Big Data Science
CSCI-GA.3033-001
Spring 2020
Prof. Anasse Bari
New York University
Final Project
Kevin Choi (kc2296)

Predicting Ethereum's Price
Based On Its Blockchain Activities and Sentiment Towards Cryptocurrency

Abstract

While the market of cryptocurrency has been an unprecedentedly volatile one in the past decade, we explore the case with Ethereum since 2019 such that we gather, preprocess, and process data from three main sources in an attempt to model its price: price data itself, blockchain data, and data from Twitter. Here, blockchain data refer to on-chain data such as average gas price, total hash rate in the blockchain network, number of existing addresses, etc. all of which signify the health of the network as a global data structure. As for Twitter, we note the advent of so-called Crypto Twitter, a Twitter clique of investors, traders, engineers, celebrities, and speculators, and thus utilize the tweets by top Crypto Twitter influencers in our prediction model. Incorporating these data in addition to Ethereum's price data, we observe that our model is able to generate predictions with smaller error when compared to traditional time series models.

Table of Contents

1. Introduction

Launched in 2015, Ethereum is currently the second most popular cryptocurrency in the market (by market cap). As it is often compared with its predecessor Bitcoin, one of the key differences is that Ethereum was the first blockchain to enable a wide usage of smart contracts (a concept that dates back to 1994 by Nick Szabo), which have been increasingly gaining popularity in various forms since Ethereum's inception. For instance, some of these smart contracts have taken the form of an auctioning game (e.g. CryptoKitties) while others have demonstrated various use cases such as certificate authentication, copyright protection, insurance, supply chain management, tunable peer-to-peer transactions, decentralized governance, and voting.

As this has been the case, it is possible for Ethereum to publicly generate a lot of auxiliary data via its network as network participants not only generate transactions among themselves, but also interact with numerous smart contracts that have been deployed in a continual manner. We outline Ethereum's network data in the following.

1. Supply: As an Ethereum block becomes mined by a miner (who by definition acts as a verifier of Ethereum transactions), new Ethereums are generated to compensate those miners. (In addition, there exist compensations related to mining slightly tangential uncle blocks as well.) As a result, the supply of Ethereum can be seen as a function that is always increasing over time.

2. TxGrowth: Transaction growth denotes the number of transactions per time frame.

3. AddressCount: Whenever a network participant generates a new public address, such event makes a contribution of 1 to Ethereum's address count.

4. BlockSize: That an Ethereum block is big means that it is packed with transactions. Bigger block size means either the network is full of transactions, miners are letting each block become big, or both.

5. BlockTime: Historically, the block time of Ethereum has hovered around 10-19 seconds. This is the time interval between when transactions get published and verified.

6. AvgGasPrice: Gas represents Ethereum's network fee. High gas price means the transaction fee is high at the moment due to (perhaps) competition of transactions when there are too many.

7. GasUsed: This denotes the amount of gas that has been used per time frame.

8. NetworkHash: The network hash rate refers to the amount of computing power miners are dedicating to the Ethereum network at the moment in an attempt to mine Ethereum blocks. Higher rate means there exists a higher competition among miners.

9. TransactionFee: This denotes the total amount of network fee in Ethereum (not in gas) per time frame.

10. NetworkUtilization: This denotes the average gas used over the gas limit in percentage. Each (smart contract) transaction can specify the maximum amount of gas that can be used for that transaction in the form of a gas limit.

11. BlockReward: Higher block reward means that miners are compensated more per block mined such that it becomes more incentivizing for miners to participate in the Ethereum network by mining.

As such, it is interesting to see that the above parameters would tend to reflect the complex game-theoretic dynamics within the Ethereum network in theory, although things have not necessarily been always predictable (in a theoretical way) in practice. Then the idea is that such dynamics would in theory either reinforce or be reinforced by Ethereum's price such that exploring their relationship is of interest.

Meanwhile, we note the advent of so-called Crypto Twitter, which has sprouted out as a substantial community on Twitter as a clique since the cryptocurrency boom in late-2017. In it, the observation is that thousands of crypto investors, traders, engineers, celebrities, and speculators consume news and sentiment-based opinions on a daily basis such that we deemed it necessary to include such data from Twitter to our exploration.

2. Prior Work

After an impactful work by Bollen et al. (in which the authors utilized the GPOMS sentiment metric on tweets to achieve an accuracy of 87.6% in predicting the daily up and down changes in the closing values of the DJIA), many more have come into existence when it comes to using social media data to predict the financial market of some sort. Specifically with regards to the cryptocurrency market, there have been works by Stenqvist and Lönnö as well as Colianni et al., in both of which the price fluctuation of Bitcoin has been explored using data from Twitter. In the former, the authors experiment with a substantial number of parameters (e.g. using 7 different time intervals with 200+ threshold parameters on sentiment changes) and thus boasts 83% accuracy with their top performing set of parameters. Meanwhile, the latter focuses on applying 3 different supervised machine learning algorithms (namely, naive Bayes, logistic regression, and SVM) on 2 separate sets of features (i.e. one embodying one hot encodings of tweets and one embodying sentiment feature vectors representing tweets).

While impressive, these works do not consider Ethereum and its blockchain data whereas, in addition, their methodologies around scraping all possible tweets inevitably suffer from a huge deal of noise and inefficiency. Additionally, rather questionable is the replicability of their results due to the short time frame (a month or less) of their data.

In fact, this point is exactly the one that is broached by Abraham et al. in their paper, where the authors claim that previous works were mostly done when the price of Bitcoin had been on a rising trend such that the usual non-negativity of cryptocurrency-related tweets (meaning most cryptocurrency-related tweets are in fact more non-negative than not) incidentally correlated with Bitcoin's rising price. Furthermore, the main argument of their paper is that the volume of

tweets matters more than the sentiment of tweets, which is why we choose to incorporate both whenever possible in our search.

In this paper, we explore the regression problem (i.e. outputting a future price) and the binary classification problem (i.e. predicting whether or not the price will go up tomorrow) of time series forecast using techniques such as GBT (gradient boosted trees), SVM, NN (neural network), and LSTM. In addition, the novelty is derived from the fact that we perform analysis on tweets from key influencers as opposed to everyone (saving a ton of data storage and compute power as a result).

