On-Chain Investor Behavior Analysis: Preliminary Design and Results

Lihan Yao, MS
Bird
lihan@bird.money

Ahmed Abdullah
Bird
ahmed@bird.money

Daniel Stephens
Bird
daniel@bird.money

February 18, 2022

Introduction

This paper introduces Bird's custom metric metric for identifying valuable investors based on on-chain data. During application development to compute the Bird Investor Score, the Bird analytics team has gained new insights into AI applications as they are applied to blockchain data: developing the data infrastructure to conduct analytics, creating data science metrics, trying out models, and thinking deeply about feature engineering.

This is a multifaceted report of experiments, decisions, and findings throughout our preliminary development effort. We begin with our conclusion - working with blockchain data is difficult and complex for purposes of AI usage and data science. Key challenges that complicate the featurization and

inference process are highlighted. To our knowledge, this is the first instance articulating *why* blockchains are structurally complex for machine learning techniques. In comparison, core challenges of computer vision and natural language processing are mature with incremental progress.

As this budding field that is at the intersection of two major technological advancements continues to evolve, we ask the reader to consider the results and findings here to be fluid and illustrative. Experiment figures and experiment results are to be further validated and should not inform business decisions.

Featurizing Blockchain Data

Examples of inference at the address level include mapping an address to scalar, e.g. scoring a wallet address for its propensity to hold project tokens for extended periods. Another example is mapping address to a set of addresses e.g. predicting an address which most likely will transact with the queried address.

Considering the 'Big Data' magnitude of addresses in a mature protocol, and the readily available toolbox of modern machine learning, the core challenge in blockchain inference is filling the missing link between these assets: the computation of high quality address representations. If addresses are adequately numericalized, all addresses may be mined, mapped to numerical representations, and ingested by a deep learning model.

In other words, the team that successfully represents blockchain entities in a signal-rich scheme immediately accesses the Big Data regime of publicly available on-chain activities, while leveraging applicable machine learning insights cultivated in the past decade. The following narrative highlights key challenges and contains miniature surveys of possible solutions.

If addresses form our input space, there is no natural metric with which to structure elements of the input. By accumulating transactions over an address set X, one may compute the features

{ wallet value, earliest transaction in block height, number of transactions }

Which lends to a representation of $X \subset R^3$ with euclidean distance $||\ \cdot\ ||$.

Using this choice, the similarity of two wallets is their euclidean distance over these values. This, however, is arbitrary in the choice of features and metric. One may find excellent performance after engineering these features further, but such a representation becomes specialized to the task. The desired featurization process is computed with an underlying architecture that is general, which can service a variety of different models, and lessens the delivery time of future products.

Network Interactions

The aggregated statistics above do not reflect the wallet's participation in the protocol network - a source of important signals for most use cases within blockchain analytics. If we interpret addresses as nodes of a graph, with a directional edge representing a transaction from sender node to receiver node, a

representation accounting for address interactions becomes possible. Yet encoding the local connectivity information in a fixed-sized vector is not simple.

In modern machine learning, the fields of representation learning and geometric deep learning have led to algorithms that map graph nodes to their vector representations. Graph neural networks [1], for example, generate node-level embeddings that encode graph topology (i.e. connectivity) at multiple scales. An examination of a node vector resulting from this family of techniques reveals the node's local neighborhoods and neighbor attributes. In theory, following the above example, one may conclude the 'middleman' value of an address (defined as being part of a bottleneck for value flows) by examining an address representation.

Sparsity

Due to the blockchain's public-private key configuration and permissionless nature, wallet addresses and project addresses are wildly abundant. Anyone may instantiate a new contract address, but protocol participants find utility in a handful of project contracts. For most L1 protocols, a project's cumulative interactions form an exponential distribution. The distribution of total interactions for wallet addresses is similarly concentrated. Recognized exchange wallets have a high number of interactions.

In data science and specifically high dimensional statistics, models often rely on sparsity to mitigate complexity in high dimensional data. In the blockchain context, the sparsity of transactions is instead a challenge in encoding protocol connectivity structure. The probability of edge existence between any two addresses is astronomically close to zero. The probability that an active wallet will interact with a popular exchange such as binance over the next few million blocks on binance smart chain is around 1%. This makes clusters formed by bi-directional edges and other connectivity structures a rarity.

