Crypto Investor Profiler

Capstone project for Udacity Machine learning engineering Nanodegree

Author : Jan Osolnik
Amsterdam, April 2020

Definition

Project overview

The aim of this project is to combine blockchain data, machine learning and cloud managed services into a final product as a web app. Blockchain data is publicly available as a GCP public dataset.
The web app predicts the investor profile (cluster) of the Ethereum address that is entered into the user interface. This is based on three features that are extracted from the above-mentioned data source: Current balance, unique transfers, and unique tokens held. The project is formulated as an unsupervised machine learning problem.
In the finance lingo, an investor profile defines an individual's preference in investing decisions. Examples of this are risk-averse/risk-tolerant, diversity of asset classes and individual assets, investment in growth stocks or value stocks, etc.
In this project, it refers to any kind of investing behavior that can be quantified and used to differentiate between different Ethereum addresses.

It's important to note that a lot of code is built by combining code from different projects throughout the nanodegree program.. The aim is to create a workable solution and play with different services, not to optimize every part of the pipeline.

Problem statement

While crypto data on public blockchains are open, there are not many machine learning applications yet that leverage the available data. One of the reasons is that the underlying data structures are based on OLTP systems which are difficult to analyze (as opposed to OLAP). Recently, the data has been made available on Bigquery and is ready to be used. The challenge is in modeling the data and using it to feed the ML models. Also, the data is not labeled which is also why the problem is formulated as an unsupervised machine learning task. As pricing data is the most accessible, most of the crypto machine learning projects are focused on predicting future prices using that. On the other hand, this project is focused on getting a deeper insight into investor behavior.

This project explores the domain of crypto investor profiling. For every transaction that is executed on the public blockchains, there is a trace that we can analyze. As opposed to other asset classes, we have available data into the behavior of individual investors. The core aim of the project is to answer the question: can we use the principle of BYOD (bring your own data) and cluster an Ethereum address only based on its address hash?

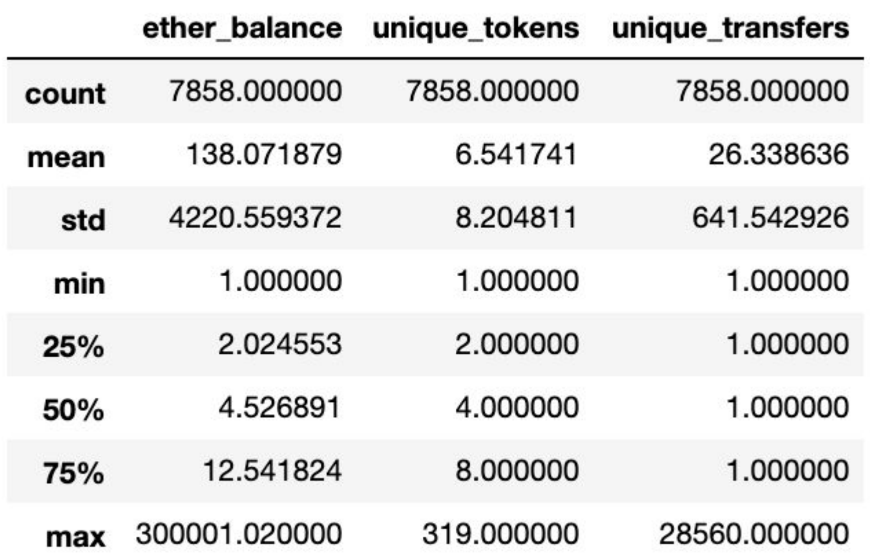
Metrics

With a given number of clusters, each model clusters all the Ethereum addresses into separate clusters. The metric used in this project is the Silhouette score. The coefficient is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample.
This is ideal for the use-case as we don't have ground truth values available. After a deeper analysis of various metrics after the project proposal, this was chosen as the best candidate to compare the performance of different clustering algorithms.

