Detecting and Predicting Changes in Crypto Markets

Author Name
Affilliation
Affiliation
City and Country
email

***Abstract*—Cryptocurrencies are becoming increasingly more popular and relevant as a form of finance. Currently, there are a lot of trading bots that can help traders trade on the crypto exchanges. This paper presents a solution for predicting the market with data from influential market makers and tweets from known cryptocurrency investors. We use these data for setting take profit, stop losses, and opening and closing positions. The results, that we conducted during research will also be discussed in detail. We conducted tests to evaluate the solution's effectiveness and profitability. The tests proved, that the solution has promising results and can be profitable in both the short and long term. Promising results would also show if our model is reliable enough to be deployed on live data.**

***Index Terms*—Blockchain, Crypto-exchange, Bitcoin, Ethereum, crypto-currency, market, fungible-assets**

I. Introduction

Most of us are already familiar with Bitcoin [1]. It was proposed in 2009 by Satoshi Nakamoto and, in 2013, started to be traded on exchanges. Once Bitcoin became more popular, it started to attract investors due to price hikes. Even some of the now well-known cryptocurrencies derived from Bitcoin at the beginning. Many studies analyzed if there is the possibility of finding a correlation between Market prices, Bitcoin price, and sentiment (from social media for example). The study by Bouri et al. [2] conducted research on this topic and found that there is a slight correlation between the price movement of Bitcoin and some other related cryptocurrencies. This, however, is becoming less relevant as the market is now more spread and Bitcoin's margin is slowly evening out with other popular cryptocurrencies (eg. Ethereum [3]). As ElBahrawy et al. [4] confirm - Bitcoin has been steadily losing its advantage over other runner-up cryptocurrencies. According to [5], the vast majority of research is focused on Bitcoin, however, there are also studies that focus on a broader scale of the market [6]. This paper does not focus on specific cryptocurrencies in general. It proposes a universal solution to many cryptocurrency pairs. The paper also does not study cryptocurrency prices. It rather focuses on event changes and event predictions based on data received from important market makers and sentiment from X tweets. There are two different kinds of market makers:

- Liquidity provider - Used in classic fiat or crypto exchanges, holds a large number of shares. They can move with the market prices. Some Liquidity providers sell their data and it can be used for predicting future changes.
- Automated market maker (AMM) - As mentioned in [7], AMMs are an essential part of DeFi (Decentralized finance) and Dex (Decentralized Exchange) protocols/applications. They allow for exchange without any counterparty by creating liquidity pools made of lots of participants.

These operate on cryptocurrency exchanges from which there are also two different kinds:

- Centralized cryptocurrency exchanges (CEX) - Governed by a centralized entity. Famous for the incident with FTX [8]. Many people start to lose trust and interest in this kind of exchange.
- Decentralized cryptocurrency exchanges (DEX) - Fully decentralized exchanges that are not operated by a single entity and are not controllable beyond what the community wants to change. Much more reliable than centralized exchanges, but they can bring some other problems too. Similar to centralized exchanges, one has to trust a specific exchange and be sure that it is not a scam. These exchanges use many sophisticated models so that exchange can be controlled by a decentralized autonomous organization based e.g., on the already mentioned AMMs.

Every exchange uses so-called "Order books" that are lists of buy and sell commands. The study by Barbon et al. [9] compares the quality of CEX to DEX. They found out that Binance, which is a centralized exchange, scored the best among a trio of Kraken and Uniswap. An interesting point made by researchers is, that Uniswap could have had the best gas fees when it came to transaction cost testing. This, however, was only on the level of assumption because Ethereum awaited an upgrade to the proof of stake consensus mechanism at the time testing was performed. Because of that, real fee results were best for Binance.

II. Related Work

Cryptocurrency trading demands a considerable amount of time and attention. The market fluctuates constantly, with prices changing every minute. According to [10], the cryptocurrency market is one of the most volatile and fastest-growing markets. Traders need to react swiftly to every price movement to avoid financial losses and maximize their profits. However, for human traders, it can be challenging to keep up with every change in the market and evaluate them to make informed decisions.

To tackle this, traders rely on strategies to evaluate the changes in the market based on the data received from the market. Once they have analyzed the data, they must apply their evaluations to the market broker application. This process could take several seconds, if not minutes, for a human trader to complete. Consequently, traders would have to be online 24 hours a day to avoid missing out on any significant market events [11]. Thankfully, trading bots have emerged as an effective solution to evaluate market changes and upcoming events in a matter of milliseconds. These bots analyze the market data and send signals to the trader's account in the market broker, indicating when to open or close positions, set take profits, or stop losses. According to [12], there have been numerous neural networks developed to predict the prices of cryptocurrencies. Most of them focus on Bitcoin. The study [13] also focuses on predicting the regime changes of Bitcoin. They use the hidden Markov model for the classification of three different regimes. Stable, intermediate, and volatile. As [5] mentions, traders focus on predictability rather than prices going up or down. This is the major reason behind the development of prediction algorithms and prediction models. According to [14], the most widely used techniques for the prediction

of the market are artificial intelligence, support vector machines, and random forests.