In natural language processing, the Word2Vec framework [2] maps words to a rich, continuous embedding according to their connectivity to other elements, where pairwise affinity increases if the words frequently co-appear in sentences. An analogous application of Word2Vec to represent addresses as dense vectors may be fruitful, where a transaction from one address to another implies a (directional) relation between these addresses. The geometric intuition is that, for frequently co-interactive addresses, their address embeddings are close. In applying Word2Vec, the technique is constrained by sparsity of address-level transactions (the edges). Interpreting this problem in the original NLP domain, our corpus is dominated by exotic words, with extremely few sentences from which connections between words can be estimated.

A key insight to the sparsity challenge is, in the address graph, there are nodes which are less sparse, or even dense, in the number of interactions with other nodes. Define the set of exchange wallets, project contracts, DeFi contracts, burn addresses, as *landmark addresses* whose interactions within the ecosystem are frequent. A comprehensive indexing of key addresses such that a common address has interacted with *some* landmark address, informs the representation of the common wallet. In other words, landmarks form the subgraph within the address graph that captures the bulk of edges.

Temporal Dynamics and System Dynamics

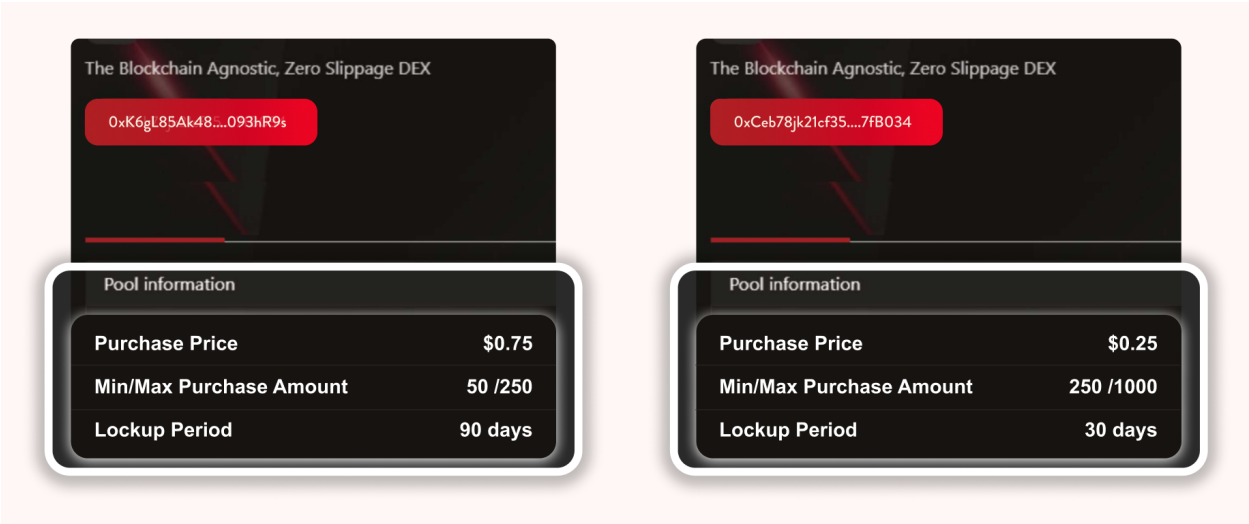
If interactions among blockchain entities inform their featurization, change over time in transaction frequency, volume, and structural relations is an additional facet to encode. High level metrics summarizing Layer 1 ecosystems include Total Value Locked, number of active addresses, and number of transactions. The temporal dynamics exhibited by these metrics can lead to rich conclusions regarding the health of the ecosystem, its rate of decentralization, and future price of the native currency (e.g. a commonly viewed trading signal are time serieses of Bitcoin value held in old wallets vs new wallets). At the entity level, temporal dynamics help query entity information conditioned on past events, and are indispensable for applications that require the most recent state or behavior of the entity. In contrast, if the entity is represented by its timeless features, average behavior across history is ingested by the model instead of recent behavior.

