Study of the flows on the Ethereum Blockchain

Explore the data of the Ethereum blockchain, which is public in nature, in order to understand the flows and quantify their impact on the price of the cryptocurrencies circulating on it.

MELANIONCAPITAL

Published on:

10/21/2019

Author:

Clément Goubet

Contents

1. THE PROJECT

1.1. General Objective

The purpose of the project is to explore the data of the Ethereum blockchain, which is public in nature, in order to understand the flows and quantify their impact on the price of the cryptocurrencies circulating on it.

This project is based on two main axes:

The development of a tool to extract and analyze the flows. The aim is to extract from a very large amount of data all the cryptocurrency flows. Among the 2.7 TB of data in July 2019 for an archive node, only the transactions that correspond to payments will interest us. The executions of smart contracts can be ignored, except for those that correspond to a transfer of ERC20 tokens, for the many crypto-currencies that live on the Ethereum blockchain. Once the flows are extracted, a categorization of the addresses is performed, in order to identify the inputs to the exchange platforms as well as the outputs.

The establishment of a price prediction model based on the inflows and outflows of these exchanges. The goal is to anticipate the direction that the price will take at different time horizons according to the net volume accumulated over a period of time at the entry and exit of the exchanges.

1.2. Scientific uncertainties and technical constraints

The first uncertainty of this project is its atypical aspect. We have not found any papers in the literature that focus on the analysis of flows on the Ethereum blockchain. As we will see in the state of the art below, the Bitcoin blockchain has been explored more by the scientific community. Therefore, we do not have anything to compare our results to.

The second constraint we encountered is the pseudonymous nature of the addresses. In order to analyze the flows, we need to identify all the addresses belonging to the exchanges in a precise and comprehensive manner.

The last main constraint is a technological one. It is now difficult to perform the initial synchronization of a full Ethereum node and then to extract information through the RPC (Remote Procedure Call) interface. This constraint must be overcome, both by minimum hardware requirements with a

minimum 500G SSD and a high-speed connection, and by software adjustments based on parallel processing and adaptation of data structures.

2. STATE OF THE ART

The analysis of decentralized public blockchain transactions is relatively recent, due to the youth of this technology. The first conceptualization that gave rise to a viable implementation was done by Satoshi Nakamoto in 2008 in the Bitcoin whitepaper [2]. The Ethereum chain as for appeared in 2014 [3]. An interest in flow analysis developed only later, driven in particular by American regulatory or repressive authorities who wanted to track and de-anonymize the money flows circulating on the Dark Net. Fergal Reid and Martin Harrigan [4] were among the first to publish academically in 2011 the limitations of the anonymity provided by the Bitcoin chain.

The Bitcoin chain does not have addresses as such, but UTXOs (Unspent Transaction Outputs) which are destroyed in their entirety when spent. So, there is no notion of a fixed wallet address as there is for the Ethereum chain. The anonymity provided by this continuous flow without reusing public keys is only apparent because it is possible to link UTXOs together by the user's behavior: if they are spent in the same transaction, there is a strong chance that they belong to the same person or entity. Thus, heuristics for analyzing the transaction graph on the Bitcoin chain have emerged. Harrigan and Fretter [5] analyze the occurrence of address clusters and the causes that make deanonymization possible. Fleder, Kester and Pillai [6] introduce data external to the chain, publicly present on the web, such as interactions on forums, to establish links with real entities, although sometimes still pseudonymous).

Compared to the recent but strong interest of the scientific community in analyzing flows on Bitcoin, few papers have focused on analyzing flows on Ethereum. A 2018 study [7] distinguishes the different uses of Ethereum, currency transmission, creation and invocation of smart contracts but only makes a superficial analysis of the transaction graph (number of nodes and edges, degrees of connectivity of nodes, etc.) without focusing on the temporality of the flows generated by the different actors in the chain.

Regarding price prediction models, papers such as [1] and [10] take the step of incorporating raw blockchain data to improve their models, both for Bitcoin and Ethereum. However, the integrated data either has little to do with short-term price variations (uncle rate or number of orphaned blocks, number of unique addresses used during the day, total supply) or is a lagging indicator of price (hash rate, miners revenue, difficulty) or volatility (trade volume, cost per transaction)

Our approach seeks to return to the fundamentals of pricing by studying the evolution of supply and demand, using the flows in and out of the exchanges as a proxy.