In short, our contributions (or at least attempts) span the following:

- Exploring the regression problem while remarking on the importance of a rigorous error analysis framework
- Comparing the performances of GBT, SVM, NN, and LSTM
- Implementing a system to retrieve and update tweets from key influencers (or any specified group of Twitter accounts as desired)
- Examining Ethereum's blockchain data

3. Data

Price data and blockchain data have been retrieved from Etherscan. They take the following form when retrieved:

Date(UTC)	UnixTimeStamp	Value
7/30/2015	1438214400	11.5297
7/31/2015	1438300800	51.4594
8/1/2015	1438387200	57.7845
8/2/2015	1438473600	67.9224
8/3/2015	1438560000	74.5737
8/4/2015	1438646400	82.0352
8/5/2015	1438732800	86.1558
8/6/2015	1438819200	88.3326
8/7/2015	1438905600	95.2981
8/8/2015	1438992000	104.8971
8/9/2015	1439078400	109.6916
8/10/2015	1439164800	120.5988
8/11/2015	1439251200	134.7695

Figure 1: One of the features of blockchain data (in raw form)

After retrieving 100 Twitter handles of top Crypto Twitter influencers from CryptoWeekly (the process of which involved scraping the web using Python), we retrieved more than 200,000 tweets from those influencers directly from Twitter using Python's Tweepy. In order to allow future retrieval of data with ease, we additionally implemented a way to prepend (as opposed to append due to the reverse-chronological nature of our Twitter data) future data to the existing data in a seamless manner.

Figure 2: Twitter handles from CryptoWeekly

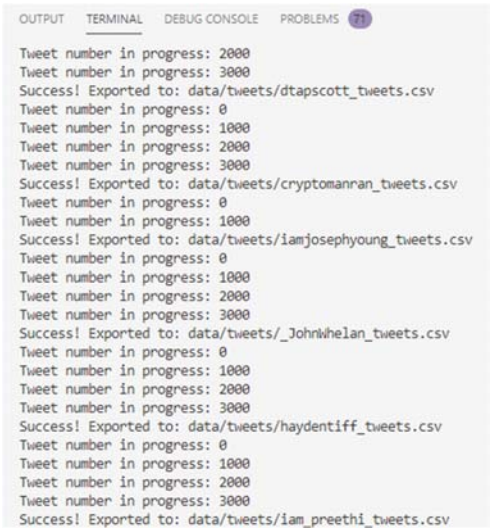

Figure 3: Utilizing Tweepy to retrieve tweets

As for blockchain data, data preprocessing involved removing and adding columns within a spreadsheet (as some spreadsheets contained useless columns such as UnixTimeStamp), joining multiple spreadsheets into one spreadsheet, and performing min-max normalization to each feature.

From there, we applied a Pearson correlation threshold of 0.8 to remove some features, i.e. MarketCap, NetworkUtilization, GasUsed, BlockSize, and Supply. This choice of features was sensical, as certain features (like NetworkUtilization and GasUsed) can be proxied by one feature (like TransactionFee) in practice. At the same time, some feature pairs like (NetworkHash, TxGrowth) and (AddressCount, BlockReward) were kept as they are (even when a Pearson correlation higher than 0.8 in absolute value was yielded), as their relationship tend to be a bit more nuanced in theory in the context of the Ethereum network.

As for data from Twitter, six columns of data were gathered per Twitter account: username, Twitter post id, time, number of likes, number of retweets, and the tweet itself. After removing handles (@) and links (text that start with http) from tweets, we performed sentiment analysis on them. Specifically, we performed VADER (Valence Aware Dictionary and sEntiment Reasoner), which is a combined lexicon and rule-based sentiment analysis framework developed by Hutto and Gilbert. As it is capable of outputting the sentiment intensity of a tweet from a scale of -1 (negative) to 1 (positive), it was originally developed as a solution to the difficulty in analyzing the language, style, and symbols used in the realm of social media. As it has been known for its consistent and competitive performance, we chose VADER as our sentiment analysis tool as well.

```
"username","id","time","likes","retweets","text"
"adam3us","1261083674474414080","2020-05-14 23:59:12","2","0","@cryptoSuperGuy1 @MultiBitcoiner @nvk And bitshares."
"adam3us","1261069427975561221","2020-05-14 23:02:35","20","1","@udiWertheimer They should join and pay attention in your pro eth educational ch
"adam3us","1261068906560720898","2020-05-14 23:00:31","7","0","@Konfidinse @nvk Isn't it just. Ironic what a scammer Larimer turned into later.
"adam3us","1261067853039652872","2020-05-14 22:56:20","32","0","@Konfidinse @nvk Serial scam coiner Dan Larimer."
"adam3us","1261066780182749185","2020-05-14 22:52:04","21","0","@nvk and who he said it to also :)"
"adam3us","1261065766583050249","2020-05-14 22:48:02","3","0","@wtogami @lucas_lclc @HillebrandMax hydro i guess. switzerland has abundant hydro
"adam3us","1261063895055241216","2020-05-14 22:40:36","4","1","@davidgerard @ahcastor @CoinDesk play silly games, win silly prizes. \n\nthey sho
```

Figure 4: Extracted raw tweets

In order to create a time series out of our tweets, we then indexed all tweets from all influencers by a daily time frame and proceeded to include as daily features: the number of likes per day, the number of retweets per day, the sum of sentiment scores per day, the total counts of influencers' tweets per day, the average number of likes per post per day, the average number of retweets per post per day, and the average sentiment score per post per day. As the relevance of average numbers (e.g. average sentiment score) was deemed higher than that of total numbers, the latter 3 as well as total counts per day (to measure volume as a factor) were included in the final set of features.

	A	B	C	D	E	F	G	H
1	time	likes	retweets	sentiment	count	likes_avg	retweets_avg	sentiment_avg
4353	4/14/2020	38374	301556	89.5793	763	50.29357798	395.2241153	0.117404063
4354	4/15/2020	32203	198645	54.7112	867	37.14302191	229.1176471	0.063104037
4355	4/16/2020	76063	119844	96.1131	872	87.22821101	137.4357798	0.110221445
4356	4/17/2020	39818	85405	121.7457	870	45.76781609	98.16666667	0.139937586
4357	4/18/2020	78167	185968	110.8719	698	111.987106	266.4297994	0.158842264
4358	4/19/2020	55047	204392	70.9192	601	91.59234609	340.0865225	0.118001997
4359	4/20/2020	83162	218829	63.66	925	89.90486486	236.5718919	0.068821622
4360	4/21/2020	70940	49401	139.6101	855	82.97076023	57.77894737	0.163286667
4361	4/22/2020	134498	254263	161.0462	790	170.2506329	321.8518987	0.203855949
4362	4/23/2020	77535	232817	127.2923	779	99.53145058	298.8664955	0.16340475
4363	4/24/2020	63417	70630	122.0794	666	95.22072072	106.0510511	0.183302402
4364	4/25/2020	198508	90708	72.6282	778	255.151671	116.5912596	0.093352442
4365	4/26/2020	59715	160526	49.7134	683	87.43045388	235.0307467	0.072786823
4366	4/27/2020	61205	133210	66.0831	628	97.46019108	212.1178344	0.105227866
4367	4/28/2020	65987	126846	146.8603	950	69.46	133.5221053	0.154589789
4368	4/29/2020	78016	137240	120.7946	857	91.03383897	160.1400233	0.140950525
4369	4/30/2020	96228	176622	136.9247	894	107.6375839	197.5637584	0.15315962
4370	5/1/2020	67966	324196	154.6093	925	73.47675676	350.4821622	0.167145189
4371	5/2/2020	95368	107688	94.5771	735	129.752381	146.5142857	0.128676327
4372	5/3/2020	53818	54609	100.5212	607	88.66227348	89.96540362	0.165603295
4373	5/4/2020	44860	46452	108.2919	740	60.62162162	62.77297297	0.146340405
4374	5/5/2020	55022	37434	101.6696	816	67.42892157	45.875	0.124595098
4375	5/6/2020	47828	130037	104.4094	614	77.89576547	211.786645	0.170047883

