput. Therefore, the effect of message sizes on data transmission between broker and producers is tested on a total number of record and size of messages being able to send per second. As shown in figure 6, we send 500,000 messages to Kafka broker with different sizes (10,100,120, 300,1000 bytes). It

BTC	Time (m)	MAPE	MEA	RMSE
Decimal	27.67665	8.489823541	474.1409408	583.2750424
Median	28.18893333	11.24358042	1032.765968	1608.583322
Mimax	28.02226667	9.238080592	510.0762249	608.8254165
Z_score	28.56198333	9.183332373	526.0535892	667.8216022
Tanh	28.83513333	467.7654997	24006.9633	24272.68365

Table 2: Result of BTC Prediction Errors on the five approaches

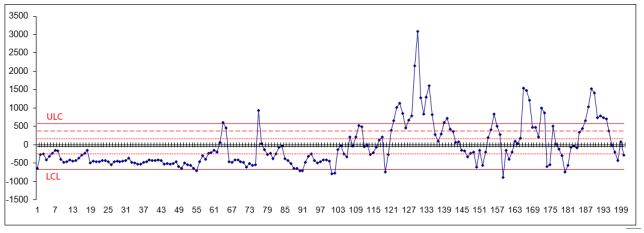

Figure 5: The control chart of prediction error in percentages calculating by percentage prediction error described in [14] using Decimal normalizer on BTC daily dataset.

is important to note that 120 and 300-bytes are the roundup size of the actual data, as described in section 5.3. They are real-time price data (average 115 bytes) and one-second trading information (average 264 bytes).

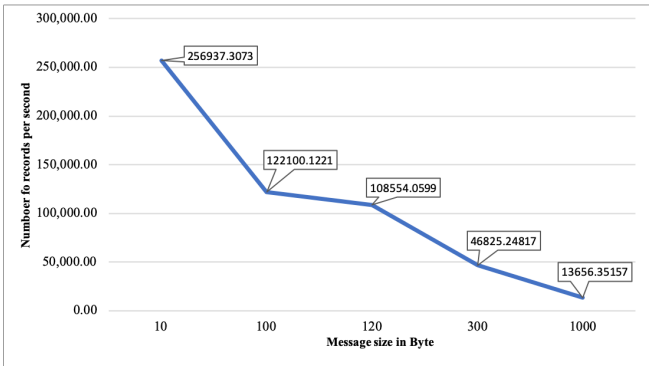

Figure 6: Size of message vs the number of records sent per second by the producer

Based on the results shown in Figure 6, we can see that the smaller the image size, the higher the number of the records that a producer can send to Kafka broker. In terms of latency, Figure 7 depicts that the bigger the size of the message, the longer it takes to complete the transaction. Therefore, with our actual data of 120 and 300-byte sizes, the test system can handle sending messages in total up to around 108,000 and 46,000 number of records respectively with a maximum throughput of around 13MB. Moreover, the maximum latency is only 2.3, and 7.2-seconds respectively, to send 500000 messages for both sizes.

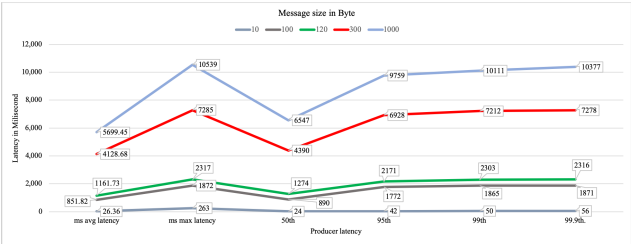

Figure 7: Latency of producer sending 500000 total to Kafka topic with different size

The scalability of Kafka is based on the data partition. The partition key is very important to achieve that. Kafka hashes the key in each message and guarantees that all the messages with the same key stay in the same partition. This feature ensures data consistency for multiple consumers in a group consuming messages. We send a total of 200,616 trading records one by one in real-time to a Kafka topic with eight partitions, and this process takes around 8.37 hours to complete. These records are equipped with key convention as follows: "key": exchange+ symbol. For exchange, key for Bitcoin symbol in USD value on Binance exchange is "key": "binanceBTCUSDT".