Summary - The state of the art is very much oriented towards the Bitcoin blockchain and few analyses focus on Ethereum. Moreover, there is a lack of heuristic methods for labeling non-public addresses, similar to what exists on the Bitcoin blockchain. Finally, no publication aggregates the incoming and outgoing flows of exchanges to relate them to the price evolution.

3. R&D WORK

The research process was broken down as follows: First, the chain had to be synchronized with the rest of the Ethereum network, and then all the essential information had to be extracted in a more suitable format than the raw chain. Then, we labelled the addresses of the exchanges, allowing us to extract the flows in and out of these platforms. Finally, we established a price prediction model based on these flows.

3.1. Synchronization of the blockchain

The initial synchronization of a public blockchain is a crucial step. Indeed, a newcomer has no knowledge of the truth about all the transactions that took place since the origin of the chain. He has to trust his pairs, who provide him with the history of the blocks, not to provide him with an erroneous history. One way to reduce this need for trust is to check the entire block history. If no rule violates the consensus, if all transactions are properly signed and if the hash of each block is valid and respects the required difficulty, it will be infinitely unlikely that all the nodes of the network with which it interacts could have provided it with such a long history without it being the actual truth. This property comes from the Proof of Work, which ensures that energy was used to create each of the blocks in proportion to the difficulty.

It is important to perform a full chain check if you want to receive large financial transactions. In our case, it was less crucial to perform the verification of the entire chain, which can take more than 2 weeks. There are two main implementations of an Ethereum node: Geth and Parity. We used Parity and used the default synchronization mode (warp mode) which does not check the beginning of the chain and uses a series of snapshots provided by the other nodes of the network to speed up the synchronization. The check is only performed on the most recent part of the chain. Using 16GB of RAM, a 100Mb/s connection and disabling all other node features, the synchronization was completed in 2

days. Removing the state history (address balances or public variables declared in the smart contracts) resulted in 190GB of compressed data on disk.

```
2019-08-13 10:33:54  Starting Parity-Ethereum/v2.5.6-stable-ff398fe-20190812/x86_64-linux-gnu/rustc1.36.0
2019-08-13 10:33:54  Keys path SSD2/ETHEREUMDATA//keys/ethereum
2019-08-13 10:33:54  DB path SSD2/ETHEREUMDATA//chains/ethereum/db/906a34e69aec8c0d
2019-08-13 10:33:54  State DB configuration: fast
2019-08-13 10:33:54  Operating mode: active
2019-08-13 10:33:54  Configured for Ethereum using Ethash engine
2019-08-13 10:33:57  Updated conversion rate to Ξ1 = US$209.14 (22768980 wei/gas)
2019-08-13 10:34:00  Public node URL: enode://8d14ed14d8eb402ee4d69596a97762e9be3b63f7d94a39e8641d8b1a05fa2e87e5263ad1618c224605897c1353e52617e4b1cd689590b64f70c86f0b8771e6fe@192.168.14.92:30303
2019-08-13 10:34:03  Imported #8341131 0x729a…a600 (124 txs, 8.00 Mgas, 1089 ms, 20.00 KiB) + another 2 block(s) containing 331 tx(s)
2019-08-13 10:34:22  Imported #8341132 0xfd85…ef04 (221 txs, 8.01 Mgas, 389 ms, 35.67 KiB)
2019-08-13 10:34:25  Imported #8341132 0xab28…d792 (145 txs, 8.00 Mgas, 430 ms, 24.69 KiB)
2019-08-13 10:34:26  Reorg to #8341133 0x08cb…323e (0xfd85…ef04 #8341131 0x729a…a600 0xab28…d792)
2019-08-13 10:34:27  Imported #8341133 0x08cb…323e (199 txs, 8.00 Mgas, 506 ms, 32.49 KiB)
2019-08-13 10:34:27    21/25 peers   400 KiB chain 12 MiB db 0 bytes queue 23 KiB sync  RPC:  0 conn,    0 req/s,    0 µs
2019-08-13 10:34:30  Imported #8341134 0xca90…28eb (176 txs, 7.99 Mgas, 365 ms, 34.07 KiB)
```

Figure 1 - Synchronized Ethereum node using the Parity v2.5.6 client

3.2. Data extraction

Once the blockchain is synchronized, the data is in compressed format and suitable for synchronization and verification; the transaction list is not directly available and exportable. Accessibility is done through Remote Procedure Calls or RPCs: the Ethereum node exposes an HTTP server to requests such as "retrieve the ids of the transactions in block n" or "retrieve the data corresponding to the transaction of id x". We need all the information {blockNumber, from, to, quantity} for each transaction. One HTTP request per block as well as one per transaction gives about 540 × 106 requests. The alternative of using online APIs such as those proposed by infura.io immediately appeared impossible in view of the quantity of requests to be made. It was therefore necessary to extract all the data locally. Parity offers the possibility to answer RPCs in parallel. The number of threads was experimentally set to 16 (2 times the number of CPU cores) in order to speed up the extraction, while limiting the overhead.