tweets_influencers

Figure 5: Processing data from Twitter to a time series form

Joining the above four features with those from blockchain data, we obtained the final set of features with the following 11 features (in addition to Price[1]): TxGrowth, AddressCount, BlockTime, AvgGasPrice, NetworkHash, TransactionFee, BlockReward, tweet counts, average likes, average retweets, and average sentiment score.

[1] For binary classification, the variable name PriceChange was used to denote 1 for up change in price and 0 for down change.

	Price	TxGrowth	AddressCount	BlockTime	AvgGasPrice	NetworkHash	TransactionFee	BlockReward	count	likes_avg	retweets_avg	sentiment_avg
2015-08-07	2.77	0.000535	0.000000	0.215844	0.640764	0.000000	0.005431	0.652755	10.0	11.900000	437.000000	0.13222(
2015-08-08	0.81	0.001151	0.000005	0.203789	0.338307	0.000032	0.010687	0.672025	13.0	32.923077	83.538462	0.11882
2015-08-09	0.74	0.000000	0.000008	0.227899	0.502159	0.000049	0.001467	0.642873	16.0	53.187500	168.875000	0.19225(
2015-08-10	0.68	0.000525	0.000015	0.203789	0.444437	0.000086	0.004381	0.667221	15.0	16.200000	53.266667	0.12918
2015-08-11	1.06	0.002695	0.000036	0.199770	0.075641	0.000133	0.000991	0.675240	17.0	33.411765	52.176471	0.03373
...	...	...	...	...	...	...	...	...	...	...	...	...
2020-03-20	133.40	0.608670	0.993622	0.019518	0.009696	0.588065	0.122654	0.125918	751.0	79.743009	1250.122503	0.14323
2020-03-21	132.72	0.634880	0.995825	0.013777	0.005464	0.608523	0.087398	0.132681	781.0	117.144686	1083.900128	0.10709
2020-03-22	122.44	0.552341	0.997188	0.036165	0.006030	0.578340	0.084486	0.122077	743.0	66.362046	513.563930	0.04657
2020-03-23	136.74	0.593157	0.998820	0.011481	0.007664	0.591897	0.099860	0.134314	735.0	107.609524	342.579592	0.07772
2020-03-24	136.77	0.586233	1.000000	0.026980	0.006829	0.601344	0.094911	0.125125	733.0	99.098226	1475.869031	0.13232

1692 rows × 12 columns

Figure 6: Ethereum price and 11 features in a spreadsheet form

To demonstrate each feature as a time series, we include the following plots.

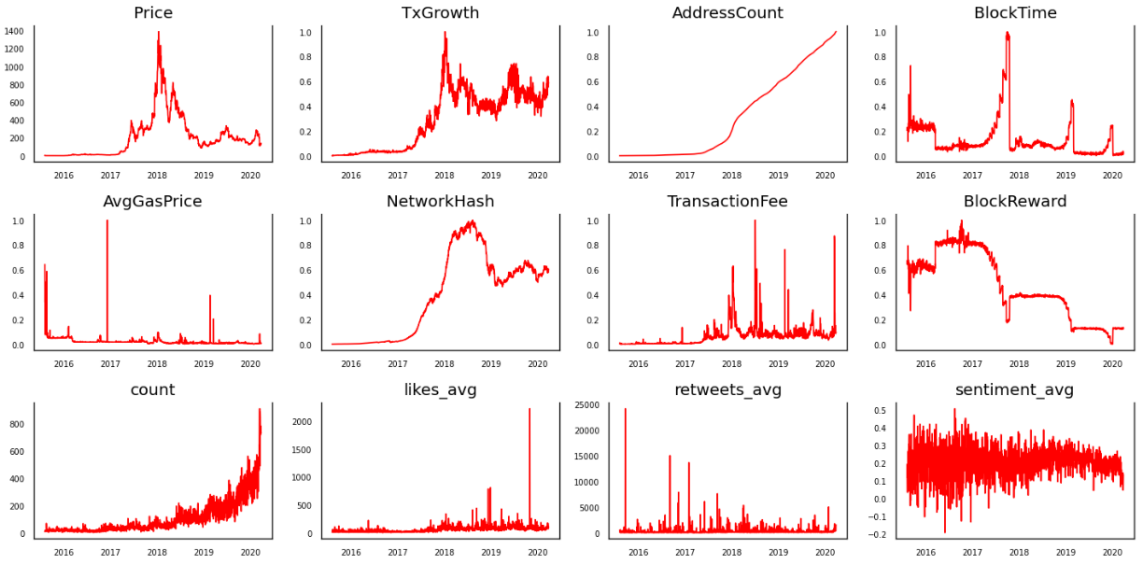

Figure 7: Plots of Ethereum price and all features

4. Methods

4.1. Traditional time series modeling

As the control case, we have performed traditional time series modeling (i.e. moving average, Holt-Winters, and ARIMA) on our Ethereum price data to begin with. Usually, such traditional time series models do tend to work well on traditional time series data. As our Results section

would show, however, we verify that our (perhaps unorthodox) Ethereum price data certainly pose challenges for these traditional models.

As for moving average, we experimented with 3 different time windows, i.e. 7 days, 30 days, and 100 days. Here, moving average refers to simple moving average, which takes the average of past n number of days in order to predict today's price.

