Massachusetts Institute of Technology
Media Lab's Digital Currency Initiative
Sloan School of Management

Cryptoasset Quality Based on Source Code and Developer Dynamics

Authors: Yashashree Kokje (MIT), Océane Boulais (MIT)

Keywords: Cryptoassets

Blockchain Lab Program
2018-2019 Working Groups Cycle
Taught by N. Narula, G. Gensler, S. Johnson, and M. Casey
Member Company: Monetary Authority of Singapore (MAS)
Project Group: Token Quality Based on Source Code

Cryptoasset Quality Based on Source Code and Developer Dynamics

Co-Authors: Yashashree Kokje, Océane Boulais

DCI Advisors: Neha Narula, Rhys Lindmark

Industry Advisor: Lizchi Chen and Danielle Jiang, Monetary Authority of Singapore

Table of Contents

Executive Summary

Abstract

With interest in cryptocurrencies booming since the introduction of Bitcoin in 2008 [1], billions of dollars have been invested in the vast landscape of tokens and cryptocurrencies. This enthusiasm also comes with a number of scams that prey on lack of technical understanding in order to exploit the waves of hype in this space. We build on prior research by looking into open-source repositories to give insight into varying dimensions of quality. By looking at a cryptocurrencies' foundation from the perspective of code base, we investigate developer dynamic.

For our definition of cryptocurrency health, we have chosen to exclude monetary value and other associated characteristics such as initial distribution, current supply, max supply. Instead, we gather Github metadata, which provides information associated with github repositories of open-source cryptocurrencies and tokens [2]. Building on previous work around implementation of cryptographically-secure code [3] , we infer a set of patterns that categorize different dimensions of a cryptocurrency project. Many cryptocurrencies list tokens that are built on existing cryptocurrencies, the most common of those being ERC20 tokens built on Ethereum. We assess only those tokens with which have files that are publically available. Upon inferring these patterns for what constitutes a 'quality' token, we use the defined features as signals to train our machine learning model.

Background

There are a couple of projects that provide commercial tools to process cryptocurrency quality. Flipside Crypto is one such a company that assesses cryptocurrency project through the Fundamental Crypto Asset Score [4]. FCAS is a

comparative metric used to assess the fundamental health of crypto projects. The score is derived from the interactivity between primary project lifecycle fundamentals: User Activity, Developer Behavior, and Market Maturity. Another project that we looked to for inspiration as we began this research thrust was CoinDesk's Crypto-Economic Explorer. Their measurements are segmented into five categories: Price, Exchange, Network, Social, and Developer [5].

Insights

Upon sifting through 12,141 cryptocurrency repositories we combined summary statistics with manual inspection to find patterns of usual and unusual activity. We identified the following areas of importance: Popularity, Engagement, and Authenticity. In order to assess Popularity, we observe the Fork count and the amount of Subscribers/Stargazers per repository. To investigate Engagement, we look at commit frequency (both additions and deletions) over a 2 year span. To assess authenticity, we look at anonymous contributions and whether issue creation is disabled or not. Issue creation is enabled by default when a repository is created on Github. Upon analyzing the data we collected and spot-checking, we came to the following key insights.

Summary Table of Key Insights

Quality Dimension	Pattern	Insight
Popularity	Number of forks	1) Ethereum (go-ethereum) 2) Bitcoin (bitcoin) 3) EOS (eos)
	Number of subscribers and stargazers	1) Ethereum (go-ethereum) 2) Cardano (cardano-sl) 3) Status (status-react)
Engagement*	Last push times, commit frequency	High activity: monthly alterations Medium activity: alterations every ~6mo Low Activity: alterations nonexistent after repository creation or once a yr
	Number of code alterations (per repository, monthly)	High activity: 5-6 Medium activity: 2-4 Low Activity: 0-1
Authenticity	Disabled Issue creation	45 cryptocurrencies had disabled issues
	Anonymous user contributions	27 cryptocurrencies were made up only of anonymous contributions

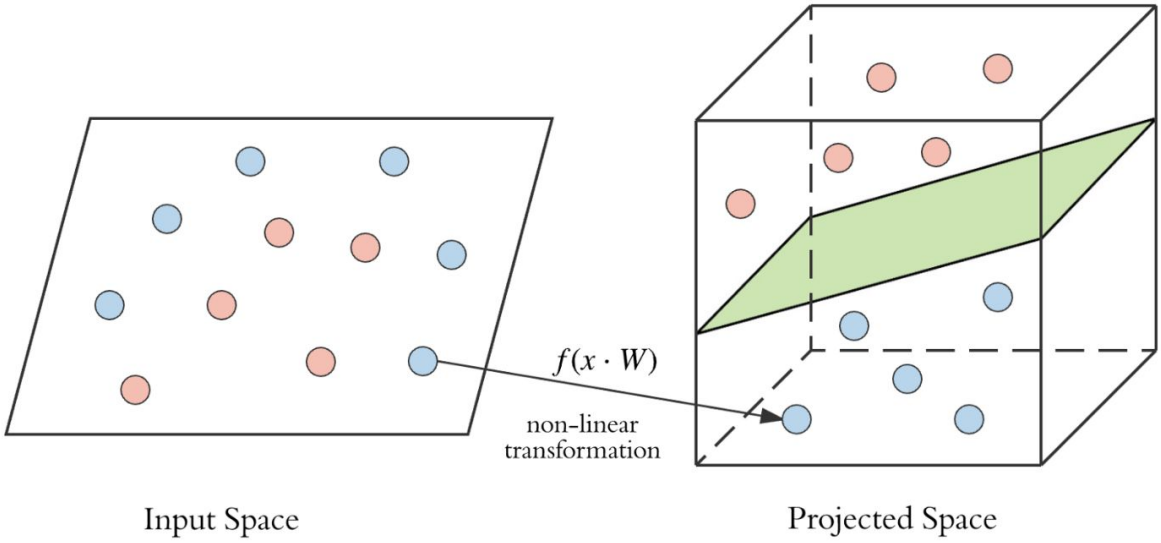
Table 1: Quality Dimension and Insight Analysis