Transactions - The transactions were extracted in clusters of 1000 blocks before being written to disk. This allows me to use less RAM and avoid losing too much work done in case of an unexpected interruption. Only the data {blockNumber, from, to, quantity} are kept; the signatures and the data needed to execute smart contracts are ignored.

Timestamps - In order to link the flow data to the price data, it is also necessary to retrieve the timestamps associated with each block. Although declarative, the entity producing the block has a great deal of flexibility on the timestamp value it declares, this information on the timing of blocks is relatively reliable. The blocks are also produced at irregular intervals according to a Poisson distribution, but the average is about 14 seconds between blocks. This provides a sufficiently regular temporal anchor so that price variations between two blocks are negligible. The extraction is done in packets of 10000 blocks before writing to disk.

ERC20 transfers - We looked at 48 ERC-20 tokens, including Augur, Basic Attention Token, Chainlink, Golem, Icon, Zilliqa and 0x. The ERC20 standard corresponds to the required presence of a certain number of operations in the smart contract. As an example, the following functions must be implemented:

```
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
function transfer(address _to, uint256 _value) public returns (bool success)
```

Once the ERC-20 token is identified by its contract address, for example 0x1985365e9f78359a9B6AD760e32412f4a445E862 for Augur, it is not necessary to scan the string before the contract creation block (5926311 for Augur). Parity gives the possibility to create filters to extract events. We can thus efficiently extract all transfers of an ERC-20 token between two given blocks. The extraction has been done by packets of 10000 blocks.

3.3. Address labeling

Which *addresses*?

The addresses are pseudonymous, but we can assign a category to some of them. They can be :

- Exchanges,
- Mining Pools,
- ICO Treasuries (Ether funds raised by ICOs in exchange for their token),
- investors or ordinary users...

Among this non-exhaustive list, the easiest to identify are the miners because of the reward produced with each block. The category we are particularly interested in is the Exchanges. Some addresses have a label on the online explorer https://etherscan.io/ but these labels do not cover all the addresses of the exchanges.

3.4. Establishment of an analysis tool

In order to manually analyze the different addresses of the exchanges, it was necessary to develop an analysis tool. Indeed, to establish statistics about an address x, we have to run through the 56G of transaction data {blockNumber, from, to, quantity} to determine which ones involve x. We therefore chose to precompute 3 new data structures. This involved aggregating by 1) {from, to}, 2) {from} and 3) {to} and computing for each key the following statistics:

- Net sum of movements,
- Amount of the average movement,
- Standard deviation of the amounts,
- Number of transactions,
- Block of the first transaction,
- Block of the last transaction.

In order to be able to perform Merge-Sort operations on the 56G of data without using Big Data platforms, the choice was made to use the Dask library (https ://dask.org/) which allows to perform this type of operations on tables which are too big to be loaded in the RAM. The principle of dichotomy was also applied by splitting the original table into 23 and combining the files 2 by 2 at each iteration. This step was necessary because the limits of Dask on the considered machine were reached.

3.5. Manual work

Once this database was created, it became possible to effectively study the behavior of the Exchanges. Depending on their security management, some Exchanges use one or more cold wallets to store long-term funds on addresses that are not very active, while current deposit and withdrawal operations are performed on hot wallets. Some small Exchanges use the same address for incoming and outgoing flows, others separate deposit and withdrawal addresses and balance them periodically. The question has arisen as to whether automatic rules for identifying Exchange addresses can be established. The difficulty is to have high confidence in the results. A low signal-to-noise ratio is already expected in the study of flows, there is no need to decrease it further by introducing false positives. The choice was therefore made to use a number of criteria to automatically extract potential candidates but to manually check the validity of the proposals. The criteria used are judgmental, but the broad outlines are:

- Hot wallets have many transactions with low average amounts and high variance,
- Cold wallets, on the other hand, have few high-volume transactions and are linked to hot wallets on the input and output sides,
- Hot wallets receive or send to many different addresses (platform users).