Through the constant updates provided by a variety of other software tools (Telegram, RabbitMQ), traders can stay informed about the trading bot's activity, even when they are away from their computers. Overall, trading bots have significantly reduced the time and effort required to trade in the crypto exchange, enabling traders to maximize their profits while minimizing their risks. One of the interesting solutions mentioned in [15] uses stochastic neural networks to build models for predicting the prices of different cryptocurrencies. They claim that their solution, on average, brings 1.56% improvement on a dataset they selected. Another similar solution worth mentioning [16] uses machine learning techniques to predict the market. They tried three machine algorithms to predict changes in Bitcoin price movement. It was discovered that the ARIMAX algorithm brought the best result, while FBProp and XGBoost were far off compared to ARIMAX. They stated, that it could be improved even further with parameter hypertunning.

III. Design

The initial work and development of ideal strategies consisted of a good analysis of historical data from the market maker. For this case, we used a neural network LSTM [17] as the model architecture. The dataset used for training contains 434,237 rows and 19 columns. The data does not have null values but has NaN values between the columns (column price and smallestDelta applies only to event liquidity). This was a necessity to solve because of data transformation. After the model was trained, we received predictions that we could apply to fine-tune our strategies. The workflow of the initial analysis is explained in Figure 1.

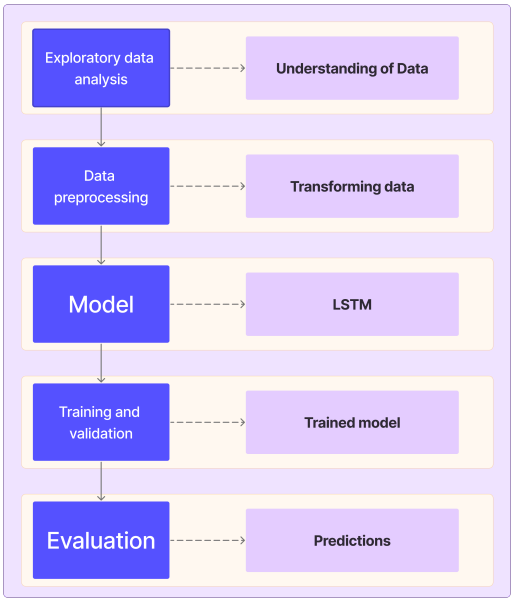

Fig. 1. Analysis workflow

The data we analyzed contain the following events:

- "liquidityLineCreated" signalizes that a zone with liquidity has been detected and this shows the price. Represents where liquidity should be left. The zones represent moments when market makers are about to sell some stock.
- "liquidityLineReached" signalizes that the price has reached an existing liquidity zone
- "zoneStart" signalizes that a zone preceding a price correction has been created
- "zoneExit" signalizes that a zone has been broken and price correction may happen. (This will be indicated when a price will start a movement.)
- "zoneExpansion" signalizes that the zone has turned into an oscillation range, where the price may swing around the fair price
- "zoneContraction" signalizes that zone being created (after the start and before exit) just got more narrow
- "zoneTentative" signalizes a high possibility a zone will be created soon
- "zoneRangeExit" signalizes that the price has left the zone and it is possible it will not come to return to the same position; the zoneRangeExit is when the price has moved significantly and the influence of the zone, on the price returning, will start diminishing
- "fairPriceCross" signalizes that the price has crossed the fair price
- "fairPrice" ("middle") represents the price around which we expect the price to oscillate until the market maker decides to make a price correction. There are a few scenarios depending on whether the price oscillates around them, or stays on one side.

After we understood all events we analyzed relationships between the selected pairs of attributes. We identified dependencies between pairs of attributes and dependencies between the predicted variable and other variables.

There were several positive correlations:

- low + high - correlation 1, which can be related to similar values, if high grows, so does low and below
- low + rangeLow - in this case we do not yet understand the rangeLow parameter as it relates to low
- low + rangeHigh - in this case we do not yet understand the rangeHigh parameter and how it is related to low
- low + range
- low + middle - correlation 1
- high + rangeHigh - lower correlation than with low + rangeHigh
- high + rangeLow - lower correlation than with low + rangeLow
- high + range - greater correlation than low + range
- high + middle - correlation 1
- rangeHigh + rangeLow - almost 1
- rangeHigh + range
- rangeHigh + middle
- rangeLow + range - lower than rangeHigh + range
- rangeLow + middle - identical to rangeHigh + middle
- price + smallestDelta - for event liquidityLineCreated

The Figure 2 of the heat map of correlations below explains this further. For each column of the table, we first get the event type. These then get grouped the values of individual events into a matrix according to the type of event. In the case of deep learning, it is necessary to first edit the data into such a form that it is possible to train a neural network. We first extracted the required information from the data in our experiments. Then we created a new dataset from the extracted pieces of information. Replacing the missing values is a crucial step. The deep learning model can not have missing pieces of information. Boolean values in the modified dataset are transformed into integer values. Next, we selected nine interesting columns with which we continued working with. We also