Analysis

Data exploration

A sample of 10000 Ethereum addresses are extracted. As some of them don't satisfy the further conditions in the specified query, about 8000 are available in the dataframe.



	ether_balance	unique_tokens	unique_transfers
count	7858.000000	7858.000000	7858.000000
mean	138.071879	6.541741	26.338636
std	4220.559372	8.204811	641.542926
min	1.000000	1.000000	1.000000
25%	2.024553	2.000000	1.000000
50%	4.526891	4.000000	1.000000
75%	12.541824	8.000000	1.000000
max	300001.020000	319.000000	28560.000000

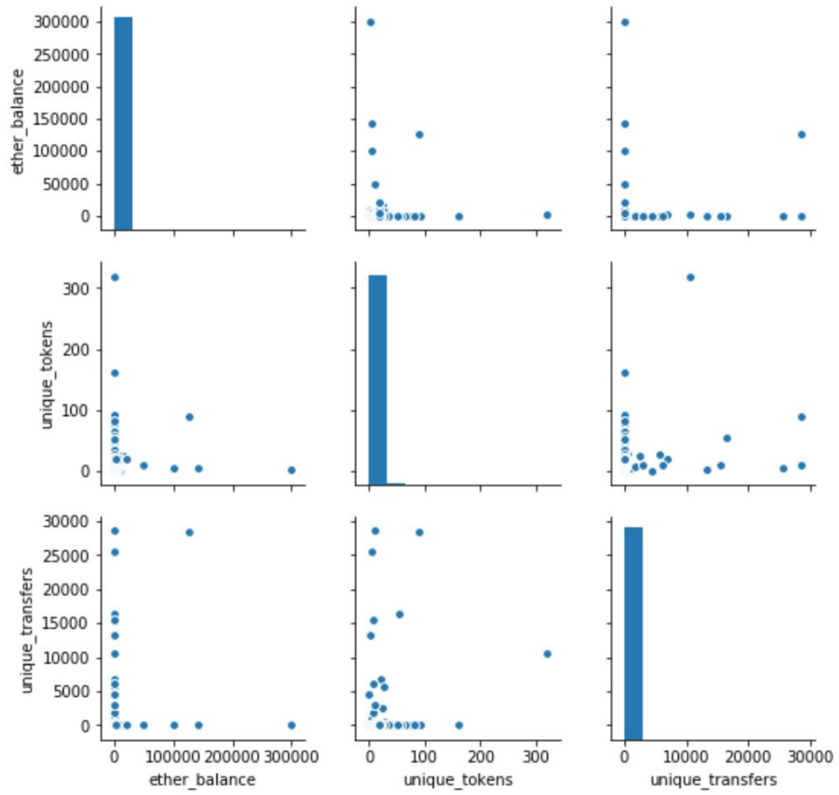
As is visible in the above table, the minimum threshold for *ether_balance* is 1 Ether. The mean is 138 Ether (almost 20k euros) which indicates the skewness of the distribution when we look at the 75th percentile value. While the 75th percentile is less than 13 Ether, the average is 138. There are clearly some strong outliers in the sample such as the largest value with 300000 Ether balance.

The *unique_tokens* feature is not as dispersed at the Ether balance. The usual value varies from 2 unique tokens to 8 tokens held.