All of these criteria make it possible to determine a good approximation of the universe of addresses controlled by the exchanges. The greatest difficulty encountered was for the Coinbase platform which does not use fixed addresses.

3.6. Special case of Coinbase

Similar to a common practice on the Bitcoin blockchain, Coinbase does not use a fixed address and continually evolves its funds in a network of continuously generated addresses. The investigation began with a manual exploration, from public forums, where some users revealed the address of their Coinbase repository. It turns out that Coinbase uses addresses with a cumulative entry amount of 1001 Ether or 501, 101 etc. All these addresses form a strongly connected graph.

The heuristic used to identify the set of these addresses is the following: the addresses having one of these scales at its peak (sum of inputs, then sum of outputs) and pointing to each other.

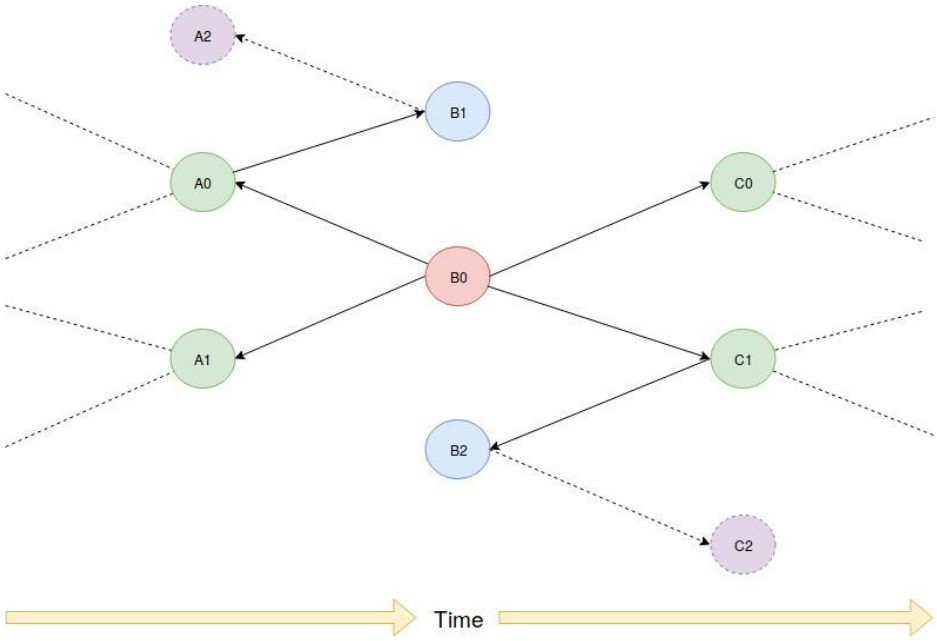

Figure 2 - Forward and Backward Propagation on the address graph

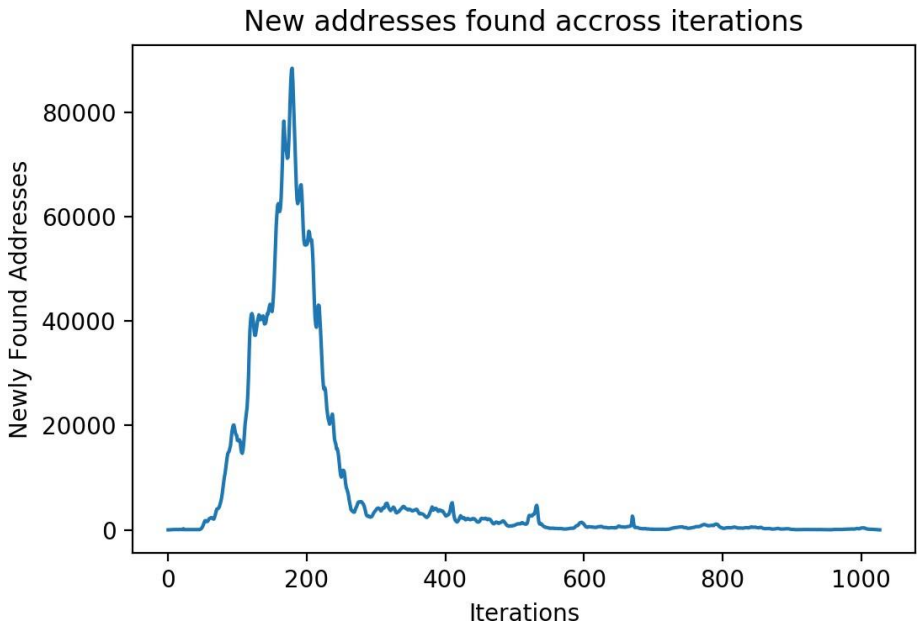

Figure 3 - New *addresses* found during the 1st round of Forward-Backward Propagation

The repeated use of Forward propagation and Back propagation from each newly found address, unless it has already been visited, allows to efficiently traverse the spider web formed by this graph.