As for Holt-Winters, we show that assuming a 30-day season yields a better result compared to assuming a quarterly (90-day) season, although any assumption of seasonality is in fact not quite foundational in the context of a volatile cryptocurrency market. This lack of seasonality can be seen in the ACF (autocorrelation function) plot of our time series, as a matter of fact.

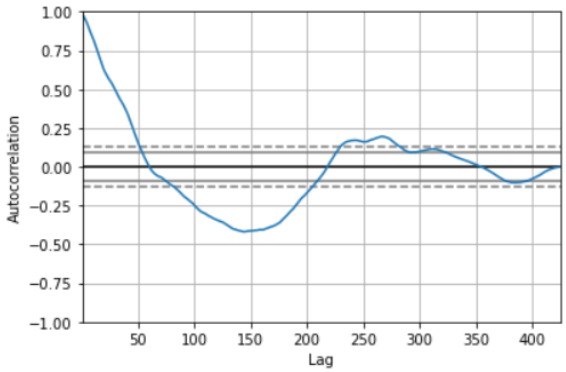

Figure 8: ACF plot of our time series

Usually, a time series that contains seasonality includes "peaks" every seasonal time frame in its ACF plot, meaning the graph would have noticeable bumps every (say) 7 days if the length of a season is 7 days. In the above graph, however, we fail to observe any obvious repeating bumps in the graph. Hence, we conclude there does not exist an obvious notion of seasonality in the time series of Ethereum's price data.

As for ARIMA, we attempt to search for our optimal ARIMA parameters p, d, and q as per ARIMA(p,d,q) both manually and automatically. In terms of an automatic search, we make use of Python's pmdarima library to yield ARIMA(0,1,0), in which case our parameters p (i.e. parameter that corresponds to the AR, autoregressive, portion of ARIMA) and q (i.e. parameter that corresponds to the MA, moving average, portion of ARIMA) are equal to zero. As this is definitely not ideal, we resort to manually taking a look at the PACF (partial autocorrelation function) plot alongside the aforementioned ACF plot of our time series.

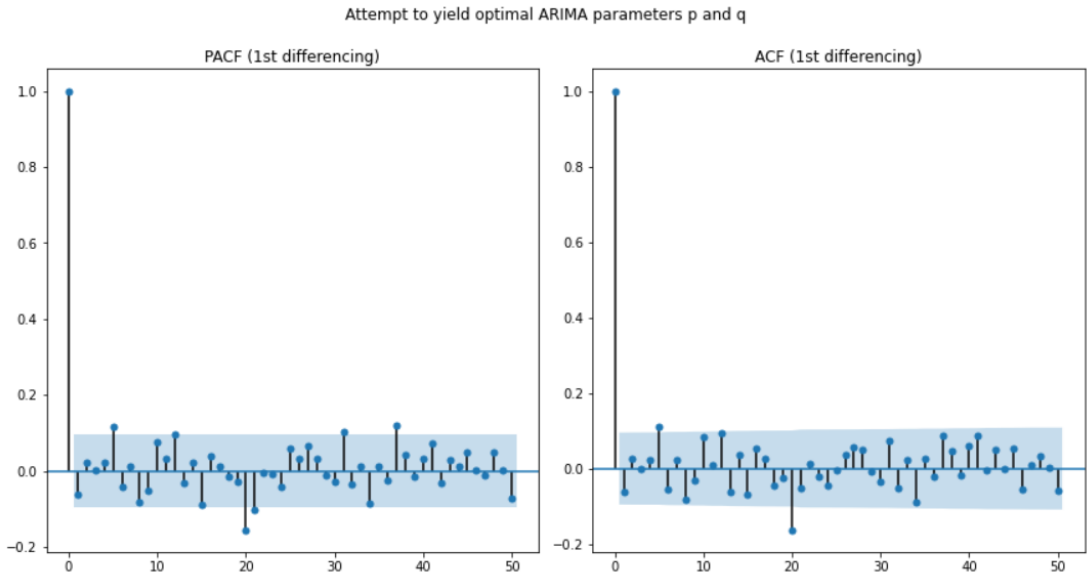

Figure 9: PACF and ACF plots of our time series to manually search for optimal ARIMA parameters

While the maximum x-value (starting from 0) that is above the shaded region in the PACF plot (or the ACF plot) above corresponds to the parameter p (or q), it is unfortunately noticeable that both PACF(x) and ACF(x) for x > 0 tend to stay within the confidence interval (i.e. the shaded region).

Due to this, we finally resort to manually experimenting with parameters p and q while allowing d to equal 1. We choose (p,d,q) to be (3,1,1), (7,1,1), (14,1,1), (21,1,1), or (28,1,1) in addition to the aforementioned (0,1,0). In the end, we only include results for when (p,d,q) equals (21,1,1) due to the lack of fruitful results otherwise.

On a different note, we also explore the notion of Granger causality out of curiosity. A statistical concept of causality, Granger causality determines whether or not the two probabilities are equal: one forecasting values of time series variable Y knowing another time series variable X in previous time inputs, and one forecasting values of Y without knowing X in previous time inputs. If X does not "cause" Y, then knowing X in previous time inputs would not do anything to the above probability, whereas we say X "granger-causes" Y if the above two probabilities are unequal.

In short, we observe the following (obtained by utilizing Python's statsmodels library and included here instead of the Results section due to its lack of full relevance in forecasting):

- TxGrowth, AddressCount, and TransactionFee granger-cause Price.
- Total number of tweets per day granger-causes TxGrowth, AddressCount, and TransactionFee. If this were a direct causation, we would possibly interpret it as that tweet volume by influencers leads to busy activities (i.e. more transactions and more generated addresses by participants) of the Ethereum network.
- The average number of likes granger-causes AddressCount and TransactionFee.

- The average likes, the average retweets, and the average sentiment score granger-cause the total number of tweets. Perhaps, this can be seen as a byproduct when major cryptocurrency-related news (quite frequently) hit the market, as the market is perhaps a highly news-driven one.

4.2. Regression and binary classification using RapidMiner

RapidMiner's Optimize Parameters functionality has been used extensively when it comes to fine-tuning the (hyper)parameters for the 4 machine learning algorithms used in this paper: GBT, SVM, NN, and LSTM. While GBT, SVM, and NN are native to RapidMiner, one needs to install RapidMiner's Deep Learning extension package (which is based on Java's Deeplearning4j) in order to effectively utilize LSTM as an algorithm. (For LSTM, an operator similar to Optimize Parameters, called Loop Parameters, has been used in practice. The idea is the same.)