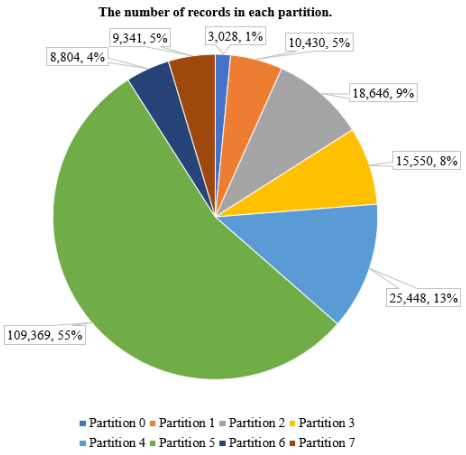

Figure 8: Messages are stored and spread across different partitions based on their key

As shown in Figure 8, by keying each record based on which exchange it is from and a trading symbol, Kafka can store them on different partitions according to those keys. We can see that a portion of the messages stored in each partition is different. This is because of the frequency of each trading symbol. As an illustration, partition five contains more than 50 per cent of the total number of records in a topic.

2000 price settings are preloaded into the system, and we measure the time it takes from a trading event message being sent by a producer to Kafka, then through stream processor for matching that preloaded price to Google Cloud Messaging service (push notification). The results of the first 1000 message are shown in

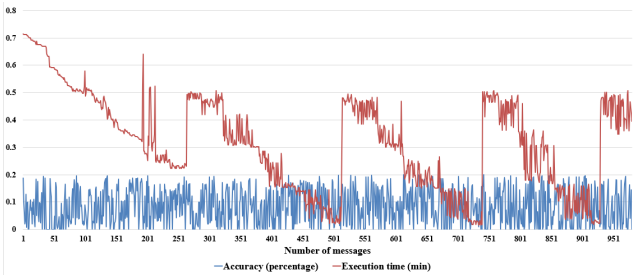

Figure 9: Response time of one producer sending 1000 message one by one.

Figure 9. The average execution time per message is around 0.31 minute (19 seconds), and the system can accurately match the price within a pre-determined threshold (0 or within 0.2 different).

7 RESEARCH LIMITATIONS AND FUTURE DIRECTIONS

The biggest challenge that we faced during the project is historical price data needed to train the LSTM model. The collection of public online datasets is used for this paper. We decided to build a model on four feature vectors only. They are low, high, close, open price, and we ignore the volumes information because of the inconsistentcy between one-day dataset and minute data.

There are several areas to be focused on future works. Firstly, adding the unsubscribe feature. This current system design only support adds operations. It means users can add as many prices alert target as they want, but they cannot unsubscribe their previous targets. Additionally, improving prediction error accuracy on a minute interval dataset. Our proposed LSTM model provides better accuracy on the daily dataset but produces higher prediction error on a minute dataset.

8 CONCLUSIONS

While the number of tradable digital assets increase, the traders and holders face the challenge of trying to keep track of all their assets price movement. It is impossible for them to monitor all the price actions manually. In order to address this issue we build an intelligent price alert system to meet their need of price tracking.

Our system can store user price alert target, and retrieve real-time trading prices of all the assets available from different exchanges. It then automatically sends alert to all the user's mobile devices whenever their price targets hits. We employ Kafka to build such a system and experimental results show that Kafka is an appropriate solution data pipeline system due to its scalability and high throughput. It ensures the trading data with the same key stays in the same partition to enable the system to scale in streaming application as well as alert processing application with ease. Test results show that it takes an average of 19 seconds for round trip for each price data; from sending it to Kafka system; getting it processed and sending an alert to a user's device.