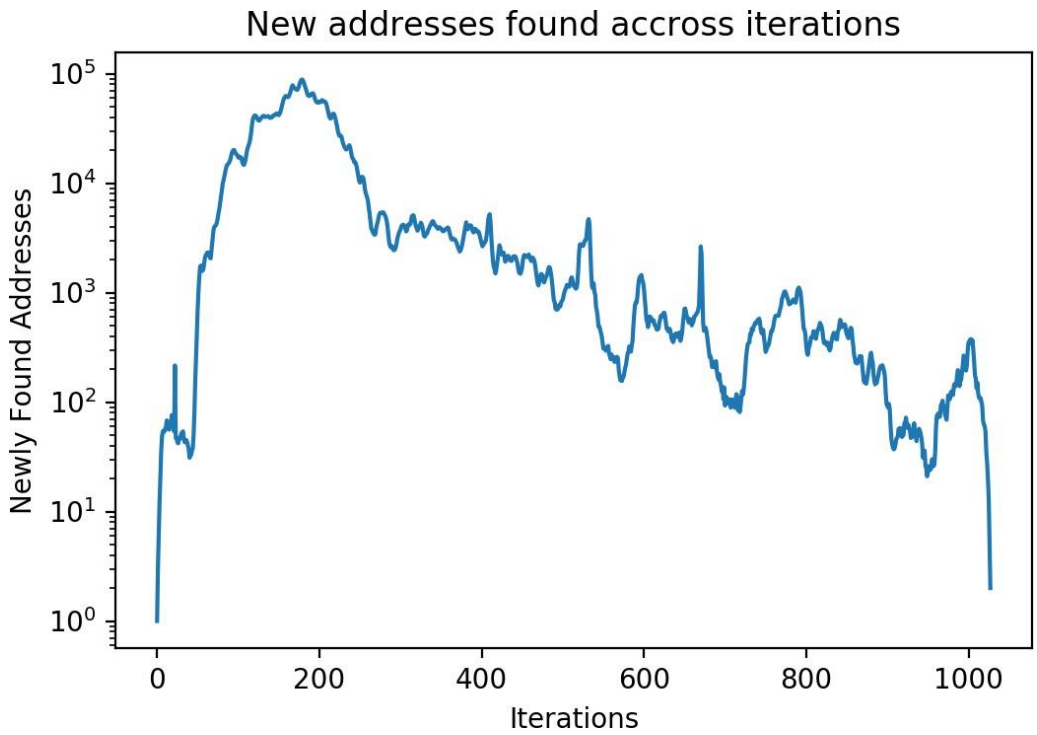

Figure 4 - New addresses found during the 1st round of Forward-Backward Propagation

Figure 2 represents an extract of this graph. During the first cycle, and using B0 as a starting point, the exploration is done in hourglass and discovers A0, A1, C0 and C1 in green. The exploration continues in the same direction. In the next cycle, all the addresses found are used as starting points and B1 and B2 can be discovered in blue. In the next cycle A2 and C2 can be discovered. Figure 3 and Figure 4 show with a linear and logarithmic scale respectively the number of new addresses found in each iteration of the first Forward-Backward Propagation cycle. We notice a slow start, a performance peak around iteration 150 and a slower end of cycle until about 1000 iterations or the cycle stops spontaneously. After 9 increasingly fast cycles with few new addresses, the entire graph defined by our heuristic has been traversed. The underlying assumption used is that all their Coinbase addresses are interconnected at some point, or formulated differently, that there is no subgraph not connected to the one we identified (from a single starting address)The set of addresses we identified shows in Figure 5 that Coinbase is a major player in the Ethereum chain with about 107 unique addresses used out of a total of about 7,107, or 1/7 of the total.