The literature of system dynamics provides one more perspective on blockchain activities. Addresses are viewed as interdependent participants of a complex system. Flows, feedback loops, and cyclic/acyclic structure may be identified. Bitcoin's stock-to-flow cycle, for example, begins with lower miner supply due to algorithmic scarcity, leading to lower overall supply, leading to higher price for the same demand, and returning back to an increase of miners motivated by higher prices. Another is the utility cycle driving the tokenomics of projects with valid token usage: as more use cases come online in forms of staking and consumption, token demand increases; for the same supply, the increased demand leads to higher token price, which leads to development funding for more use cases. By estimating the role of different addresses and their contributions to system level behavior, a richer representation is achieved. More importantly, inexplicable, anomalous transactions common in today's blockchains may be explained or atleast contextualized with the system dynamics lens.

Holding Time Prediction for Launchpads

Launchpads can offer unique investment terms to each wallet address, ensuring that early stage projects attract investors that are likely to support the company long-term. One factor crucial to the identification of high value wallets is its holding time. In Bird's upcoming launchpad application, the average holding time serves as a target variable. The application scores wallets for future holding times of the client project; the Blockchain Individualized Risk Default (BIRD) score proxies the wallet's value as a project customer. Highly scored wallets are rewarded by launchpad projects.

In this section we define average holding time, the layer 1 ecosystem being examined, and metric choices for the application.



Sample Launchpad Application Interface

The holding time prediction task defines a model M which maps a wallet w and project p to a real number forecasting the wallet's average holding time. The average holding time accounts for the notion that a wallet may enter or leave a position by varying amounts, and measures how long, in blocks or days, an average p token has been held by w. The computations follow the First-In-First-Out rule (FIFO), and a position maintained past the block number when data collection occurred is assumed to have held at least that long.

Suppose w begins day 0 with 6 tokens. After 1 day, w sold 3 tokens. After 6 days, w sold the remaining 3 tokens. The average holding time y is

$$y(w, p) = (3 \cdot 1 + 3 \cdot 6) / 6 = 3.5 \text{ days}$$

With event-based data collection (in contrast to continuous data collection), if w's transactions are observed on day 5, 1 day prior to the sale, the holding time lower bound is assumed.

$$y\_event(w, p) = (3 \cdot 1 + 3 \cdot 5) / 6 = 3 \text{ days}$$

The current implementation uses event-based data collection. In expectation, the application tends to be pessimistic in its holding time estimates.

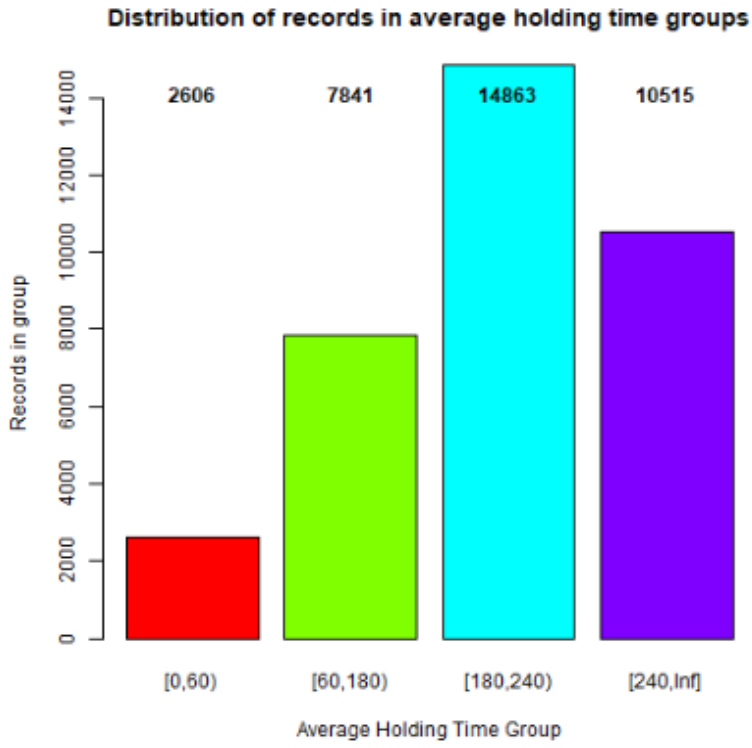
Binance Smart Chain and Data Infrastructure