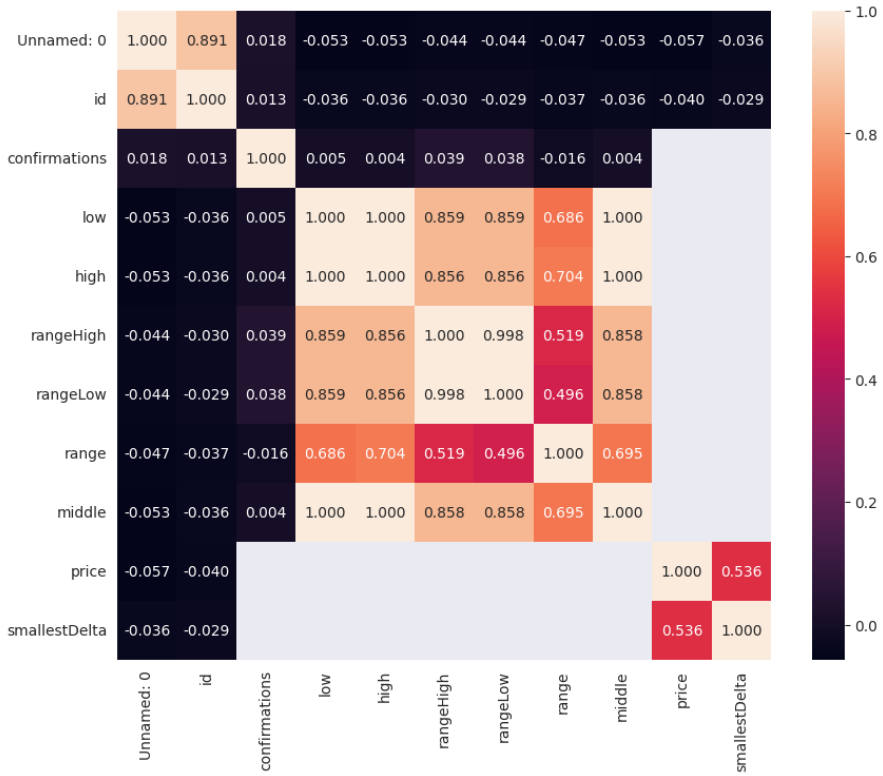

Fig. 2. Heatmap of correlations

transformed eight different events into a numerical form. Rescaling these numerical values is a good practice for training neural networks. Algorithm 1 shows the process of transforming the dataset.

Algorithm 1 Transform data

Require: extracted data from market makers' dataflow D
Ensure: transformed data Y

```
Fill missing values in D.
for each column y ∈ 𝒟 do
    if y is Boolean then
        transform y to Integer
    end if
    if y has low as value then
        y ← 0
    end if
    if y has high as value then
        y ← 1
    end if
    if y is event then
        code y values as Integer
    end if
end for
```

For our solution, we used the type of recurrent neural network LSTM. According to [18] - "LSTMs are predominantly used for learning, processing, and classification of sequential data because these networks can learn long-term dependencies between time steps of the data." The final architecture for our solution consists of 4 layers of 32 neurons, and as an activation function, we use the ReLU function after each layer. To avoid overfitting, we use a dropout level of 0.2, which means we turn off 20% of the neurons in each layer.

We have eight input features during training, while our model generates event predictions as an output. During the experiments, we tried different numbers of epochs during training. For the final experiment, we chose 50 epochs, while the learning rate, in this case, was 0.001. The loss function is Mean Square Error, which is suitable for predictions, and we used Adam as the optimizer. If the value of Loss decreases, the model is learning.

The data must be further divided into training, validation and testing sets. In the case of the validation and test set, we combined the data in different ways to avoid over-learning. The test set is the data the model has not seen, thus serving as the final prediction.

Algorithm 2 shows the training of our deep learning model. Validation is done the same, but updating parameters and gradients is skipped.

Algorithm 2 Training model

Require: number of epochs E, input features X, target Y
Ensure: trained model

```
for each epoch ∈ ℰ do
    for each x, y ∈ X, Y in random order do
        calculate the gradient
        train LSTM model
        calculate ŷ
        calculate the loss function
        perform an optimization step
    end for
end for
```

After initial analysis with a neural network, we gained enough promising results to start work on the solution itself. The solution works with data from influential market makers and Twitter. Market

maker data is defined by big organizations that have enough capital to move with the market and it consists of events, that signal their next actions. Our solution uses this data for setting take profit, stop losses, opening and closing positions. In some cases, we also rely on the current price of the cryptocurrency to make successful next steps. Our program consists of two separate processes.

- First process contains the main strategy and fetches data from the market maker data stream. It also evaluates fetched data to create an optimal strategy.
- Second process continuously fetches the current price from Binance API [19] and evaluates it with specific passed events.

Both of these processes interact with a trader who receives orders from our solution right after event evaluation. It then performs operations with the market broker (for example Binance). Our solution can work with multiple cryptocurrency pairs at the same time. It can also have multiple trading accounts attached. There is a limitation to this currently. Every connected account can only listen to the same orders. It is planned to be resolved in future work.