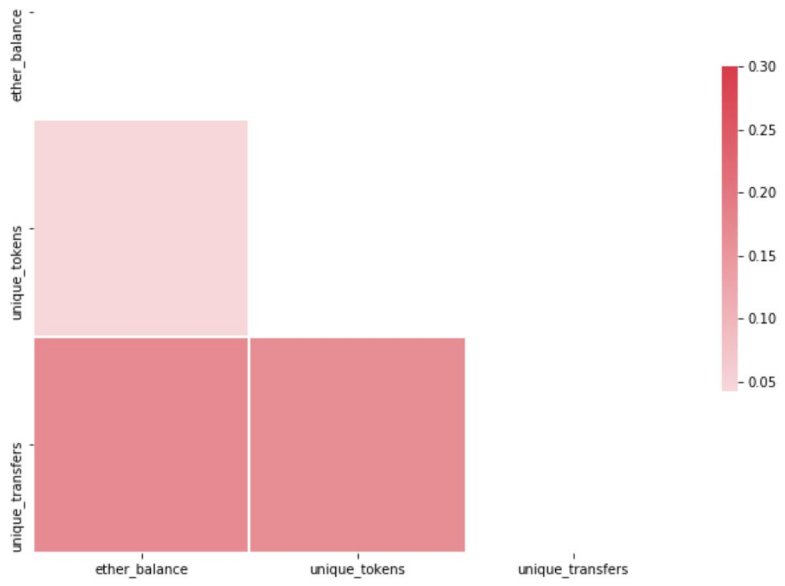
Unique_transfers feature varies widely. While the mean number of transfers is 26, the largest amount is 28560. We can see that the top 25% of addresses generate most of the transfers as none of the addresses in the 75 percentile has more than 1 transfer.

Exploratory visualization

There is a scatter matrix below for all constructed features. As it's clear in the visualization there is a power law distribution present in both *eth_balance* and *unique_transfers* features. The vast majority of the cumulative value of both is generated by a minority of addresses.
Meanwhile, there is not much correlation visible between different features.



Below there is a correlation heatmap for all available features. As it was already indicated above, there is little correlation between different features. That's a positive sign as we want to explore different areas of the feature space with different features. If there would be a strong correlation we could be explaining the same phenomena with different features.



Algorithms and techniques

The techniques used to predict clusters are all unsupervised machine learning algorithms. The chosen ones are K-means, Gaussian Mixture Models and Hierarchical clustering.
Because in K-means, the boundaries between clusters are always linear, the other two algorithms are used to handle more complicated boundaries. Also, the possibility to extend the analysis further by measuring the probability or uncertainty of cluster assignments. This is not done in the analysis but it's a possible extension.

Benchmark

The benchmark model for the project is the K-means algorithm's Silhouette Score. After that, more sophisticated algorithms are used to compare the performance. As a rule of thumb, a Silhouette score of 0.5 is defined as a threshold for the solution to be considered with a strong performance.

Methodology

Data Preprocessing

Features are extracted for a sample of 10000 Ethereum addresses that have at least 1 Ether in balance available. As there are currently about 100 million Ethereum addresses, a sample was extracted that still enables a certain level of signal to be extracted. As a lot of addresses have close to zero balance, the threshold to include an address in the sample was chosen at 1 Ether (currently around 140 euros).

Three features are extracted:

- Current Ethereum balance
- Unique tokens that have historically been held
- Unique transfers by the Ethereum address

Implementation

There are quite a few python modules used in the project. Meanwhile, these are some that are core for the implementation. Pandas_gbq library was used to query the BigQuery Ethereum data source. Google-cloud-bigquery was used to authenticate into GCP's service account. Scikit-learn was used as a solution for all machine learning algorithms.

AWS services used in the project:

- S3 to store training data, pre-processing transformer object and GCP credentials
- SageMaker to leverage notebook instances and connect the project end-to-end
- Lambda function to trigger the prediction script
- API Gateway to create an API that is used in the web UI that a user interacts with
- CloudWatch to log and debug possible issues that were encountered in the process of building the solution