The following figures and performance pertain to data from the Binance Smart Chain (BSC) between mainnet launch and November 2021. The Ethereum-ETL library was used to extract block transactions into a transaction level tabular database. Where 'project' is mentioned, one of 16 selected projects active

in the BSC ecosystem, such as CAKE, 1INCH, SFP, and BAKE are referenced. The off-chain query + aggregation process is done through Amazon Web Service. For this process, Bird is re-organizing upstream tasks into uniform Extract-Transform-Load (ETL) infrastructure. The data exploration, training pipeline, and inference pipeline are developed with reusability and modularity principles.

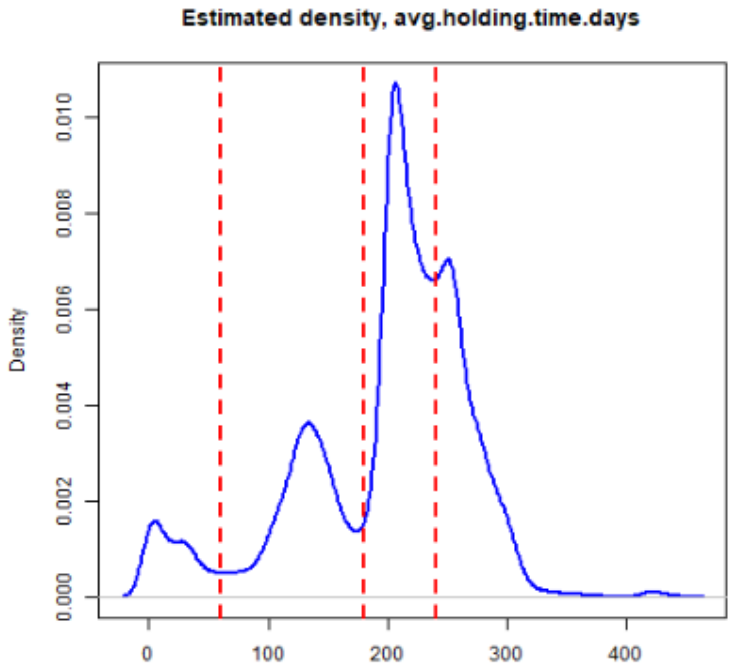
Metrics

Evaluating a model's holding time prediction performance is related to the business use case. In the launchpad investor scoring use case, the identification of high value investors is prioritized over guessing the actual number of days that investor would hold. This is a classification problem, where the cost tradeoff between false positives and false negatives is dependent on the launchpad and project resources. After binning the available wallets into 4 intervals, the histogram below provides the class breakdown. One entry in the histogram corresponds to a (wallet, project) entry.



The true cost associated with each error is dependent on budget and growth objectives particular to the client. It is possible a large budget enables broader targeting of high-value investors, and consequently more high-value investors are converted at the cost of a higher false positive rate. If confidence in the model is high, the project may reward investors using a steep reward curve.

We also evaluate model performance as solving the regression problem: "how many days is this wallet expected to hold?". After wallet holding times are plotted as a distribution, peaks are observed. The peaks likely match with adoption events when a large wave of users are on boarded around the same time. Red lines indicate bin boundaries from the class breakdown in the histogram.



Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) has been straightforward to reason with in this formulation. We prioritize evaluating regression performance using MAE. MAE linearizes the error penalty where errors on outlier predictions are penalized in proportion to the error magnitude. An advantage of MAE is it enables statements to the effect of "this prediction plus or minus X days, on average".