The following summarizes the range of parameters that have been experimented with for the purpose of this paper (while one can ideally experiment with as many parameters as possible, of course):

- GBT: 30 combinations
  - Number of trees: 20, 40, 60, 80, 100
  - Learning rate: 0.05, 0.1, 0.15, 0.2, 0.25, 0.3
- SVM: 153 combinations
  - Kernel type: dot, radial, polynomial
  - C: 0-1000 with 50 quadratic steps in between
- NN: 210 combinations
  - Training cycles: 200, 650, 1100, 1550, 2000
  - Learning rate: 0.001, 0.003, 0.01, 0.032, 0.1, 0.316, 1
  - Momentum: 0.1, 0.26, 0.42, 0.58, 0.74, 0.9
- LSTM[2]: 180 combinations
  - Number of LSTM layer neurons: 8, 21, 64
  - Activation function: ReLU, sigmoid, tanh
  - Updater: Adam, RMSprop
  - Learning rate: 0.001, 0.003, 0.01, 0.03, 0.1
  - Regularization (L2): true, false

[2] Mean squared error loss was used for regression. Cross entropy loss was used for classification. Training was done for 200 epochs per set of parameters.

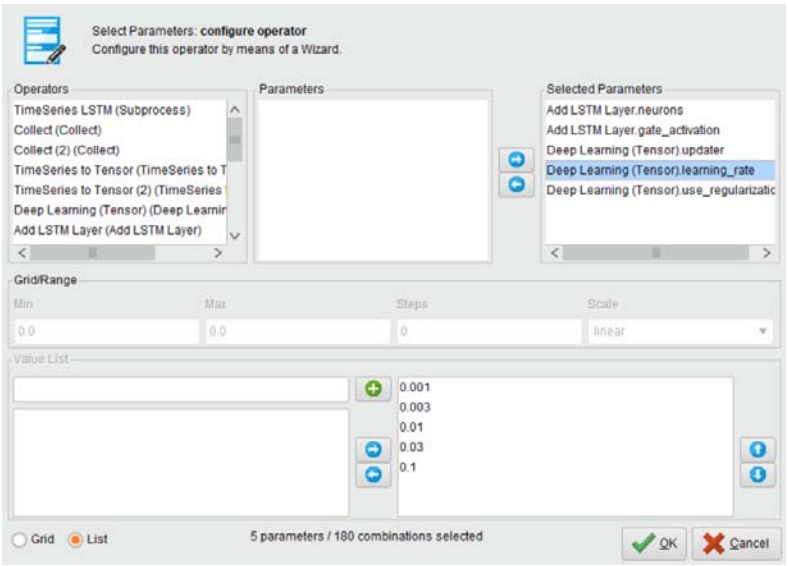

Figure 10: RapidMiner's Optimize Parameters operator

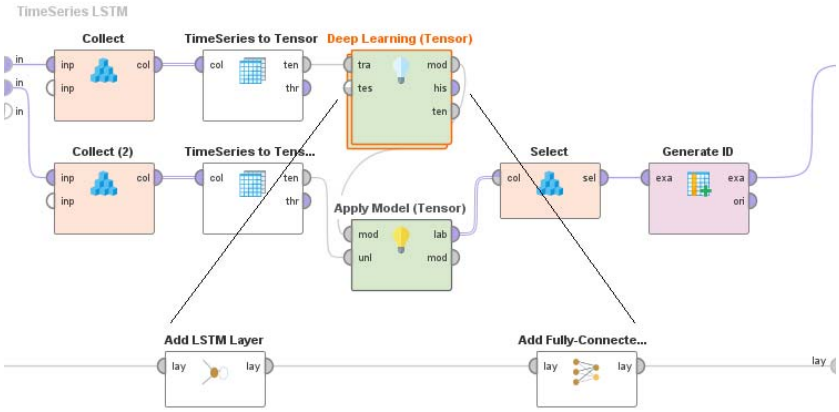

Figure 11: Our simple LSTM architecture

Alongside the Optimize Parameters operator in RapidMiner, we also utilize an operator called Explain Predictions. As the name suggests, this operator is able to explain each prediction our algorithms make by explicitly revealing the weights of features that matter per prediction. It also emits global weights of features given a trained machine learning algorithm, which we broach later.

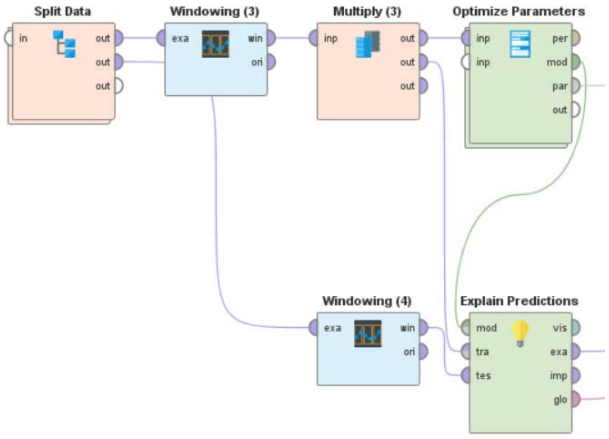

Figure 12: Windowing, Optimize Parameters, and Explain Predictions

One vital remark remains. Because of the fact that our time series depends on being chronological by definition, we need to be a bit cautious with regards to feeding our training data into a machine learning algorithm. To begin with, recall that our preprocessed but raw (i.e. not processed in RapidMiner yet) training data consist of Ethereum's price data and 11 features. What one might end up doing naively is training a machine learning algorithm such that it learns the price of the same day given 11 features. This is not desirable. Instead, one should be able to make predictions on tomorrow's price based on 11 features from today.

Furthermore, what if, in fact, we are able to leverage 11 features from yesterday and 11 features from today -- hence 22 features total -- to predict the price tomorrow? This is where RapidMiner's Windowing operator has been used extensively. As its variable called window size denotes the lag of n days (such that past n days' feature data are considered to make the next forecast), we experimented with window sizes 1, 2, 3, 7, and 10. As changing the window size shifted the training time quite substantially at times, we increased modularity by separating the processes per window size.