**Engagement is analyzed on a daily basis over a 2 year period for a selected group of cryptocurrency projects that we found as "outliers" in our data analysis. Those projects are, respectively, 42-coin, Adcoin, Bitcoin, Capricoin, Eryllium, Ethereum, SelfSell.*

Next steps

Our current insights are based on deep diving into a selected subset of repositories. We have collected anecdotes that, at a high level, validate the signals we're looking at. Additional sources of information used for validation are market capital from CoinmarketCap and reports specifying the rules for participation in regulated exchanges. In order to scale this assessment to all the 12k repositories in our dataset we trained a supervised learning algorithm using Support Vector Machines (SVM). Our algorithm classifies data points with an accuracy of 70% and although this is less than ideal, it performs better than random. We believe this is a classification problem because the outputs we are trying to predict are discrete values. Alternatively, if we were trying to predict the price movement of a coin, our inputs and outputs would be timeseries making this a regression problem. Furthermore, the list of ranked repositories we use to train our algorithm may not be accurate. We did a random sampling to spot check the labels and verified that the general distribution is valid. For example: we agree that Bitcoin, Ethereum, EOS are good quality and hence should be listed at the top but hesitate to accept an absolute ordering ie does Bitcoin necessarily outrank Ethereum? This rationale applies to all the repositories down the list justifying our classification approach.

Given github metadata about a repository our algorithm outputs a probability of that repository falling into the category of good quality tokens. An SVM model works best when the data is linearly separable. However if the data is not immediately separable, we can transform it into a higher dimensional space where it may be perfectly or nearly separable.



In the diagram above, data points that were not linearly separable in a 2-D space can be projected onto a 3-D space with a plane separating the classes. The choice of kernel (Linear, Polynomial, Gaussian etc) determine the shape of the decision boundary. A detailed description of our approach is listed in the Results section.

Some of our ideas for improving these results are:

- Use a more complex function to capture the relationships between our data. For example: Neural Networks with more than 2 hidden layers or higher degree polynomial functions as SVM kernels. Due to time constraints during the semester we were unable to deploy these models as they take longer to finish training and require more computational resources.

- Collect additional data that can be used as signals to assess code quality. We plan to include information about code reviews such as primary and secondary reviewers, keyword analysis and test coverage. We would also like to filter out insignificant commits to files like READMEs or License updates etc

Conclusion

We finished the initial portion of this project with the understanding that we will continue into the summer and fall semester with a paper submission that discusses the insights we gather post-processing with our SVM or alternate model. At this time, we have laid down the foundation for understanding the complexities that engulf the term “quality” when it comes to evaluating open source projects. With the Github metadata API, we were able to scrape key data points from 12,141 repositories that made up about 1,011 coins. These data points enabled us to categorize the patterns we saw in the data in three bins - popularity, engagement and authenticity. These categorizations became the ‘dimensions of quality’ that we now use as features in the support vector machine as we attempt to get better accuracy scores.

Project problem and scope

Problem statement

Quantitative analysis of Github metrics collected from source code can provide rich insights into the health of the token’s foundation and technical community dynamics. Source code may determine how a particular software will behave. In this research thrust we aim to explore the possibility of classifying token quality based on source code. After analyzing 12,141 repositories, we identified source code features to determine token quality in three dimensions: Popularity, Engagement, and Authenticity.

Hypothesis

Our hypothesis is that quantitative analysis of source code and developer dynamics can provide rich insights into the health of the token’s foundation. For our

definition of token quality we have chosen to exclude monetary value and other associated characteristics such as initial distribution, current supply, max supply but we do take into consideration whether a token is in active circulation by spotchecking with exchanges. When we initially set out to analyze developer dynamic, we generated a list of the following features we thought would be insightful as we :

- Github Metadata: Information associated with github repositories
- Count of users who have forked, starred, watched
- Rate of pull requests, commits, issues opened, closed, triaged
- Number of reviewers required for merging
- How long the reviewers have been working on the repo
- Github ranking of the committers
- Code coverage based on unit/functional tests
- Number of comments per review
- Time to merge
- Activity distribution among committers
- Committers actively contributing to other 'well-respected' repos
- Travis statistics. Does it exist; If so, how often green/red
- Time to respond to critical bug reports / CVEs
- Time till resolution for bugs
- IRC/discord/slack chat activity for keyword analysis
- Github social graph