The (Wallet, Project) data primitive

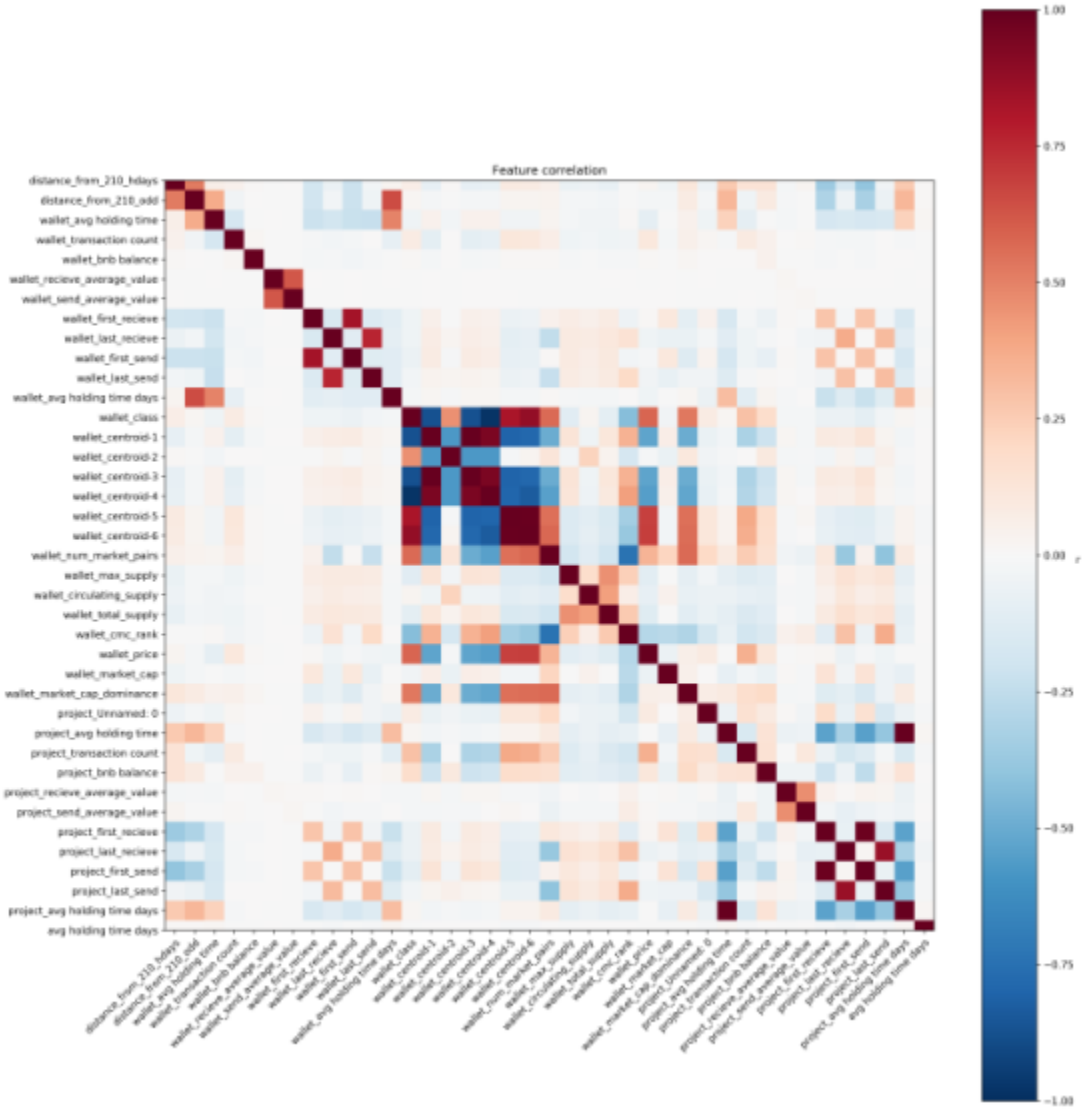
The tuple (wallet hash, project hash) contains features that quantify wallet behavior specific to a project. From transaction level data found on-chain, specifying (wallet, project) retrieves the transaction subset where the wallet is either the sender or receiver of a token transfer, and token address is that of the specified project. Features may be engineered from transaction history, e.g. trading frequency and average holding time (the target variable).