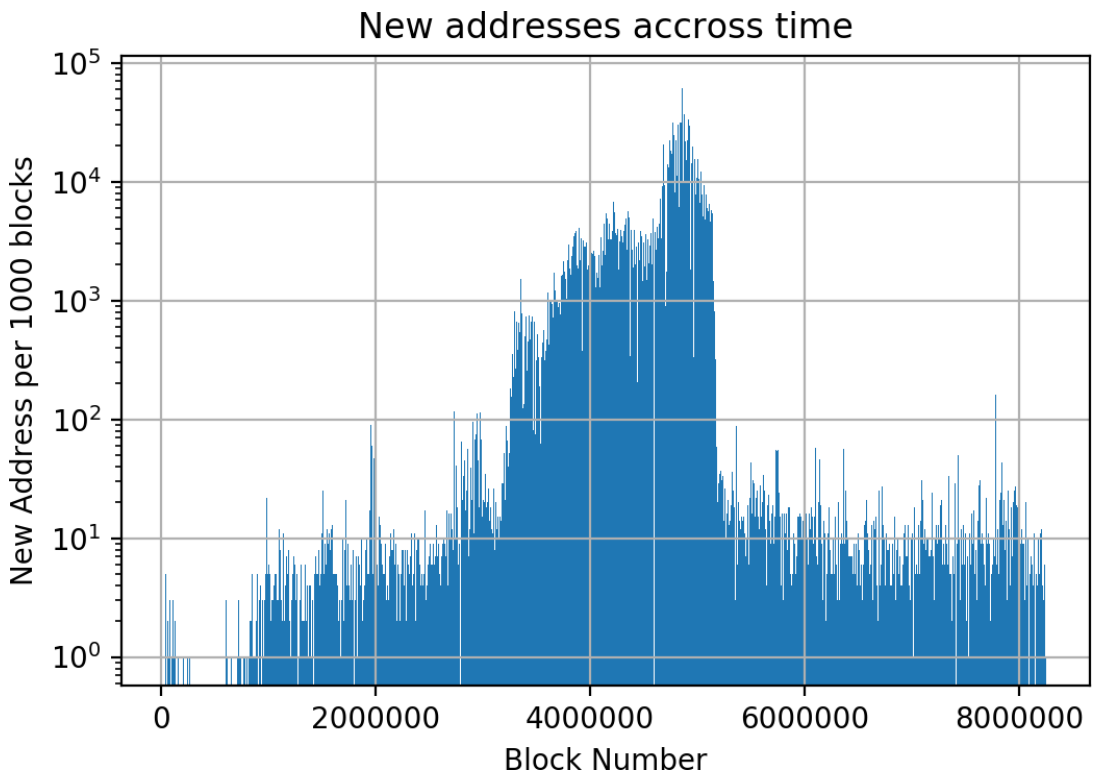

Figure 5 - Distribution of the creation dates of the new Coinbase addresses

3.7. Extraction of flows

The same work of identifying exchanges having been done for ERC-20 tokens, it remains to establish all the incoming and outgoing flows of the universe of addresses of the Exchanges. The assumption used is that if Ether or another currency is transferred to the Exchanges, it is to be sold; similarly, an outgoing flow corresponds to a recent purchase. All flows can be assimilated if we consider that the market is sufficiently arbitrated so that a significant price difference between 2 exchanges is passed on to the others or corrected.

The extraction of the price history was done from CoinMarketCap (https://coinmarketcap.com/). This platform is sometimes criticized for the lack of data filtering and the importance it can give to exchanges where the volume is not real. We consider that it is nevertheless a good approximation of prices. Indeed, even if it is a biased estimator, the constant component of the bias disappears when we take the first derivative with respect to time.

3.8. Establishment of a price prediction model

Our model aims to generate buy or sell signals. Thus, unlike the models in the literature, we are interested in a classification problem. The three categories are :

- BUY
- SELL
- NEUTRAL

3.9. Features

The particularity of our study is to use the incoming and outgoing flows of trade. To this we add some price variation data. For a point at time t0, the construction of the raw features amounts to aggregating over windows of variable size (from 14 days to 30 min before t0) the incoming flows or inputs, outgoing flows or outputs and the net flows. In the same way, we calculate the price variations over these same time windows. We thus obtain 40 initial features:

window	inputs	outputs	Net flow	Price return
14D	inputs14D	outputs14D	netflow14D	return14D
7D	Inputs7D	outputs7D	netflow7D	return7D
4D	Inputs4D	outputs4D	netflow4D	return4D
2D	Inputs2D	outputs2D	netflow2D	return2D
1D	Inputs1D	outputs1D	netflow1D	return1D
12H	inputs12H	outputs12H	netflow12H	return12H
6H	Inputs6H	outputs6H	netflow6H	return6H
3H	Inputs3H	outputs3H	netflow3H	return3H
1H	Inputs1H	outputs1H	netflow1H	return1H
30Min	Inputs30Min	outputs30Min	netflow30Min	return30Min

3.10. Features Selection

The set of second-order variables is then created to account for possible interactions between the initial raw variables and a Principal Component Analysis (PCA) is applied to the set to reduce to only 10 features with the most explanatory variance.