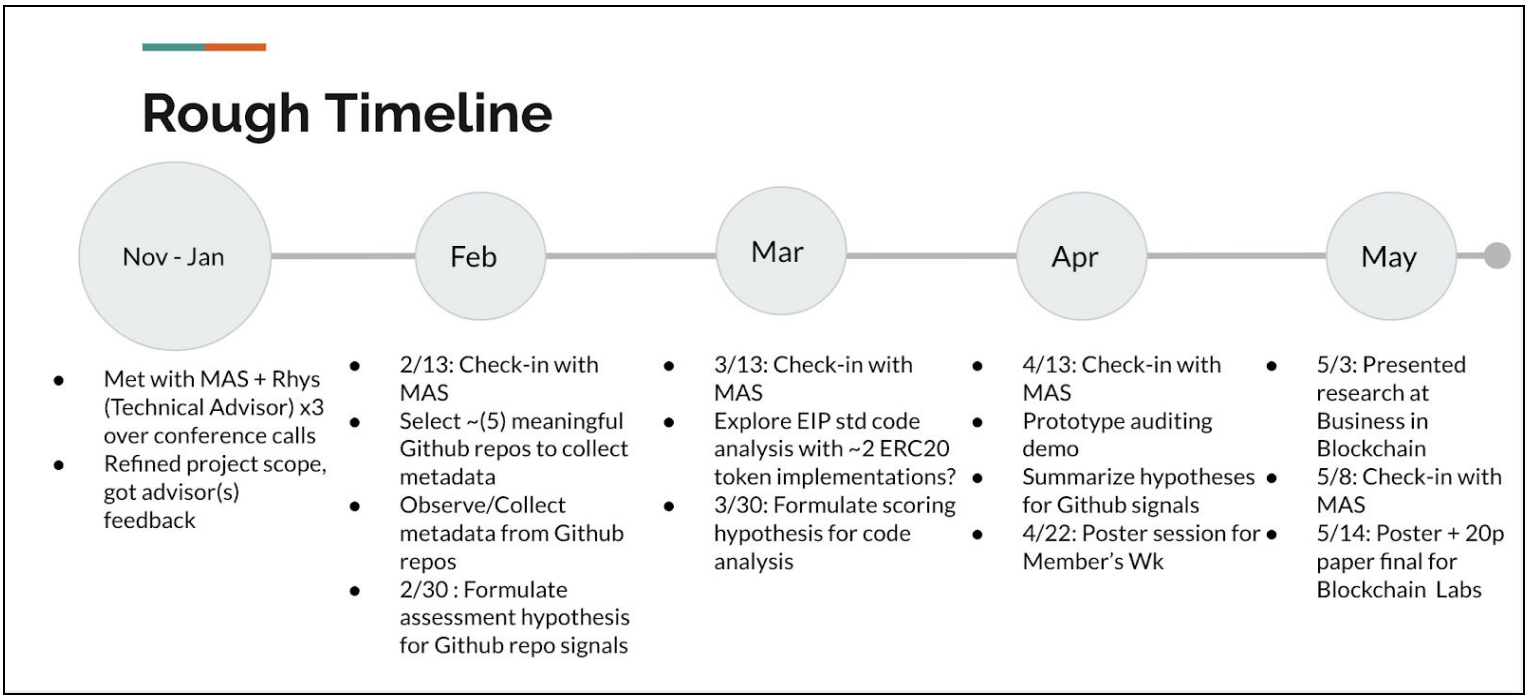
The above list was our "wish list" of features we thought would be telling features for insight into developer dynamic in the beginning of our project timeline. We whittled the above list down to the following list of features that we've used to build our initial "cryptocurrency quality" framework:

- Most/Least forked repositories

    - The more the forks on a specific repository, the more popular the repositories
- Count of repositories within a cryptocurrency project that are forks of other projects
- Count of repositories with issues disabled
    - If Issues are disabled, the repository/coin is informally considered "closed-source" to the community, and the opposite is true
- The more the open issues being actively addressed, the more active the developer community
- Actively Pushed Repositories
    - An active repository will have a push within the last 1-3mo
- The more Subscribers/Stargazers on a specific repository, the more popular the repository is

Scope/Timeline

The following figure was given to the Monetary Authority of Singapore and the DCI at the beginning of the term. This scope and timeline forced us to whittle down our ambitious ideas into sizeable chunks.



Assessment Methodology

From previous research we gathered a ranked list [5] of 13,695 repositories and analyzed tokens (i.e. Bitcoin) which has multiple repositories (i.e. bips, bitcoin, libbase58 etc). After filtering out erroneous repositories (HTTP 404 responses due to repositories being renamed or deleted), we scraped Github to retrieve metadata for 12,141 repos that corresponded to 1011 tokens. The repository metadata information included *url*, *forks_count*, *subscribers_count*, *network_count*, *open_issues_count*, *watchers_count*, *stargazers_count*, *size*, *created_at*, *updated_at*, *pushed_at*, *has_wiki*, *has_downloads*, *has_issues*, *is_fork*. We also collected information about user contributions to each repository and aggregated data as (total unique) *users*, *user_changes* (sum of additions and deletions to code), *anons* (contributors without accounts) and *anon_changes*. Finally, we also scraped the *weekly commit frequency* (sum of additions and deletions per week) over the past 2 years for each of the repositories.