We designed our solution in Python programming language which is widely known for its ease of use and broad scale of use cases. The solution consists of two processes that have their own memory and processing power allocated separately. The first process listens on the RabbitMQ consumer channel (queue) for upcoming events from the market maker data stream. RabbitMQ [20] is an open-source message broker that is used by many enterprises. It supports multiple messaging protocols and can be deployed in distributed and federated configurations to meet high-scale and high availability requirements. Handling upcoming events consists of starting strategy with data received from the event and strategy runs several checks where attributes from events are compared with already opened positions and in case, the conditions are met, we open a new position. The last step if a new position is opened is sending the order to the trader. These orders are also saved in a database which consists of a few different tables.

- Table for opened positions
- Table for closed positions
- Table for "zoneStart" events
- Table for "zoneExit" events

The first process also handles these events for multiple cryptocurrency pairs. The second process handles the current price of cryptocurrency pairs from Binance API. Its diagram can be found in the Figure 5. We fetch a new candle for a specific cryptocurrency pair and compare it to an associated opened positions in the database. This process also compares current price candles with some specific saved zone events in the database and in case of meeting conditions necessary to open a new position is opened. This is done by sending a signal to the trader through the RabbitMQ channel. Because we need to deal with multiple cryptocurrency pairs at the same time this process has to have multiple threads.

As already mentioned solution uses RabbitMQ to fetch data from the market maker stream, which sends new upcoming events to our RabbitMQ receiver server. Its architecture is explained in Figure 3. Our solution then sends order signals to another remote RabbitMQ server responsible for making trades.

The solution has two types of evaluating strategies.

- First strategy is based on the evaluation of events called "zoneStart" and "zoneExit".
- Second strategy is based on events called "liquidityLineCreated" and "LiquidityLineReached". It is also known as the "scalping" strategy.

In the first strategy, we await an event that is called "zoneStart". Once we receive such an event, the solution saves the event to the database and also looks for every opened position in it. If it is found, the solution moves it from the opened positions table to the table for closed positions. It then sends a close signal for every position through the RabbitMQ producer. This is done because we expect, that cryptocurrency will be moving horizontally until the "zoneExit" event. When the zone ends, the horizontal way will switch to the vertical way and that could mean potential risk for open positions as we do not know where will zone move. Positions are differentiated by the so-called "position-id".

If we receive an event that is called "zoneExit" our solution just saves the event to the database for further use.

The second strategy awaits "liquidityLineReached" events. Once received, the solution opens a new position and specifies it as opened from "liquidityLineReached" for further use. The position is saved to the database and a signal containing the position id is sent to the trader. Stop-loss of this position is calculated for two scenarios in the following ways:

- Long position $\rightarrow liquidityLineReachedPrice \cdot 0.90$
- Short position $\rightarrow liquidityLineReachedPrice \cdot 1.10$

The type of position depends on the side of the "liquidityLineReached" event. If the liquidity line that is reached is side "High", we open a short position. On the other hand, if the event is side "Low", we open a long position.

The second type of event we look for in the second strategy is "liquidityLineCreated". Once we receive that event we check every opened position from the event "liquidityLineReached". The "liquidityLineCreated" event is for setting take profits. It updates every position opened from "liquidityLineReached" and sets its take profits to the price we received in "liquidityLineCreated". LiquidityLines can be on both sides. If the side is "Low" then we set take profits to short positions and if the side is "High" then take profits is set to long positions.

In the Figure 4, we can observe process 1 decision-making. As we can see, we are constantly fetching price candles from Binance API. We then compare every price candle with every opened position in the database. If the price candle matches the value in the column take profit or stop the loss of queried opened positions, if conditions are met we close the position by moving the selected position from open positions into closed positions. There is no signal to the trader in this case.

In the second case, we check the database table "passed-zone-events", if there is an event "zoneExit" and there is no "zoneStart" afterward. If this is the case, it means, that the new zone has not yet started and we check for certain conditions:

- We open long position if price is equal to $highSideOfZoneExit + (highSideOfZoneExit - lowSideOfZoneExit)$ and "exitSide" is equal to "High"
- We open short position if price is equal to $lowSideOfZoneExit - (highSideOfZoneExit - lowSideOfZoneExit)$ and "exitSide" is equal to "Low"

These calculations are done to ensure that the price has a trend to go in a vertical way which we need. The "zoneStart" event opens the zone and the zone represents a horizontal way on the graph which by itself is not enough for good trading decisions. "zoneExit" represents the end of currently last zone and we can predict, that price will most likely oscillate into the vertical way.