REFERENCES

[1] 2019. Data Analysis of March 30 BTMX Price Volatility and Action Plan for Enhanced Risk Management. Retrieved September 12, 2019

from https://bitmaxhelp.zendesk.com/hc/en-us/articles/360020765654-Data-Analysis-of-March-30-BTMX-Price-Volatility-and-Action-Plan-for-Enhanced-Risk-Management

[2] Kayode E. Adetunji and Meera K. Joseph. 2018. Development of a Cloud-based Monitoring System using 4duino: Applications in Agriculture. *(2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD))*. IEEE, 4849–4854.

[3] Ashiq Anjum, Richard McClatchey, Arshad Ali, and Ian Willers. 2006. Bulk scheduling with the DIANA scheduler, Vol. 53. IEEE Transactions on Nuclear Science, 3818–3829.

[4] Jun Yue Bao, Wei and Yulei Rao. 2017. A deep learning framework for financial time series using stacked autoencoders and long-short term memory., Vol. 12. PloS one.

[5] Samit Bhanja and Abhishek Das. 2018. Impact of Data Normalization on Deep Neural Network for Time Series Forecasting. In *14th International Conference on Control and Automation (ICCA)*, Vol. 7. IEEE, 811–816. https://doi.org/arXivpreprintarXiv:1812.05519

[6] Baker Charlie, Ashiq Anjum, Richard Hill, Nik Bessis, and Saad Liaquat Kiani. 2012. Improving cloud datacentre scalability, agility and performance using OpenFlow, Vol. 122. IEEE Fourth International Conference on Intelligent Networking and Collaborative Systems, 20–27.

[7] Peter J. Denning. 1980. What is experimental computer science. *(Communications of the ACM 23)*. ACM. https://cs.uwaterloo.ca/~brecht/courses/854-Experimental-Performance-Evaluation-2011/readings/what-is-experimental-computer-science.pdf

[8] Philippe Dobbelaere and Kyumars Sheykh Esmaili. 2017. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper. *(11th ACM International Conference on Distributed and Event-based Systems)*. ACM, 227–238.

[9] K. A. J. Doherty, R. G. Adams, and Neil Davey. 2007. Unsupervised learning with normalised data and non-Euclidean norms, Vol. 7. Applied Soft Computing, 203–210.

[10] D'silva, Godson Michael, Azharuddin Khan, and Siddhesh Bari. 2017. Real-time processing of IoT events with historic data using Apache Kafka and Apache Spark with dashing framework *(International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT))*. IEEE, 1804–1809.

[11] Poojitha G. and Sowmyarani C N. 2018. Pipeline for Real-time Anomaly Detection in Log Data Streams using Apache Kafka and Apache Spark. In *International Journal of Computer Applications*, Vol. 182. 8–13. https://doi.org/10.5120/ijca2018917942

[12] Nishant Garg. 2013. Apache Kafka. Packt Publishing Ltd.

[13] Alex Greaves and Benjamin Au. 2015. Using the bitcoin transaction graph to predict the price of bitcoin.. In *No Data*.

[14] Massimo Baraldo Guang, Wu and Mario Furlanut. 1995. Calculating percentage prediction error: a user's note. *(Pharmacological research)*, Vol. 32.

[15] Khawar Hasham, Antonio Delgado Peris, Ashiq Anjum, Dave Evans, Stephen Gowdy, José M. Hernandez, and Eduardo Huedo et al. 2011. CMS workflow execution using intelligent job scheduling and data access strategies., Vol. 58. IEEE Transactions on Nuclear Science, 1221–1232.

[16] Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. Advances in neural information processing systems, 190–198.

[17] Habib Irfan, Ashiq Anjum, Richard Mcclatchey, and Omer Rana. 2013. Adapting scientific workflow structures using multi-objective optimization strategies, Vol. 8. ACM Transactions on Autonomous and Adaptive Systems (TAAS).