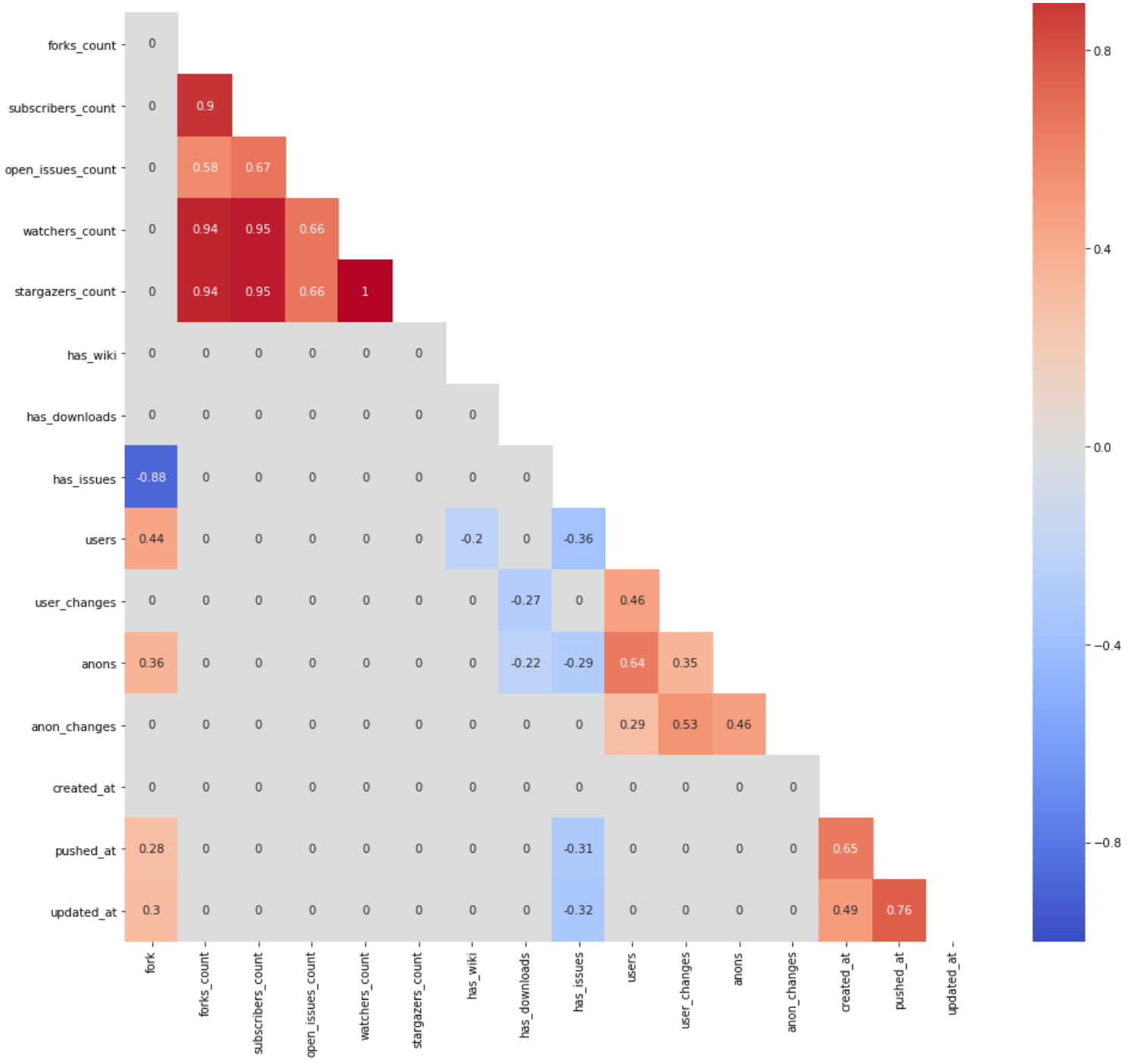
Correlation Matrix

Next we used a correlation matrix to group together related information and see if any high level patterns emerge. We use Pearson Correlation Coefficient[7] to measure the strength of a linear relationship between every pair of variables in our dataset. The output is a score between -1 and 1. A positive value indicates that when one variable increases, the second one will also increase while as a negative value indicates that when one variable increases the other one decreases.

- For example, *stargazers_count* and *watchers_count* are synonymous and have a correlation coefficient of 1. Github released the stargazers feature as a replacement for watchers but decided to continue supporting this endpoint for backwards compatibility[8]. The data however in both these metrics is identical.

- It was interesting to observe the strong negative correlation between *has_issues* and *fork*. This indicates that when repositories explicitly disable issue creation, these repositories tend to be forks of other repositories.

- Finally, we observed three clusters of correlated variables (shown in red) that we decided to categorize as Popularity, Engagement and Authenticity.


forks_count
subscribers_count
open_issues_count
watchers_count
stargazers_count
has_wiki
has_downloads
has_issues
users
user_changes
anons
anon_changes
created_at
pushed_at
updated_at
fork
0.8
0.4
0.0
−0.4
−0.8

Results

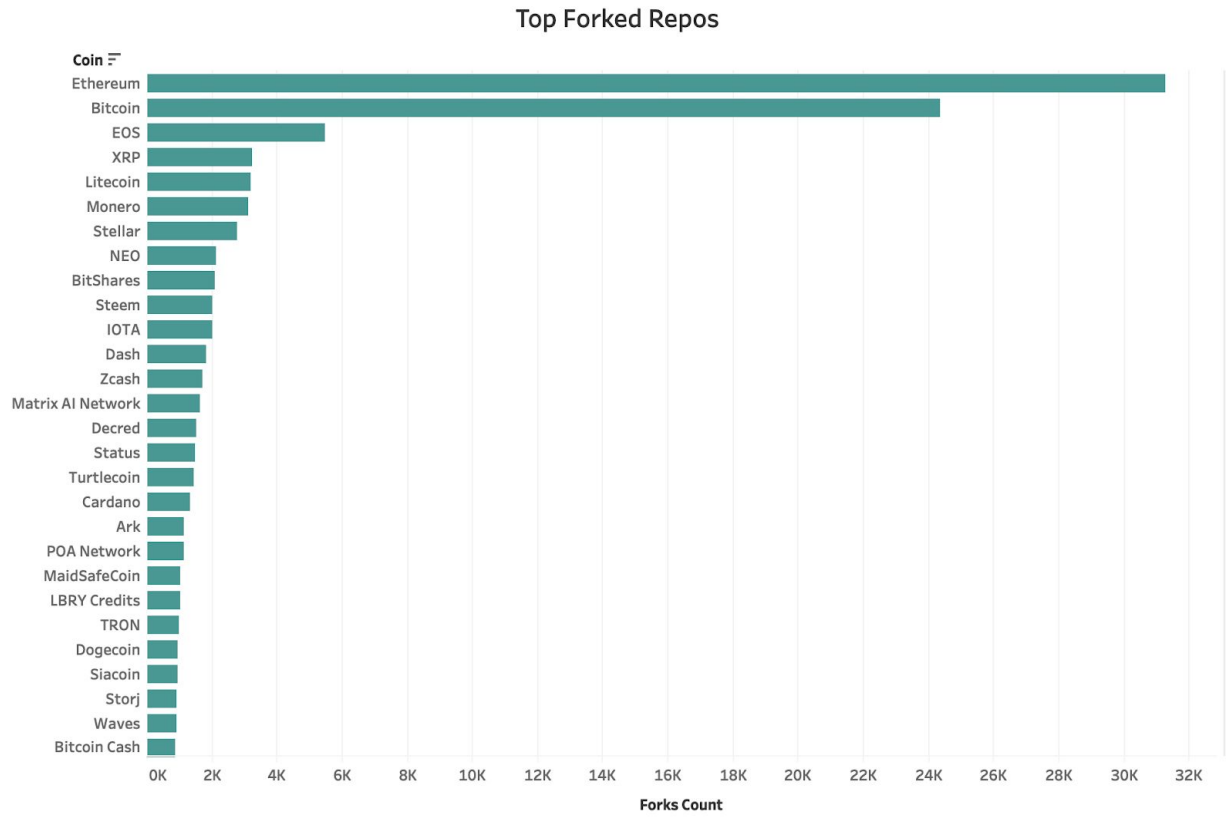
The following visuals are representative of the data we looked at over the course of the semester that enabled us to analyze the cryptocurrency dimensions of quality.