During our research, we discovered, that it is very efficient to be able to change strategy in real time if necessary. Changing

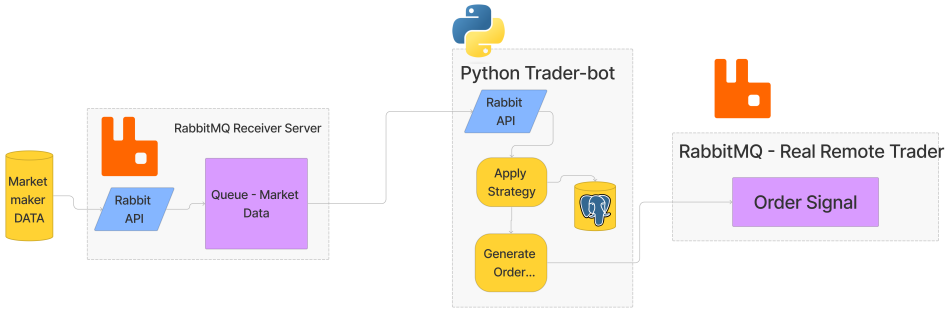

Fig. 3. RabbitMQ Architecture

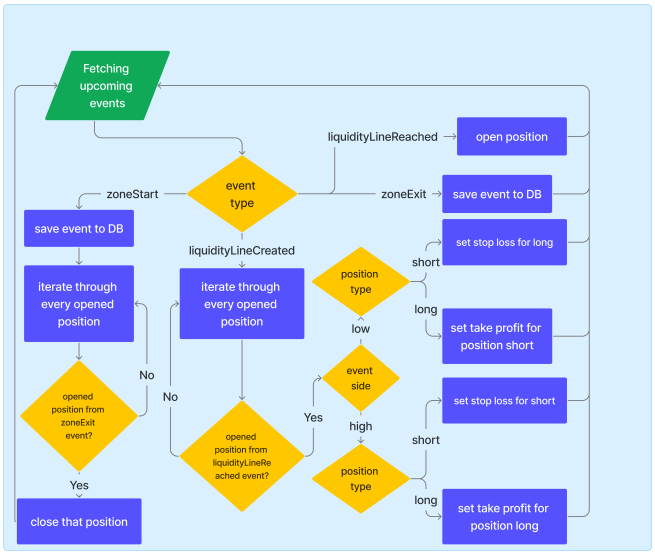

Fig. 4. Process 1, decision diagram

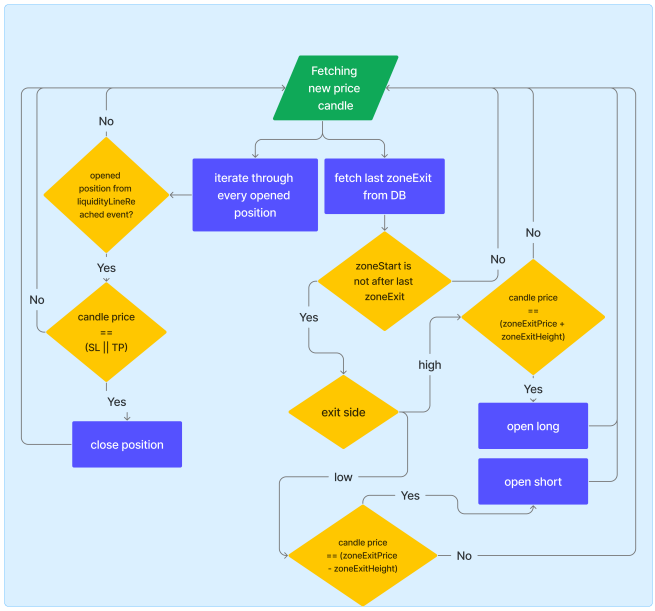

Fig. 5. Process 2, decision diagram

certain calculation values, decisions in strategy or adjusting different conditions for opening and closing turned out to bring more profitable outcomes. These strategies are tested in a special variation of our solution which does not operate with real money. To get broader results we also tried testing on historical data. Once the strategy proves to be profitable we can deploy it to our solution which runs live. This is done with a specific GitHub pipeline that updates the source code on the server, then the script sends a block signal to the trader so no trades are being made. Once everything is stopped our solution applies an update automatically and restarts itself.

The second important part of our solution is Twitter sentiment analysis. Before we can start pulling tweets we need to know which tweets from which accounts are relevant. The first important step was an analysis of accounts, that have influential posts regarding Blockchain as well as relevant hashtags. In hashtags we looked at popular Blockchain terms, checking if they are used on Twitter and if they are used with information regarding cryptocurrencies and the cryptocurrency market. From the comprehensive list we made, @elonmusk, @krakenfx and @Polkadot are a few, that we decided to mention. From hashtags, we decided to mention #Binance, #cryptoAlerts and #DEFI. They proved to be among the most influential.

We have created two different Python scripts to pull tweets and each has its purpose:

- The first script is used to fetch historical data
- The second script is used to pull tweets from selected accounts and hashtags daily.

Pulling tweets happens in three cycles. The first cycle consists of pulling tweets from accounts. This cycle uses API call `.user_timeline` which allows for pulling the last 3200 tweets from selected accounts. The second cycle goes through all selected hashtags. This cycle uses API call `search_tweets`, which typically allows for pulling tweets that are up to one week old. The third cycle uses the same API call as the second one, but this one pulls combinations of hashtags. Pairing multiple hashtags together could filter out irrelevant data.