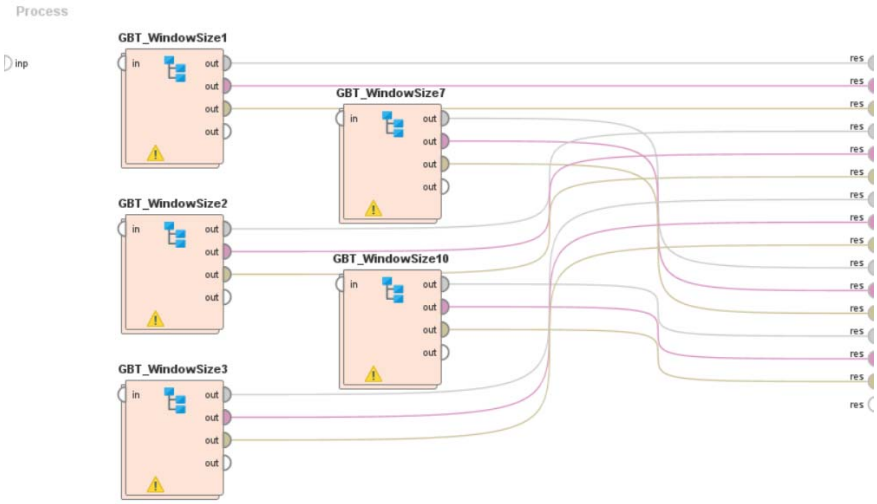

Figure 13: Modularity per window size

By the same token, we note the importance of a coherent, rigorous error analysis framework dealing with time series when it comes to splitting the training data a la validation and testing our machine learning models on out-of-sample test data. Here, two scenarios (which should be mediated) can happen. First, our models can perform extremely well when training but perform very poorly when testing due to a temporal mismatch. For instance, training on data from 2016 wouldn't be smart if testing on data from 2020, so one should always keep this in mind. Second, even if testing yielded favorable results (without any temporal mismatch), it could be possible that the model merely benefited from pure luck such that (say) it performs very well on the second month of 2020 while performing poorly on the first month or the third month.

Of course, the latter point is perhaps somewhat inevitable due to the nature of a cryptocurrency market, let alone any sort of financial market. Nonetheless, how can we minimize this risk such that we choose a model more robust and safer, rather than one that may perform better in the extreme short term but may remain less robust and risky in the long term? As we deal with the nature of time series, we utilize sliding window validation here (denoted by the Sliding Window Validation operator in RapidMiner) as opposed to a more traditional cross validation in which case temporal mismatch would be an issue when validating across different folds.

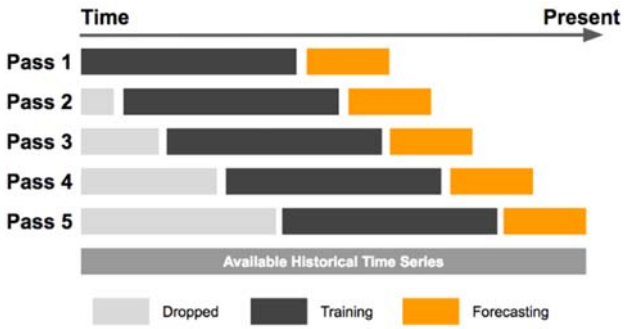

Figure 14: Sliding window validation[3]

Hence, the idea is that we perform sliding window validation within our training sample data, take the average performance across all folds, choose the model that performs best on average (meaning the model was a robust one), and finally test it on out-of-sample data. In our study, we choose 90-30 (days) split for our validation fold, slide the window forward by 30 days, and repeat the process within our training sample data. The training data ranged from January 1, 2019 to December 31, 2019 while the test (out-of-sample) data ranged from January 1, 2020 to March 24, 2020.

The following are performance measures used in this paper:

- RMSE (root mean square error)
- MAPE[4] (mean absolute percentage error)
- MAE[5] (mean absolute error)
- Corr (Pearson correlation)
- Accuracy, precision, recall, F1 score -- used in binary classification

5. Results

5.1. Regression[6]

Again, all data (training, validation, and test) concern dates from January 1, 2019 to March 24, 2020. While testing has been done on data from 2020 for GBT, SVM, NN, and LSTM, it has been done on data from March (only) of 2020 for traditional time series models (but the errors are still bigger as expected despite a shorter time frame). Notice that not all results are shown in the following.

[3] The image is from Uber Engineering.
[4] In RapidMiner, this is denoted by relative error.
[5] In RapidMiner, this is denoted by absolute error.
[6] Note that some results may differ slightly from those presented in the presentation, as more simulations have been done.

	2020-01-01 to 2020-03-24 (out-of-sample data)		Validation Error (avg +/- std dev)
		2020-03-01 to 2020-03-24	
Moving Average (7 days)		RMSE: 79.054 MAPE: 0.281 MAE: 64.866 Corr: 0.227	
Moving Average (30)		RMSE: 93.742 MAPE: 0.316 MAE: 78.771 Corr: -0.611	
Moving Average (100)		RMSE: 50.542 MAPE: 0.275 MAE: 48.597 Corr: -0.820	
Holt-Winters (90)		RMSE: 67.257 MAPE: 0.256 MAE: 55.204 Corr: 0.438	
Holt-Winters (30)		RMSE: 30.225 MAPE: 0.140 MAE: 24.116 Corr: 0.844	
ARIMA(21,1,1)		RMSE: 63.435 MAPE: 0.253 MAE: 52.702 Corr: -0.235	
GBT (window size 1, 100 trees, 0.2 learning rate)	RMSE: 19.520 MAPE: 6.47% MAE: 12.304 Corr: 0.937		RMSE: 27.573 +/- 21.696 MAPE: 10.90% +/- 7.34% MAE: 22.559 +/- 18.259 Corr: 0.494 +/- 0.255
GBT (window size 2, 80 trees, 0.25 learning rate)	RMSE: 17.459 MAPE: 5.52% MAE: 10.341 Corr: 0.944		RMSE: 27.509 +/- 21.833 MAPE: 10.95% +/- 7.31% MAE: 22.748 +/- 18.726 Corr: 0.540 +/- 0.264
GBT (window size 3, 20 trees, 0.3 learning rate)	RMSE: 16.045 MAPE: 5.35% MAE: 9.989 Corr: 0.953		RMSE: 27.073 +/- 22.098 MAPE: 10.80% +/- 7.48% MAE: 22.544 +/- 19.305 Corr: 0.504 +/- 0.250
GBT (window size 7, 40 trees, 0.25 learning rate)	RMSE: 17.811 MAPE: 6.23% MAE: 11.708 Corr: 0.947		RMSE: 30.097 +/- 23.725 MAPE: 11.87% +/- 8.51% MAE: 25.965 +/- 22.372 Corr: 0.381 +/- 0.289
GBT (window size 10, 40 trees, 0.15 learning rate)	RMSE: 18.312 MAPE: 6.06% MAE: 11.720 Corr: 0.944		RMSE: 32.518 +/- 25.952 MAPE: 13.28% +/- 10.07% MAE: 29.178 +/- 25.621 Corr: 0.296 +/- 0.302
SVM (window size 1, dot kernel, C of 40)	RMSE: 14.239 MAPE: 4.95% MAE: 8.581 Corr: 0.958		RMSE: 11.374 +/- 4.223 MAPE: 4.45% +/- 1.45% MAE: 8.941 +/- 3.798 Corr: 0.792 +/- 0.196
NN (window size 1, 1100 cycles, 0.003 learning rate, 0.58 momentum)	RMSE: 14.318 MAPE: 5.30% MAE: 9.008 Corr: 0.957		RMSE: 14.727 +/- 8.416 MAPE: 5.68% +/- 2.51% MAE: 11.704 +/- 7.144 Corr: 0.772 +/- 0.213
LSTM (window size 1, 64 LSTM neurons, sigmoid, RMSprop, 0.01 learning rate, L2 regularization)	RMSE: 16.705 MAPE: 7.13% MAE: 11.849 Corr: 0.958		(due to compute constraints, this step was omitted)