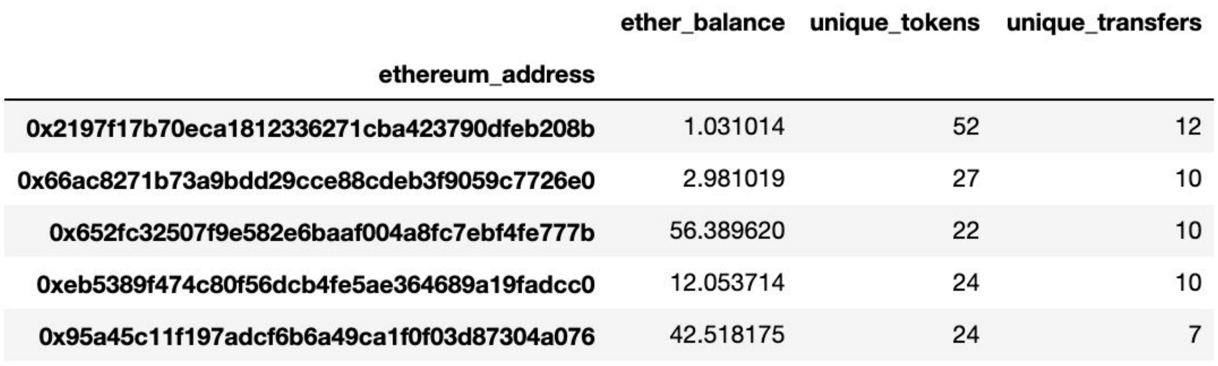
Two tables from the Ethereum public data source are used:

- `bigquery-public-data.crypto_ethereum.balances`
- `bigquery-public-data.ethereum_blockchain.token_transfers`

In the SQL query, we first extract the sample addresses in a CTE (common table expression). In the second CTE, we extract the top 1000 tokens in the Ethereum ecosystem based on transfer count.
By using the second CTE we extract for each Ethereum address the balance in a certain token address. Besides that, we extract the number of unique transfers for each of the addresses.
In the final query, we create a final view of features. We join the first CTE with the third CTE which was built using the second CTE. The feature set is built.

To construct the SQL queries and get a better understanding of the Ethereum data model Awesome Bigquery views Github page was used.

Below we can see some examples for the constructed features of different Ethereum addresses.