Among the challenges identified in "Featurizing Blockchain Data", this early solution addresses **network connectivity** and **sparsity** as simply as possible. A select set of project addresses containing 16 projects forms the set of landmark addresses (or landmark set, a concept introduced in the sparsity discussion). Wallet addresses are represented by featurizing its BEP20 transactions where a token transfer's token address has membership in the landmark set.

A model ingesting this dataset format enjoys high project fidelity, while inter-wallet actions, and change in activity over time, are missed. A previous, simpler approach is to accumulate token transfers for the wallet, without distinguishing which project they belong to. The wallet features are then ingested to predict its average holding time with respect to a particular project. Besides additional linkage

opportunities provided with the (wallet, project) format, accessing the interaction features recreates the wallet-level features by weighted averaging. These additional two benefits are examined below.

By accumulating on-chain transactions across time into this form, the versatility of (Wallet, Project) is high. First, additional project level features, such as public project information found on CoinMarketCap, may be integrated by joining with (Wallet, Project) on the project hash column. Thus, knowledge of a wallet's project preferences can improve downstream predictions. Consider a one-hot encoded project category feature. A wallet that holds gaming tokens for long periods but leaves non-gaming positions quickly will be scored higher for a gaming project client than others. Other factors such as max supply, number of holding wallets, etc. will be accounted for as part of the wallet preference.



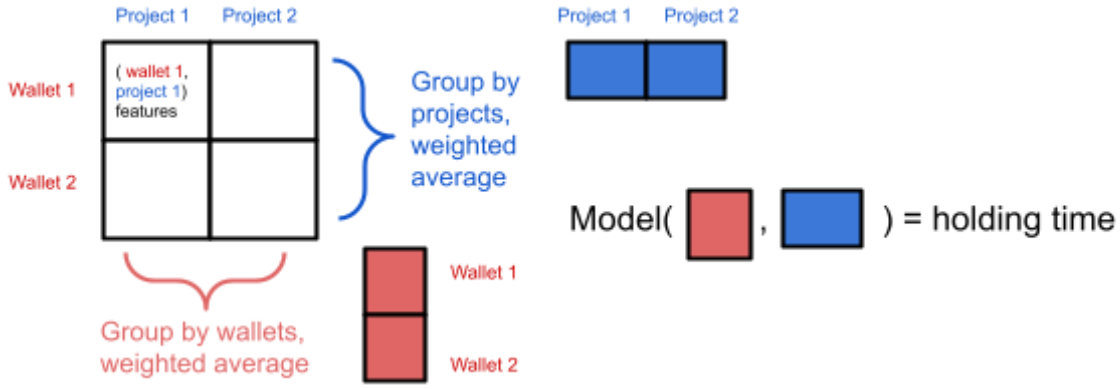
Uncovering associations between wallets are another signal source. In the Netflix Movie Recommendation Dataset, techniques involving matrix factorization and collaborative filtering impute a user's movie preference based on preferences of users with similar movie tastes. In the next iteration of

application development, Bird's analytics effort seeks to uncover wallet groupings, insights about distinct behavior archetypes, and imputing missing values.

To recreate average wallet1 statistics from a set of interaction statistics {(wallet1, project1), …, (wallet1, project_n)}, weigh interactions by cumulative project value and compute the weighted average. In practice, wallet statistics are computed directly, prior to computing its interaction statistics.

Production Considerations, Concept Drift and Leakage

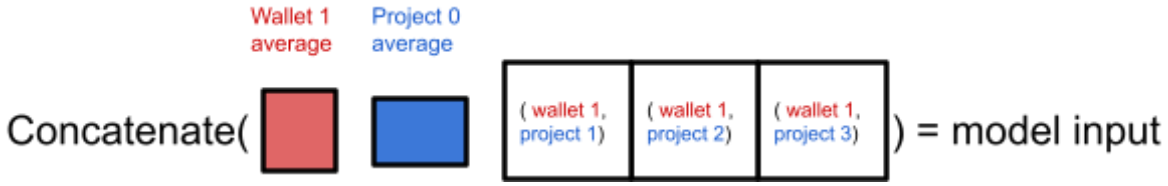
For the launchpad use case, features particular to the (wallet, project) interaction will not be accessible because tokens of a project being launched, by definition, will not be found in wallets. Following this, projects to be predicted in the test set do not appear in the validation set and training set. For example, the SafeMoon project, and wallets holding it (wallet, SafeMoon), are only in the test set.