Tweets are stored in files with a ”.csv” extension due to the limitations of our current system. In future work, we want to upgrade that storing mechanism to store data in the database. This database will consist of 4 columns that will contain the following:

- Account name

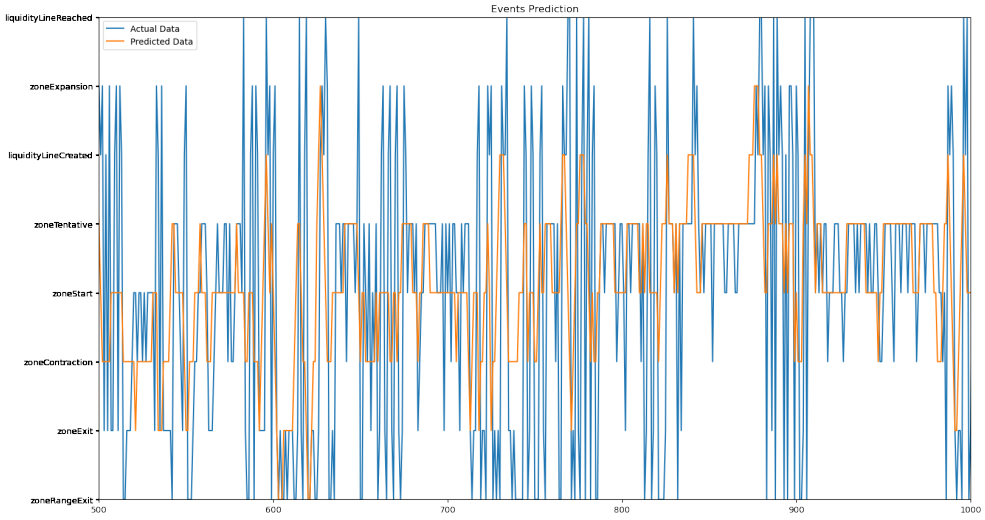

Fig. 6. Predicted data

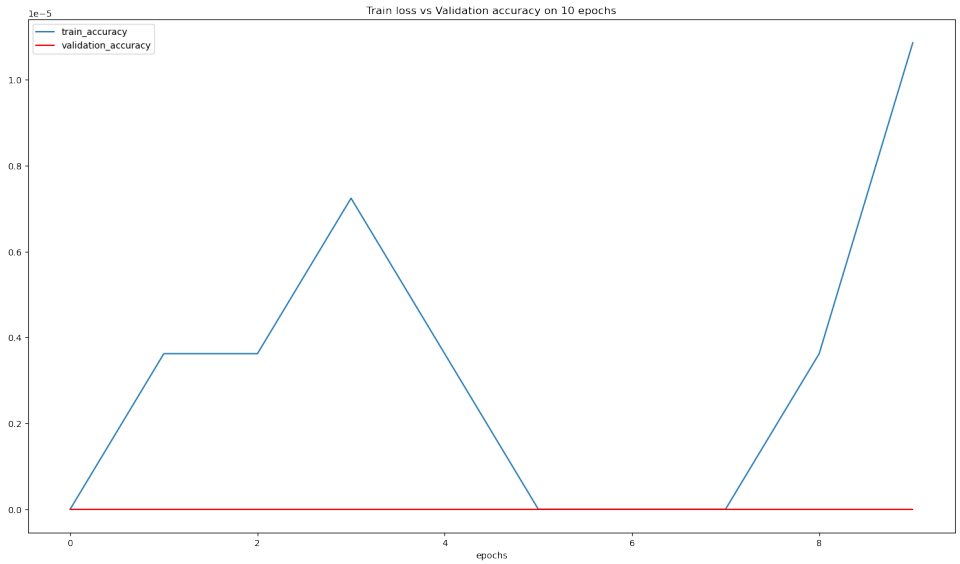

Fig. 7. Accuracy of model

- Tweet
- Timestamp
- Hashtag used for search (Empty if the search is performed by account name instead of the hashtag)

IV. Test scenarios

To fully test our solution before it is used with real finances we designed two tests and a few simulations. These simulations are tested on historical data from the market maker for the year 2021. Another essential part is prices corresponding to specific times. These are pulled from Binance which provides API for historical data. Simulations consist of:

- Process 1 consists of a RabbitMQ listener, that listens for upcoming events.
- Process 2 fetches prices from the Binance API in an infinity loop. After the price is fetched, our program sleeps for 0.5 seconds and the next timestamp is moved 5 minutes ahead and this loop is repeated with a new timestamp. Sleep delay is necessary to avoid Binance API parser timeout.
- Process 3 is responsible for sending new events to the RabbitMQ listener running in Process 1. The process consists of a loop that parses the next event in the .csv test file. It also checks the time stamp against the last saved price in the database. If they are approximately matching, we send an event to the RabbitMQ listener and continue to the next event.

We designed the following two tests:

- Test 1 - Running solution for 19 days on historical data with fetching candle prices every 5 minutes
- Test 2 - Running solution for 53 days on historical data with fetching candle prices every 5 minutes

Tests were designed to compare the short-term to long-term efficiency of our solution.