3.11. Training, validation and testing periods

We are in the case of time series where training the model at all instants of the training period is like trying to interpolate noise. We have to keep in mind that our raw data is only a noisy proxy of supply and demand. We chose as training target the variation of the cryptocurrency price during the 12H following t0. We then assign the label BUY if the target is more than 2% above the price at t0, SELL if the target is more than 2% below the price at t0, NEUTRAL otherwise. We thus seek to train a highly biased model where the majority response is neutral and where we privilege the accuracy of non-trivial predictions.

3.12. Choice of model

We have used 2 different classifiers whose constructors in Python, using the scikitlearn library, are as follows:

m1 = LogisticRegressionCV ([0 . 0 1 , 0 . 1 , 1 , 1 0] , cv =10, multiclass='multinomial ')

m2 = RandomForestClassifier (max depth=6, n_estimators =10, randomstate =0)

The result of the two models is then used in an ensemble. There is no majority vote for the final decision because only two models make up the set. The decision must be unanimous or neutral. This reduces the number of trades (non-trivial signal) but significantly increases the accuracy.

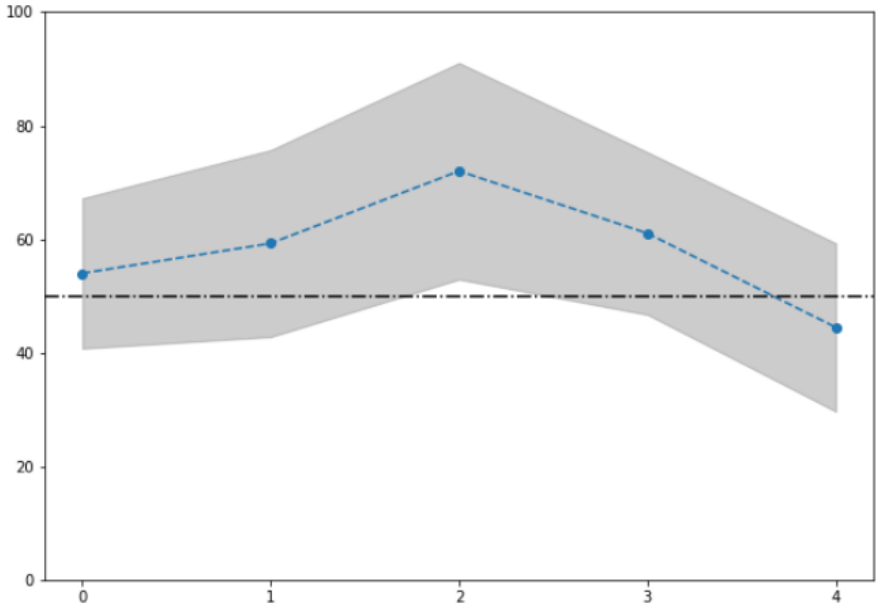

Figure 6 - Average accuracy over the 5 sub-test periods

Figure 6 shows the average accuracy over the 5 sub-test periods. The average precision over the whole test period is 57.1%. The shaded area represents the standard deviation of this accuracy, depending on the token: an average accuracy is calculated for each token.