In a production setting, the application can access wallet features, and any information available about the project. For the (wallet, project) dataset, interaction tuples whose holding time is being predicted is first held out before featurization continues.

The final feature set is created from grouping by wallets, then weighted averaging of interaction statistics. A wallet average in this case is aggregated over projects held by this wallet (minus the held out tuple). The final feature set is concatenated with group by project then average; a project average represents the average behavior of all wallets holding this project (minus the held out tuple).

Lastly, to mitigate the loss of information from the two averaged feature sets, three (wallet, project) tuples are concatenated to this representation. If the wallet has more interactions than three, the three most transacted are kept. If the wallet has less than three interactions, the concatenated representation is padded.



In conclusion, a wallet observation has 285 features. The training set contains 13689 observations, the validation set contains 2004 observations, and 1087 observations in the test set.

Models and Preliminary Results

MLP Neural Network

By composing multiple feed forward linear layers with nonlinearities such as ReLU, the MLP Neural Network serves as a good starting model for holding time prediction. The neural network learns complex structure in data from simpler structure, now supported by insightful geometric intuition [Montufar 2014] and competitive engineering practices [Tolstikhin 2021 , Bird 2020] .

As a regressor, the MLP Neural Network outputs scalar predictions for predicted holding time in days. Parameter tuning suggests three feed forward layers with ReLU, with 64 neurons each, a learning rate of 0.001, and AdaBoost works well. This model achieves a MAE of 23.4 days on the test set.

By normalizing the predicted holding time into scores, ranking the wallet scores provide an ordering of investor value. For select projects in the test set like SafePal, the model can identify top hodlers (defined as >300 holding days) with 93% accuracy.

Each dot represents a wallet address. X-axis is the true holding time in days, y-axis the model's predicted holding score. Top Hodlers are defined as those with >300 days.

To reproduce production conditions, a subset of projects in the test set may not appear in validation or test sets. For example, the SafePal token was selected as a test project. If wallets holding the SFP token were observed in the training set, SafePal would have already been publicly traded. Wallets holding the SafePal token SFP were simpler to predict, for reasons under investigation. The current analytics effort looks to generalize this performance to any project.

Additional model improvements include the implementation of a recurrent model such as Long-Short Term Memory (LSTM) model, or Gated Recurrent Unit (GRU) model. If an observation is unrolled

across time, say every 1 million BSC blocks, recurrent models can account for change over time of the 285 statistics. The key benefit being placing more weight on recent wallet behavior.

Gradient Boosting (Light GBM and XGBoost)

Gradient boosting approaches benefit from good performance and interpretability. In modern computational packages such as Light GBM and XGBoost, leaf configuration of decision trees can provide insight into feature importance. A wide range of functionalities including implicit feature selection, sparse optimization, parallel training, multiple loss functions, regularization, bagging, and early stopping make for competitive models underlying applications.

Conclusion

Insights from developing the Bird Investor Score for identifying valuable investors have been presented. The infrastructural, data science, and modeling decisions were motivated, and solutions under development in these areas were presented. Challenges that complicate the featurization of blockchain data were outlined.

Practical next steps include a complete implementation and tuning of gradient boosting models. Another objective is incorporation of temporal data to leverage recurrent models. More fundamental work includes a general featurization framework that addresses the highlighted challenges.

References

1. Scarselli, Franco, et al. "The graph neural network model." *IEEE transactions on neural networks* 20.1 (2008): 61-80.

2. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

3. Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." *Advances in neural information processing systems* 27 (2014).

4. Tolstikhin, Ilya O., et al. "Mlp-mixer: An all-mlp architecture for vision." *Advances in Neural Information Processing Systems* 34 (2021).

5. Bird, Jordan J., et al. "Cross-domain MLP and CNN transfer learning for biological signal processing: EEG and EMG." *IEEE Access* 8 (2020): 54789-54801.