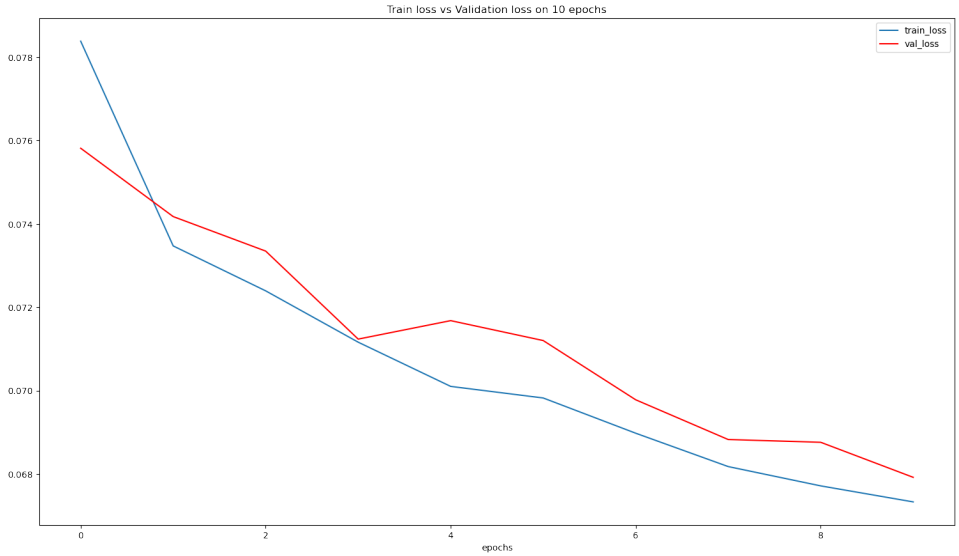

Fig. 8. Loss of model

V. Results

Data that are also worth mentioning are predicted data, the accuracy of the model, and the loss of the model. These are from initial analysis and they can be seen in the Figures 6, 7, and 8.

We ran two simulation tests to evaluate our solution and its profitability. As already mentioned first test was designed for the 19-day interval from 05.07.2023 to 23.07.2023. We noticed positive results in this time frame. The solution was able to trade up to 14% in profits. In the second test, we let the simulation run in the time frame from 01.08.2023 to 19.10.2023. It successfully opened and closed 209 positions. The profitability of this test was higher. The solution was able to trade up to 39% in profits. This diametrical difference shows us, that our solution profitability also depends on the time frame. In conclusion, our solution seems to be profitable in both tests and time frames. The strategies could always be tuned to perform better however we found this to be the most optimal setup for now. The Figures 9, 10 show the results of both tests.

VI. Conclusion and future work

Our tests proved, that our model has promising results and can be profitable in the both short and long term. This is due to research that was done to fine-tune the strategies of our solution. These can always be improved even more. As for future work, we wish to improve in many aspects. In the case of evaluation, a metric for accuracy, we compare rescaled normalized actual event values and predicted values this could be researched more. Upon an LSTM deep learning model for predicting cryptocurrency events, we also started building an LSTM classifier. Further experiments for searching the best depth and width and also hyper-parameters tuning will be done in the future. Promising results would also show if our model is reliable enough to be deployed on live data. From other limitations, our solution has, we wish to resolve the receiving of different orders for different accounts as currently, all accounts receive the same orders. Another future work would be creating a database for tweet pulling. This database would consist of four main columns as mentioned above in the design section. It would save space and make working with tweets a lot easier.

References

[1] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System”. en. In: ().

[2] Elie Bouri, Syed Jawad Hussain Shahzad, and David Roubaud. “Co-explosivity in the cryptocurrency market”. en. In: *Finance Research Letters* 29 (June 2019), pp. 178–183. ISSN: 1544-6123. DOI: 10.1016/j.frl.2018.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1544612318302976 (visited on 03/12/2023).

[3] *Ethereum official website*. URL: https://ethereum.org/en/.

[4] Abeer ElBahrawy et al. “Evolutionary dynamics of the cryptocurrency market”. In: *Royal Society Open Science* 4.11 (Nov. 2017). Publisher: Royal Society, p. 170623. DOI: 10.1098/rsos.170623. URL: https://royalsocietypublishing.org/doi/full/10.1098/rsos.170623 (visited on 03/12/2023).

[5] Erdinc Akyildirim, Ahmet Goncu, and Ahmet Sensoy. “Prediction of cryptocurrency returns using machine learning”. en. In: *Annals of Operations Research* 297.1 (Feb. 2021), pp. 3–36. ISSN: 1572-9338. DOI: 10.1007/s10479-020-03575-y. URL: https://doi.org/10.1007/s10479-020-03575-y (visited on 03/12/2023).

[6] Guglielmo Maria Caporale, Luis Gil-Alana, and Alex Plastun. “Persistence in the cryptocurrency market”. en. In: *Research in International Business and Finance* 46 (Dec. 2018), pp. 141–148. ISSN: 0275-5319. DOI: 10.1016/j.ribaf.2018.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0275531917309200 (visited on 03/12/2023).