Insight Summary

Popularity

Most Forked Repositories

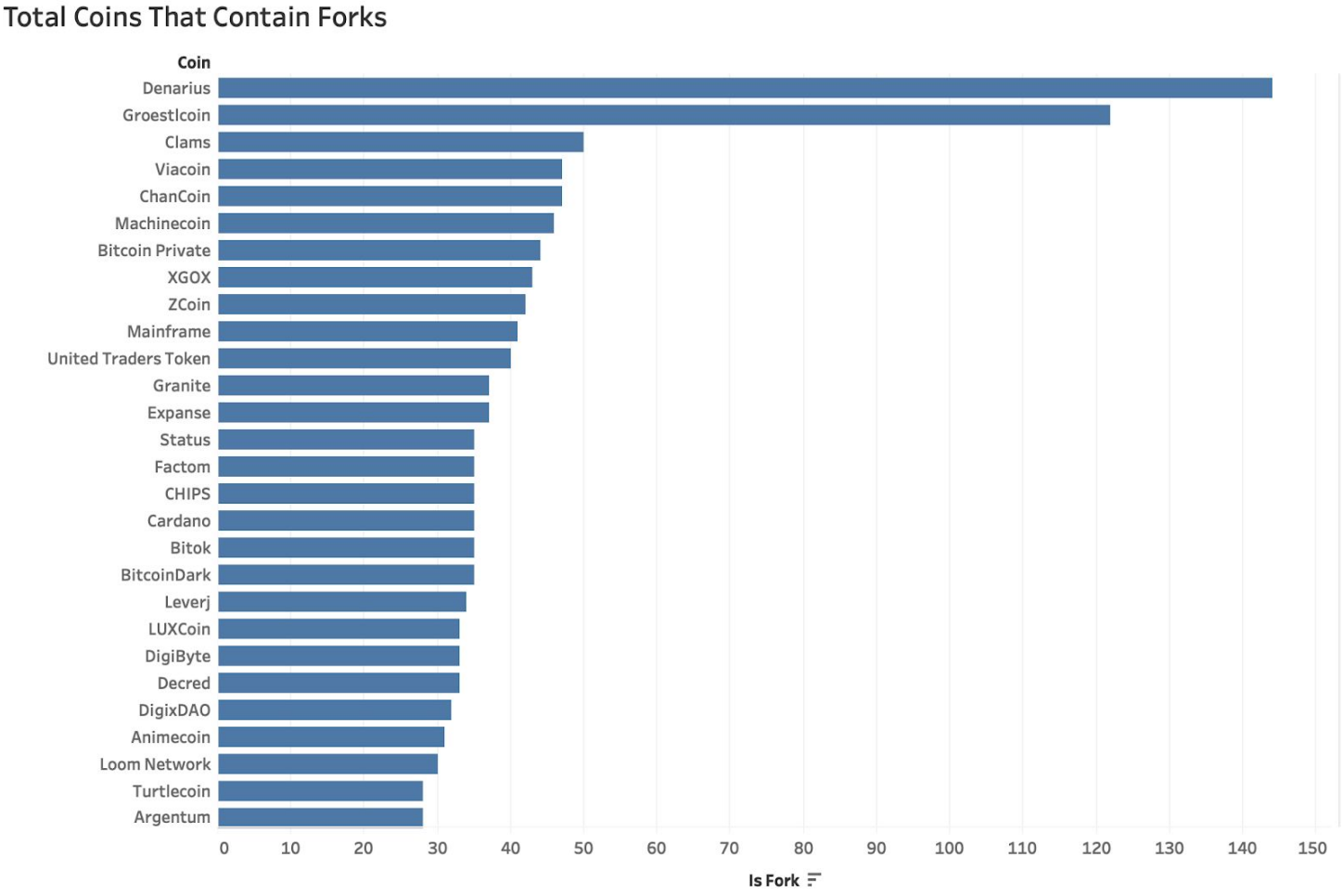
A fork is a personal copy of another user's repository that lives on one's account. Forks allow one to freely make changes to a project without affecting the original. Forks remain attached to the original, allowing one to submit a pull request to the original author to update with changes. One can also keep your fork up to date by pulling in updates from the original. The top forked repositories were Ethereum (go-ethereum), Bitcoin (bitcoin) and EOS (eos), indicating that these projects were the most popular in terms of development.



Total Number of Repositories that are Forks of other Projects

Upon analyzing the 12,141 repositories, 37% of those repositories are forks of other projects.

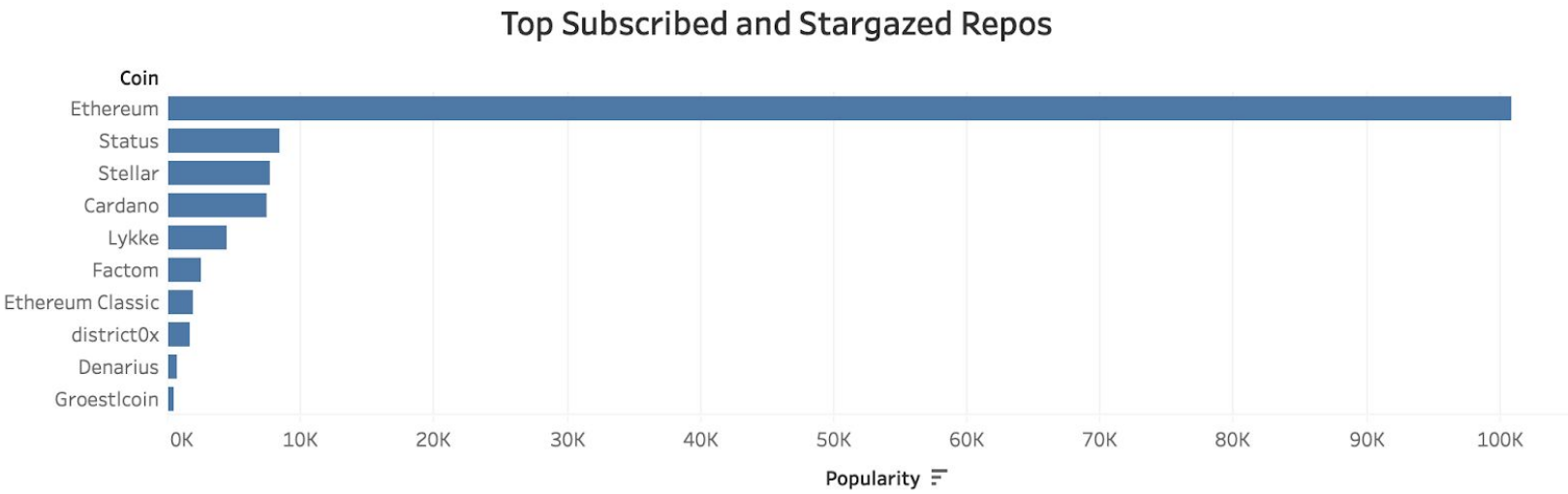
The following visual demonstrates which cryptocurrency projects are primarily forks of other projects: Danarius (82%), GroestIcoin (88%), Clams (89%).