Figure 15: Regression results in terms of RMSE, MAPE, MAE, and Corr

The following delineates (in a descending order) some of the global weights of features per trained machine learning model[7]:

- GBT (window size 3, 20 trees, 0.3 learning rate)
    - Price (-0 day): 0.913
    - likes_avg (-0 day): 0.121
    - BlockTime (-0 day): 0.086
    - TxGrowth (-0 day): 0.072
    - Price (-2 days): 0.066
    - Price (-1 day): 0.065
    - retweets_avg (-0 day): 0.063
    - AddressCount (-1 day): 0.061
    - count (-0 day): 0.053
- SVM (window size 1, dot kernel, C of 40)
    - Price (-0 day): 0.837
    - TransactionFee (-0 day): 0.107
    - NetworkHash (-0 day): 0.064
    - sentiment_avg (-0 day): 0.013
- NN (window size 1, 1100 cycles, 0.003 learning rate, 0.58 momentum)
    - Price (-0 day): 0.997
    - BlockReward (-0 day): 0.075
    - AddressCount (-0 day): 0.046
    - TxGrowth (-0 day): 0.031
    - sentiment_avg (-0 day): 0.017

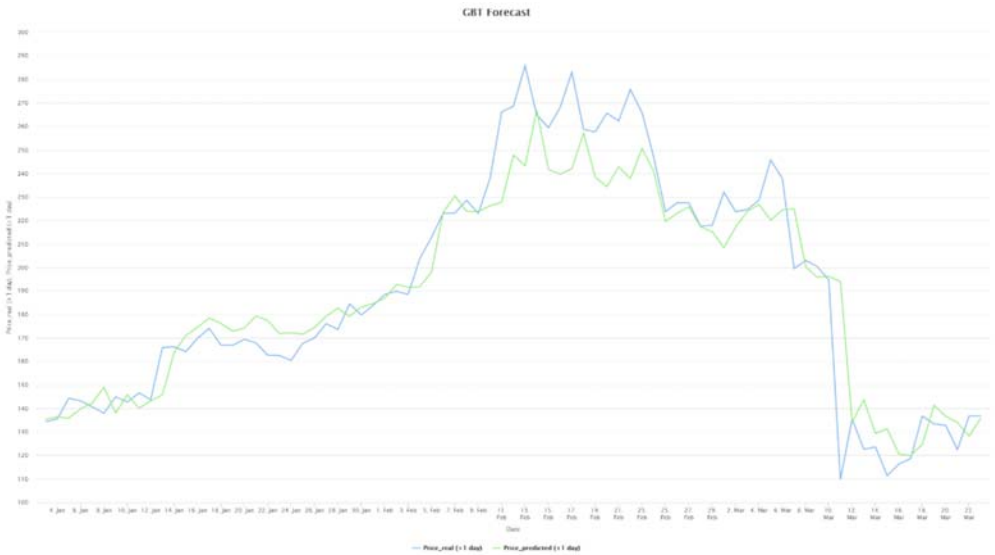

Figure 16: Sample forecast for 2020 (using GBT)

[7] The Explain Predictions operator is not yet available for LSTM.

5.2. Binary classification

	2020-01-01 to 2020-03-24 (out-of-sample data)	Validation Error (avg +/- std dev)
GBT (window size 7, 20 trees, 0.05 learning rate)	accuracy: 57.14% precision: 53.33% recall: 66.67% F1: 59.26%	accuracy: 57.08% +/- 9.50% precision: 59.48% +/- 10.00% recall: 54.59% +/- 25.31% F1: 53.09% +/- 16.77%
SVM (window size 1, polynomial kernel, C of 0.4)	accuracy: 53.01% precision: 48.98% recall: 63.16% F1: 55.17%	accuracy: 55.93% +/- 11.76% precision: 57.36% recall: 54.48% +/- 29.53% F1: 55.43%
NN (window size 3, 200 cycles, 1.0 learning rate, 0.58 momentum)	accuracy: 56.79% precision: 52.78% recall: 51.35% F1: 52.05%	accuracy: 56.67% +/- 12.13% precision: 57.04% recall: 60.75% +/- 34.12% F1: 58.06%
NN (window size 7, 1550 cycles, 1.0 learning rate, 0.26 momentum)	accuracy: 55.84% precision: 53.12% recall: 47.22% F1: 50.00%	accuracy: 60.00% +/- 8.54% precision: 60.07% +/- 8.62% recall: 56.76% +/- 28.29% F1: 54.87% +/- 20.43%
LSTM (window size 3, 21 LSTM neurons, sigmoid, Adam, 0.003 learning rate, L2 regularization)	accuracy: 62.96% precision: 58.54% recall: 64.86% F1: 61.54%	(due to compute constraints, this step was omitted)

Figure 17: Binary classification results

The following delineates (in a descending order) some of the global weights of features per trained machine learning model:

- GBT (window size 7, 20 trees, 0.05 learning rate)
    - PriceChange (-0 day): 0.163
    - TransactionFee (-3 days): 0.12
    - sentiment_avg (-6 days): 0.108
    - BlockReward (-4 days): 0.084
    - likes_avg (-0 day): 0.064
    - PriceChange (-2 days): 0.059
    - sentiment_avg (-3 days): 0.058
    - BlockTime (-0 day): 0.051
- SVM (window size 1, polynomial kernel, C of 0.4)
    - AvgGasPrice (-0 day): 0.235
    - TransactionFee (-0 day): 0.179
    - likes_avg (-0 day): 0.108
    - retweets_avg (-0 day): 0.043
- NN (window size 3, 200 cycles, 1.0 learning rate, 0.58 momentum)
    - PriceChange (-0 day): 0.088
    - retweets_avg (-0 day): 0.081
    - count (-2 days): 0.08
    - count (-0 day): 0.069
    - count (-1 day): 0.065
- NN (window size 7, 1550 cycles, 1.0 learning rate, 0.26 momentum)
    - PriceChange (-4 days): 0.091
    - TxGrowth (-1 day): 0.084