[7] Vijay Mohan. “Automated market makers and decentralized exchanges: a DeFi primer”. en. In: *Financial Innovation* 8.1 (Feb. 2022), p. 20. ISSN: 2199-4730. DOI: 10.1186/s40854-021-00314-5. URL: https://doi.org/10.1186/s40854-021-00314-5 (visited on 03/25/2023).

[8] *FTX Recent bankruptcy report*. URL: https://www.investopedia.com/what-went-wrong-with-ftx-6828447.

[9] Andrea Barbon and Angelo Ranaldo. *On The Quality Of Cryptocurrency Markets: Centralized Versus Decentralized Exchanges*. arXiv:2112.07386 [q-fin]. Oct. 2022. DOI: 10.48550/

Fig. 9. Simulation test number one

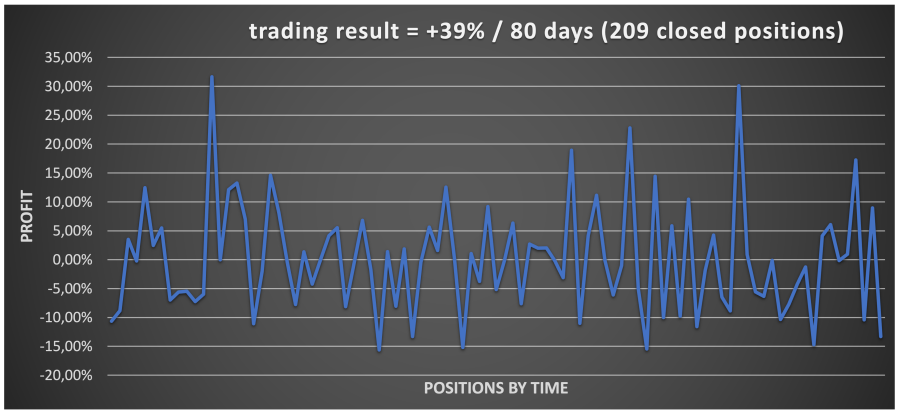

Fig. 10. Simulation test number two

arXiv.2112.07386. URL: http://arxiv.org/abs/2112.07386 (visited on 03/25/2023).

[10] Fan Fang et al. "Ascertaining price formation in cryptocurrency markets with machine learning". In: *The European Journal of Finance* 0.0 (Apr. 2021). Publisher: Routledge _eprint: https://doi.org/10.1080/1351847X.2021.1908390, pp. 1–23. ISSN: 1351-847X. DOI: 10.1080/1351847X.2021.1908390. URL: https://doi.org/10.1080/1351847X.2021.1908390 (visited on 03/12/2023).

[11] Hamid Jazayeriy and Mohammad Daryani. "SPA Bot: Smart Price-Action Trading Bot for Cryptocurrency Market". In: *2021 12th International Conference on Information and Knowledge Technology (IKT)*. 2021, pp. 178–182. DOI: 10.1109/IKT54664.2021.9685662.

[12] Sina E. Charandabi and Kamyar Kamyar. "Prediction of Cryptocurrency Price Index Using Artificial Neural Networks: A Survey of the Literature". en. In: *European Journal of Business and Management Research* 6.6 (Nov. 2021). Number: 6, pp. 17–20. ISSN: 2507-1076. DOI: 10.24018/ejbmr.2021.6.6.1138. URL: https://ejbmr.org/index.php/ejbmr/article/view/1138 (visited on 03/12/2023).

[13] Osamu Kodama, Lukáš Pichl, and Taisei Kaizoji. "REGIME CHANGE AND TREND PREDICTION FOR BITCOIN TIME SERIES DATA". In: *CBU International Conference Proceedings* 5 (Sept. 2017), p. 384. DOI: 10.12955/cbup.v5.954.

[14] Franco Valencia, Alfonso Gómez-Espinosa, and Benjamín Valdés-Aguirre. "Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning". en. In: *Entropy* 21.6 (June 2019). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, p. 589. ISSN: 1099-4300. DOI: 10.3390/e21060589. URL: https://www.mdpi.com/1099-4300/21/6/589 (visited on 03/12/2023).

[15] Patel Jay et al. "Stochastic Neural Networks for Cryptocurrency Price Prediction". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 82804–82818. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990659.

[16] Mahir Iqbal et al. "Time-Series Prediction of Cryptocurrency Market using Machine Learning Techniques". In: *EAI Endorsed Transactions on Creative Technologies* "8".28 (July 2021). ISSN: 2409-9708. URL: https://eudl.eu/doi/10.4108/eai.7-7-2021.170286 (visited on 03/12/2023).

[17] *LSTM Neural network explanation*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[18] *LSTM explanation*. URL: https://www.mathworks.com/discovery/lstm.html.

[19] *Binance API documentation*. URL: https://binance-docs.github.io/apidocs/spot/en/#introduction.

[20] *RabbitMQ Message broker*. URL: https://www.rabbitmq.com/.