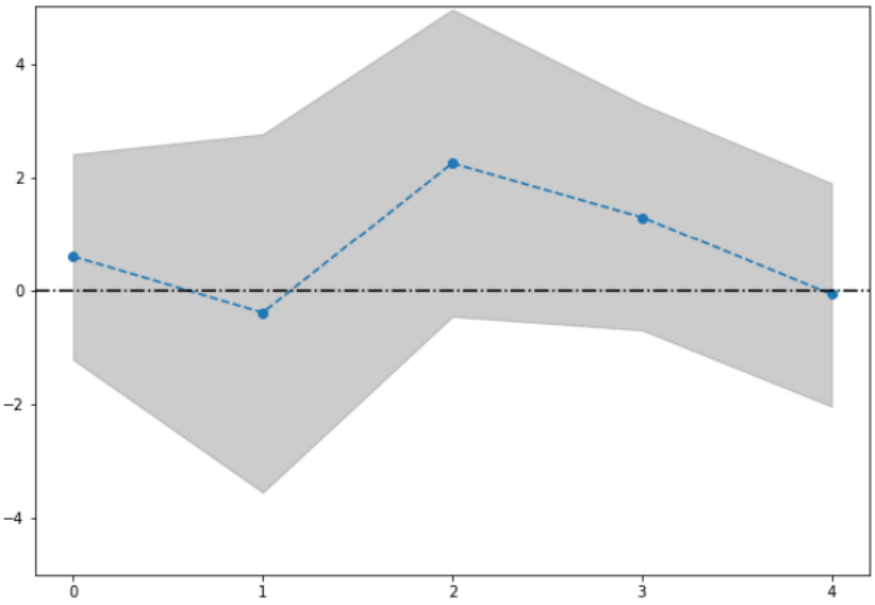

Figure 6 - Average performance over the 5 sub-test periods

Figure 7 shows the average performance over these same sub-periods. The average over the entire test period is 0.67%. It would be interesting to calculate a Sharpe ratio in order to evaluate the performance of the strategy. However, the values presented here represent a system that is not implementable in practice: the value of the Sharpe ratio would therefore be overestimated. Among the main restrictions are the impossibility to sell most of the tokens short. Secondly, the liquidity of most tokens is very limited and the market impact of the strategy would quickly become significant. Only Ether is sufficiently liquid and can be sold short. The accuracy of the model on Ether alone is however lower (53.7% accuracy). This is due to the fact that market participants tend to hold more Ether on the exchanges; this creates buffer zones that decrease the predictive power of the proxy we have chosen.

4. CONCLUSION

4.1. Results and knowledge gained

Our work has allowed us to gain a better understanding of how the Ethereum ecosystem works as well as the different policies used by exchanges to manage their users' funds. While much of the traffic to and from exchanges is readily available, Coinbase flows are an important buried part of the Ether movement iceberg. The study of the predictive power of these flows has proven to be conclusive although difficult to transpose into a trading system. We can also note the interesting but predictable result that the predictive power is higher on the downside than on the upside. Indeed, if we decompose the incoming flow, each deposit necessarily predicts a sale, while for the outgoing flow, each purchase necessarily predicts a withdrawal.

4.2. Prospects for the future

The work we present here leaves different perspectives for improvements:

- After this preliminary study focusing not only on Ether but on the whole ecosystem living on the Ethereum Blockchain, it would be interesting to refocus the study on Ether. Its greater liquidity and the ability to sell short makes it a better candidate for a trading system.

- The data, other than flows, used as input to the prediction model had been deliberately restricted to the strict minimum in order not to introduce noise. One avenue of research would be to broaden the scope of the input data by including more trade data such as volume, funding rate, etc., or channel-specific data such as Ether trading volume or average age of the corners moved.

5. BIBLIOGRAPHIE

- Predicting digital asset market based on blockchain activity data. Zvezdin Besarabov, Todor Kolev. Computational Finance, Oct. 2018.
- Bitcoin : A Peer-to-Peer Electronic Cash System. Satoshi Nakamoto. www.bitcoin.org, 2008.
- Ethereum : A Secure Decentralized Generalised Transaction Ledger. Gavin Wood. EIP-150 Revision, 2014.
- An Analysis of Anonymity in the Bitcoin System. Fergal Reid, Martin Harrigan. arXiv, 2011.
- The Unreasonable Effectiveness of Address Clustering. Martin Harrigan, Christoph Fretter. arXiv, 2018.
- Bitcoin Transaction Graph Analysis. Michael Fleder, Michael S. Kester, Sudeep Pillai. arXiv, 2015.
- Quantitative Description of Internal Activityon the Ethereum Public Blockchain. Andra Anoaica, Hugo Levard. IRT SystemX, 2018.
- Classifying Ethereum users using blockchain data. Matthias De Aliaga. https://medium.com/tokenanalyst/classifying-ethereum-users-using-blockchain-data-dd6edb867de3, 2018.
- Clustering Ethereum Addresses. Will Price. https://towardsdatascience.com/clustering-ethereum-addresses-18aeca61919d, 2018.
- Automated Bitcoin Trading via Machine Learning Algorithm. Isaac Madan, Shaurya Saluja, Aojia Zhao. Stanford, 2014.

Melanion Capital

17 Avenue Georges V
75008 Paris

For info:
Contact@melanion.com