Most Subscribed and Stargazed Repositories

Repository Starring is a feature that lets users bookmark repositories. Stars are shown next to repositories to show an approximate level of interest. Stars have no effect on notifications or the activity feed. Since the Watchers API key is deprecated in the Github

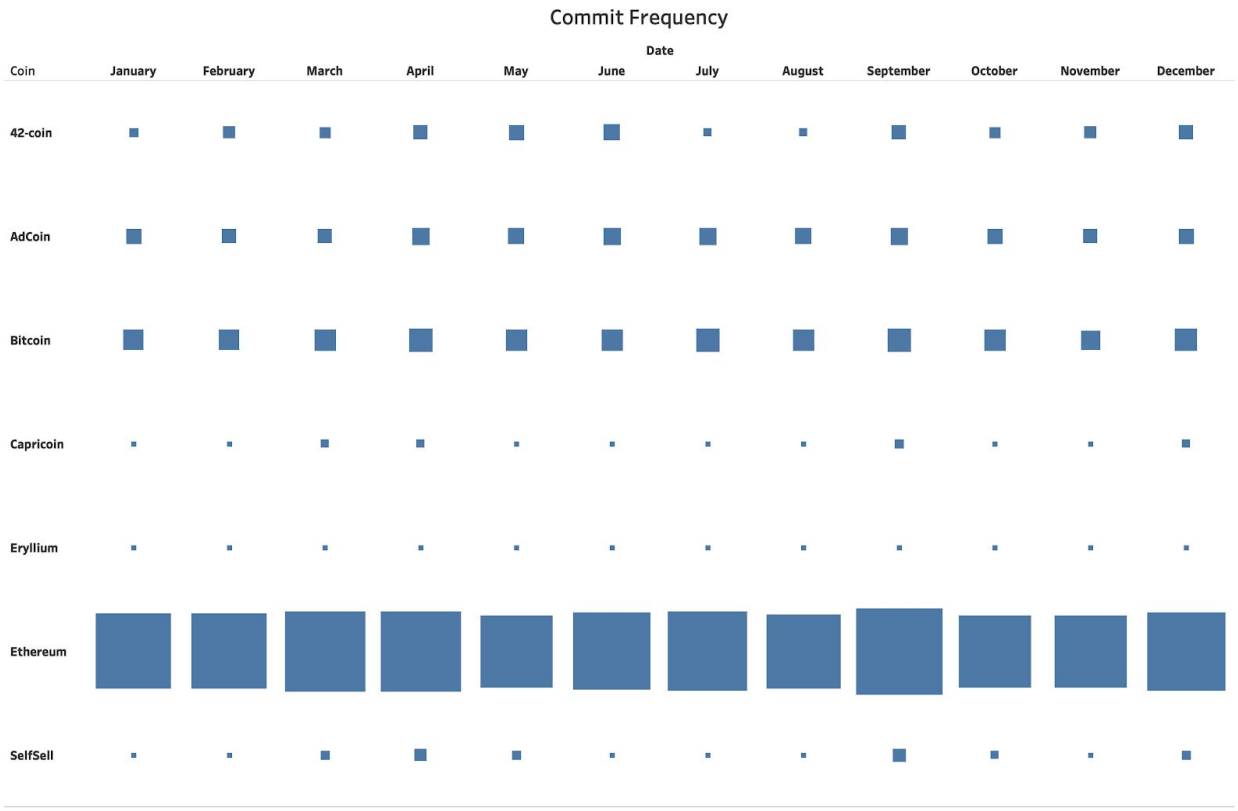
API, it was replaced with the Subscribed API key. One can subscribe to individual conversations in issues, pull requests, and team discussions, even if they're not watching the repository or a member of the team where the conversation is occurring. Upon analyzing the data, we found that the most subscribed and stargazed repositories were Ethereum (go-ethereum), Cardano (cardano-sl), Status (status-react).