- PriceChange (-5 days): 0.083
- sentiment_avg (-2 days): 0.077
- PriceChange (-3 days): 0.066
- count (-4 days): 0.063

6. Discussion and Conclusion

When it comes to regression, SVM has performed best in terms of RMSE (root mean square error). While so, it is notable that all traditional time series models have performed quite poorly even on a shorter time frame (namely, just the range 2020-03-01 to 2020-03-24 as opposed to 2020-01-01 to 2020-03-24). For instance, Holt-Winters with 30-day seasonality (i.e. the best model among them) scored slightly above 30 in terms of RMSE, whereas GBT with window size 1 (i.e. one of the worse performing models among GBT, SVM, NN, and LSTM) scored below 20.

Also notable is the fact that the window size of 1 (as opposed to 2, 3, 7, or 10) has consistently been the better performing parameter across the 4 machine learning models in the regression problem. Perhaps, this signifies that looking at a window size greater than 1 for regression may not be as necessary as suspected. In simple words, this means that one is able to make predictions on tomorrow's price just using today's data.

When it comes to binary classification, on the other hand, the window sizes of 1, 3, and 7 have been employed by top performing models. Quite notably, LSTM with window size 3 has performed best in terms of accuracy in the case of binary classification problem. In fact, it achieved about 63% in accuracy in predicting every day's price changes for the first 84 days of 2020. Before starting this project in full, we naively hoped that we would be able to achieve more than 70% using LSTM and other methodologies, but it does seem to be the case that time series forecasting is indeed a highly challenging task in hindsight (especially in a volatile market like Ethereum's) and that our 63% (which could certainly be improved further by tuning and fine-tuning) poses an auspicious number at the very least.

It is perhaps worth remarking that our utilized LSTM architecture is in fact quite a basic one amid a myriad of state-of-the-art architectures that are used in industry settings. Despite its simplicity, it has performed competitively. Exploring other recurrent neural network-based architectures for this type of project is definitely in the future line of work.

Regarding weights of features, we note that in the regression problem, the average likes and the average retweets have been weighted nontrivially by GBT while the average sentiment has been weighted more by SVM and NN. In the classification problem, a bit of the reverse occurred such that the average sentiment score was weighted quite highly by GBT. While fascinating to observe these weights of various features, it is indeed curious as to how robust these weights are in the scope of RapidMiner's Explain Predictions operator when it comes to testing dozens of sets of parameters.

Having experimented with hundreds of parameters around 4 machine learning models (i.e. GBT, SVM, NN, and LSTM) as well as the control case utilizing traditional time series models (i.e. moving average, Holt-Winters, and ARIMA), we conclude the paper with notes on questioning the following assumptions of ours that have been taken for granted within the scope of this work:

- Sentiment towards cryptocurrency can be estimated by daily tweets by a selected group of Twitter influencers due to the prevalence of Crypto Twitter.
    - Does the specific set of Twitter accounts matter? Perhaps, one does not need to rely on public rankings (like CryptoWeekly) but can explore other sets of Twitter accounts.
    - Can we incorporate non-crypto Twitter accounts (e.g. Trump, Elon Musk, etc.) that may still be of relevance?
    - Can we incorporate other sources (e.g. other social media) for sentiment?
    - How about sources related to cryptocurrency news (e.g. CoinDesk)?
- The price of Ethereum affects and is affected by Ethereum's network status.
    - While theoretically linked by the complex game-theoretic dynamics within the Ethereum ecosystem, what if Ethereum's price and its network data are not as related as we deem? Perhaps, pure speculation is the most important factor in determining Ethereum's price.
- Our sliding window validation procedure is able to determine whether or not a trained model is more robust than others.
    - Is a 90-30 split per validation fold justifiable? Can we experiment with various folds with different splits?
- We have used 365 days of 2019 as training (as well as validation) data and 84 days of 2020 as test data.
    - Is it optimal to use as many (or as little) as 365 days to train models? Perhaps, data from a year ago may end up injecting more noise than guidelines.
- We have adopted VADER to measure sentiment.
    - Can there be other metrics other than VADER? Is there a need to develop a domain-specific lexicon for fine-tuning purposes?
- RMSE has been used as the major error metric for the purpose of performance analysis, as it is most commonly used in the context of time series.
    - As models have learned to minimize RMSE, it has become apparent in some cases that predictions made by these models tend to output suboptimal forecasts, in the sense that one is not able to make profitable trades using these models in practice. In the scope of this goal, can there be a more suitable way to analyze errors and train models?

While the above list is certainly not complete, it sheds light on how complicated time series forecasting can be and how we have attempted, at least in the scope of our assumptions, to predict Ethereum's price utilizing GBT, SVM, NN, and LSTM. Ending the paper, we remark that this issue of predicting the price of Ethereum (or any other type of traded goods) is likely to be an everlasting one due to its volatility and unpredictability, which simultaneously is why there is value, interest, and a degree of excitement in endeavoring to analyze such time series.

References

John Bollen, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." Journal of computational science 2, no. 1 (2011): 1-8.

Evita Stenqvist and Jacob Lönnö. "Predicting Bitcoin price fluctuation with Twitter sentiment analysis." (2017).

C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International AAAI Conference on Weblogs and Social Media, 2014.

Jethin Abraham, Daniel Higdon, John Nelson, and Juan Ibarra. "Cryptocurrency price prediction using tweet volumes and sentiment analysis." SMU Data Science Review 1, no. 3 (2018): 1.

Stuart Colianni, Stephanie Rosales, and Michael Signorotti. Algorithmic trading of cryptocurrency based on twitter sentiment analysis. 2015. http://cs229.stanford.edu/proj2015/029_report.pdf

Hong Kee Sul, Alan R Dennis, and Lingyao Ivy Yuan. Trading on twitter: Using social media sentiment to predict stock returns. Decision Sciences, 2016.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis.

Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. Foundations and Trends R in Information Retrieval, 2(1–2):1–135, 2008.

CoinMarketCap. https://coinmarketcap.com/currencies/ethereum/

Etherscan. https://etherscan.io/charts

CryptoWeekly. The 100 Most Influential People in Crypto 2020 Edition. https://cryptoweekly.co/100