[18] Kreps Jay, Neha Narkhede, and Jun Rao. 2011. Kafka: A distributed messaging system for log processing.. In *NetDB*. 1–7.

[19] M.R Karim. 2018. Java Deep Learning Projects: Implement 10 real-world deep learning applications using Deeplearning4j and open source APIs. Packt Publishing Ltd.

[20] Saad Liaquat Kiani, Ashiq Anjum, Michael Knappmeyer, Nik Bessis, and Nikolaos Antonopoulos. 2013. Federated broker system for pervasive context provisioning., Vol. 86. Journal of Systems and Software, 1107–1123.

[21] Won-Joo Park Kee-Seong Cho Lee, Kyong-Ha and Won Ryu. 2014. RealCatch: A community-based real-time platform for financial fraud protection on smartphones. *(2014 International Conference on Information and Communication Technology Convergence (ICTC))*. IEEE, 362–366.

[22] Chung-Sheng Li. 2005. Real-time event driven architecture for activity monitoring and early warning. *(Emerging Information Technology 2005. (ICTC))*. 4–pp.

[23] Chunyu Liu and Tong Lai Yu. 2018. Open-Source Neural Network and Wavelet Transform Tools for Server Log Analysis. In *Proceedings of the 2018 International Conference on Data Science (ICDATA'18)*. CSREA Press, USA, 130–136. https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/ICD8077.pdf

[24] Laurence Moroney. 2017. Firebase Cloud Messaging. *(The Definitive Guide to Firebase)*. Apress, Berkeley, CA, 163–188.

[25] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. 2018. Deepant: A deep learning approach for unsupervised anomaly detection in time series. 1991-2005. In *IEEE Access*, Vol. 7.

[26] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. https://doi.org/arXivpreprintarXiv:1312.6026

[27] Pratik Sarkar Paul, Goutam and Sarbajit Mukherjee. 2014. Towards a more democratic mining in bitcoins. *(International Conference on Information Systems Security)*. Springer, Cham, 185–203.

[28] Thearasak Phaladisailoed and Thanisa Numnonda. 2018. Machine Learning Models Comparison for Bitcoin Price Prediction.. In *10th International Conference on Information Technology and Electrical Engineering (ICITEE)*. IEEE, 501–511.

[29] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft.. In *10th International Conference on Information Technology and Electrical Engineering (ICITEE)*. https://doi.org/1906.03821

[30] McClatchey Richard, Andrew Branson, Ashiq Anjum, Peter Bloodsworth, Irfan Habib, Kamran Munir, Jetendr Shamdasani, Kamran Soomro, and neuGRID Consortium. 2005. Providing traceability for neuroimaging analyses., Vol. 82. International journal of medical informatics, 882–894.

[31] McClatchey Richard, Irfan Habib, Ashiq Anjum, Kamran Munir, Andrew Branson, Peter Bloodsworth, Saad Liaquat Kiani, and neuGRID Consortium. 2013. Intelligent grid enabled services for neuroimaging analysis., Vol. 122. Neurocomputing, 88–99.

[32] Hong SeongHu, SangWon Jung, and ChangSung Jeong. 2018. Recognition method of license plate for black box video using Apache Kafka *(International Conference on Electronics, Information, and Communication (ICEIC))*. IEEE, 1–3.

[33] Frank Van Lingen, M. Thomas, Tahir Azim, I. Chitnis, Ashiq Anjum, D. Bourilkov, and M. Kulkarni et al. 2005. Grid enabled analysis: architecture, prototype and status.

[34] van Lingen Frank, Conrad Steenberg, Michael Thomas, Ashiq Anjum, Tahir Azim, Faisal Khan, Harvey Newman, Arshad Ali, Julian Bunn, and Iosif Legrand. 2005. The Clarens Web service framework for distributed scientific analysis in grid projects. IEEE International Conference on Parallel Processing Workshops (ICPPW'05), 45–52.