Engagement

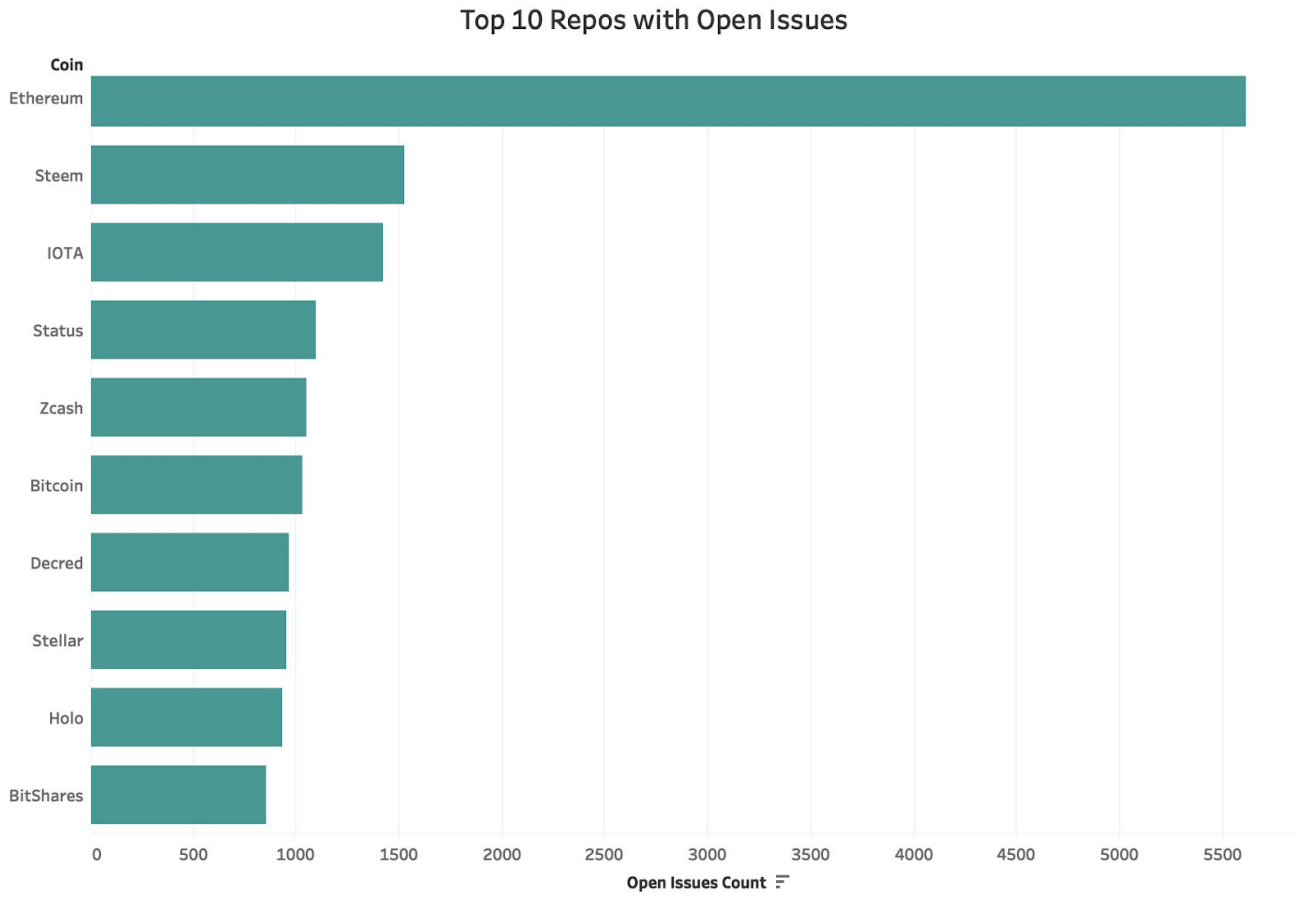
Commit Frequency

A commit, or "revision", is an individual change to a file (or set of files). It's like when you save a file, except with Git, every time you save it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who [10]. To better understand commit frequency on a granular level, we pulled commit frequency data (the summation of additions and deletions) over a span of a year and looked into seven cryptocurrency projects that fell into three categories: High, Medium and Low activity. From the figure below, you can see the quantified visual of the commits per each month over the year 2018.



Open Issue Count

Any user with pull access to a repository can create an issue if the repository has issues enabled, which is the default state of any new repository. If issues are disabled in the repository, the Github metadata API returned a 410 Gone status. Although we did not take Open Issue count into consideration during this round of dimensions for visual analysis we do factor it in as an input to our machine learning model.



Authenticity

Repositories with 'Issues' Disabled

As mentioned in the executive summary, out of the 12,141 repositories we investigated, ~2% of all repos have disabled issues or no open issues, which translates roughly to 45 coins that have completely disabled issues.

Repositories with Anonymous-only contributors

A contributor is someone who has contributed to a project by having a pull request merged but does not have collaborator access. Users can register for Github accounts or contribute anonymously using an unverified email account as their identity. Examples of coins we found that were comprised entirely of Anonymous users were Eryllium, PixieCoin, SecureCoin.


SecureCoin
Roofs
GET Protocol
NANJCOIN
NumusCash
SongCoin
CrevaCoin
Cosmo Coin
Crypto
MarxCoin
UGAIN
BriaCoin
RedCoin
Eryllium
Pura
Bibox Token
Vault Coin
Atmos
Aricoin
CompuCoin
Acoin
Plancoin
Pixie Coin
Level Up Coin
Joincoin
Ulatech
WhaleCoin

Support Vector Machines

A Support Vector Machine (SVM)[9] is a type of supervised learning algorithm that calculates a decision boundary to separate data points. The algorithm learns from a subset of labelled data used for training and outputs an optimal hyperplane to classify new information. The objective is to develop a function to minimize errors but also to prevent overfitting on the training data. The model needs to allow for generalization so it can continue to predict new information correctly. We believe this is a classification problem because the outputs we are trying to predict are discrete values. Alternatively, if we were trying to predict the price movement of a coin, our inputs and outputs would be timeseries making this a regression problem.

For our use case, we take the list of previously ranked repositories[6] and split them into two classes: the top 500 repositories are considered good quality and labelled as Class 1 while the remaining are Class 0. After performing a 10% random sampling to spot check these rankings we trust that the categorization is accurate even though the individual rankings may not be. The output from our model is a probability that a particular repository belongs to one of the two classes. For example: we expect the Bitcoin repos (ranked 1) to be predicted as Class 1 with a high probability. In the future we can expand this to a multiclass problem with higher granularity in the categories.

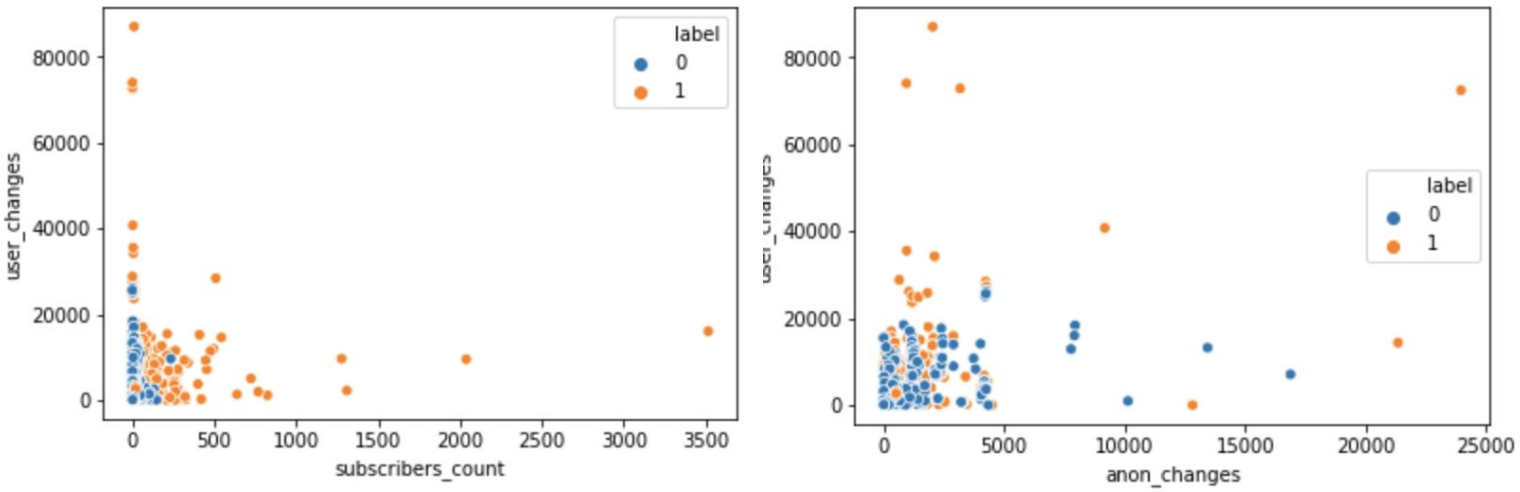
Feature Scaling

Since each of our features have vastly different numeric ranges, we to normalize our data using feature scaling. For example: *open_issues_count* may be in the range of tens or hundreds but user_changes may be in thousands. The scaler shrinks the range of values so they now fall in between -1 to 1 for all features. The distribution of our datapoints is not Gaussian as seen in the diagrams below. The Min-Max scaler implemented in sci-kit learn is the most common choice of scalers

$$\frac{x_i - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}$$

However due to the popularity of Bitcoin and Ethereum they end up being outliers in our dataset skewing the mean and standard deviations. Instead we choose the RobustScaler[11] which is sensitive to outliers as it uses interquartile ranges for standardization.

$$\frac{x_i - Q_1(\boldsymbol{x})}{Q_3(\boldsymbol{x}) - Q_1(\boldsymbol{x})}$$



Although in 2-D pairwise plots of our features it may appear that there is a decision boundary (hyperbolic in this case for the diagram on the left). While as another pair of features may seem inseparable. It is important to note that high dimensional vector spaces are impossible to visualize however the data may still be perfectly or nearly separable.

Training

After appropriately scaling our data we shuffled and split our dataset into half and used the one half for training our model and the other for testing. Training occurs in multiple iterations where parameters of the hyperplane are updated are adjusted at each step

using gradient descent. Two input configurations play an important role during training process: the loss function and the regularization parameter.

$$J(\Theta) = \left(\frac{1}{n} \sum_{i=1}^{n} \underbrace{L(h(x^{(i)}; \Theta), y^{(i)})}_{\text{loss}} \right) + \underbrace{\lambda}_{\text{constant}} \underbrace{R(\Theta)}_{\text{regularizer}}$$

At a high level the SVM objective can be expressed as shown above. At one end we try to minimize the loss, but at the other end we want our resulting function to not be dependent on this particular set of input values. This is known as overfitting. Therefore our regularization parameter results in some amount of misclassifications at the expense of generalizing our model to new input values. Finally, we evaluate the results by comparing the predicted values of the testing set to the actual labels.

Next Steps and Conclusion

As mentioned in the executive summary, we have laid down the foundation for understanding the complexities that engulf the term “quality” when it comes to evaluating open source projects. With the Github metadata API, we were able to scrape key data points from 12,141 repositories that made up about 1,011 coins. These data points enabled us to categorize the patterns we saw in the data in three bins - popularity, engagement and authenticity. These categorizations became the ‘dimensions of quality’ that we now use as signal inputs to our SVM model. As mentioned previously, SVM may not be the best representation to capture the relationships between our features and how they can be combined to make a final prediction (as a probability).

Over the next few weeks we would like to follow a two-prong approach to gain prediction accuracy. We would like to dedicate more resources and deploy a more complex function to model our data ex: Neural Networks or High Degree Polynomial kernels. Our second improvement would be to enhance our our quality attribute list and collect more data

including information about code reviews such as primary reviewers or keyword analysis in review comments.

Resource Report

Code Base

Our code is available for viewing on Github: https://github.mit.edu/kokje/tokenquality
We began collecting a series of funny stories we've come across as we dig through the bowels of Github repositories for research on 'token quality based on source code'. In each strip, we check out interesting-looking code, look up a token's perceived value and ask questions on public platforms. The Token Funnies are available for viewing on PubPub: https://viral.media.mit.edu/pub/tokenfunnies/

References

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. bitcoin.org/bitcoin.pdf.
[2] https://github.com/manganese/alteramentum-repo-data/blob/master/repos.csv
[3] https://files.sri.inf.ethz.ch/website/papers/diffcode-pldi2018.pdf
[4] https://www.flipsidecrypto.com/fcas-explained
[5] https://www.coindesk.com/data
[6] https://github.com/manganese/alteramentum-repo-data/blob/master/repos.csv
[7] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[8] https://developer.github.com/changes/2012-09-05-watcher-api/
[9] https://en.wikipedia.org/wiki/Support-vector_machine
[10] https://help.github.com/en/articles/github-glossary
[11] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html