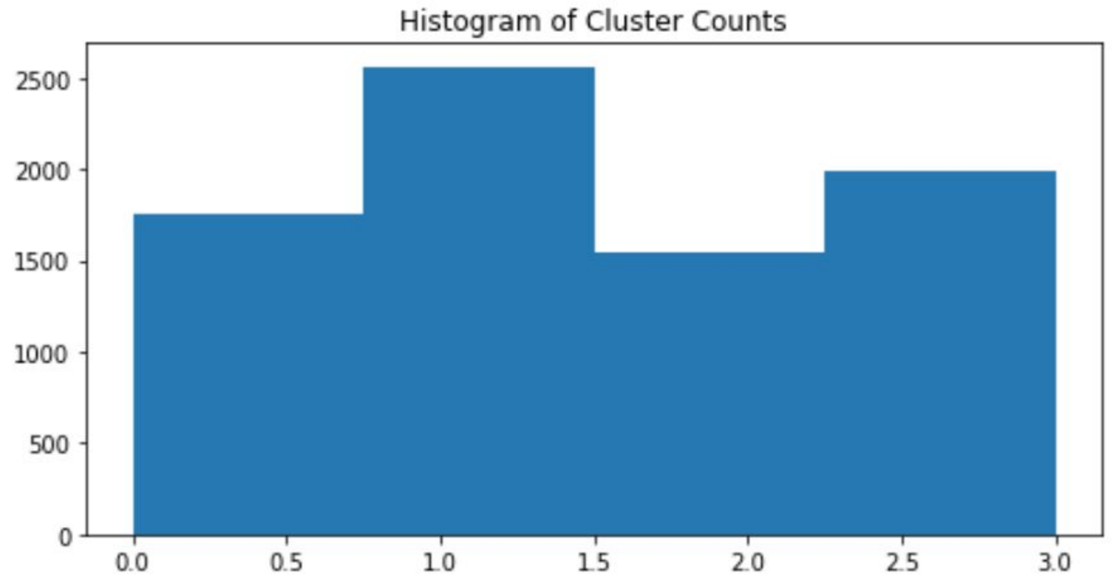


	ether_balance	unique_tokens	unique_transfers
ethereum_address			
0x2197f17b70eca1812336271cba423790dfeb208b	1.031014	52	12
0x66ac8271b73a9bdd29cce88cdeb3f9059c7726e0	2.981019	27	10
0x652fc32507f9e582e6baaf004a8fc7ebf4fe777b	56.389620	22	10
0xeb5389f474c80f56dcb4fe5ae364689a19fadcc0	12.053714	24	10
0x95a45c11f197adcf6b6a49ca1f0f03d87304a076	42.518175	24	7

After that, Sklearn's pipeline functionality is used to preprocess the features. We normalize them to make sure that the features are useful to feed into the models. This is done by using power transform, standard scaling and PCA transformation. PCA transformation might be redundant here as there are only three features mapped to three components but can be useful if the feature set is expanded and the dimensionality needs to be reduced considering the sample size.

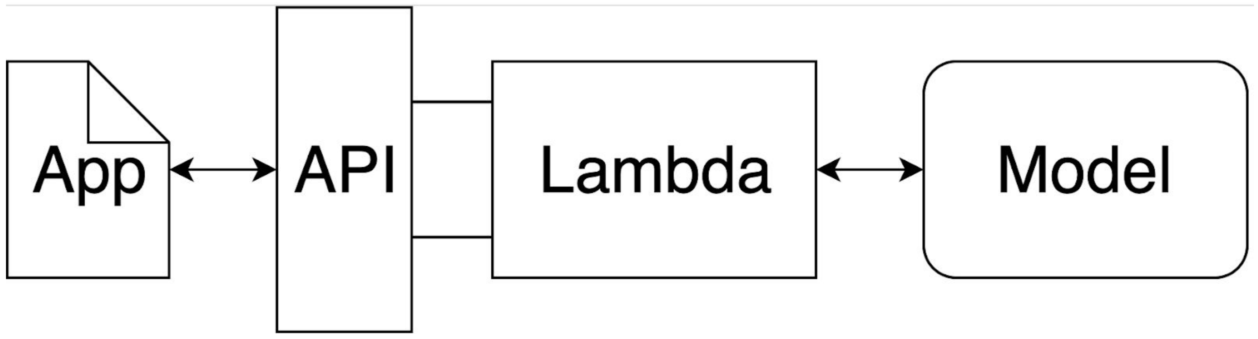
With the final set of features, all three clustering algorithm alternatives are trained: K-means model, GMM model, and Hierarchical clustering model.

The visualization below plots the number of instances in each of the 4 clusters.



The chosen number of clusters is 4. It's chosen based on trial and error of different values. As the performance of alternative algorithms don't improve the Silhouette coefficient considerably, K-means is used in the solution that is used in the model that is deployed.
After the training using the training script, the trained model's metadata is combined with the prediction script to predict the cluster value of individual Ethereum addresses.

The prediction script performs on-the-fly calculation of feature values of the provided Ethereum address. It queries the Ethereum data source and returns the normalized feature values. These are then used as input into the trained model which returns the predicted cluster value.



Source: Udacity MLND

The final solution the prediction endpoint triggered as a Lambda function through the API Gateway. The Lambda function is a cloud function based on serverless architecture. It is deployed through the API Gateway as an API that can be used in the user interface of the web app.
Whenever a user enters an Ethereum address and presses the Submit button, the predicted cluster value is shown in the user interface.

Refinement

The first version of preprocessing didn't include Sklearn's pipeline preprocessing steps. Without the normalization, the performance was poor as there is a lot of variation of variance between different features. Also, the number of sample addresses was increased from beginning 1000 to final 10000.
Other than that, not much refinement performed as the final goal was to construct a workable solution and connect the dots together.

Results

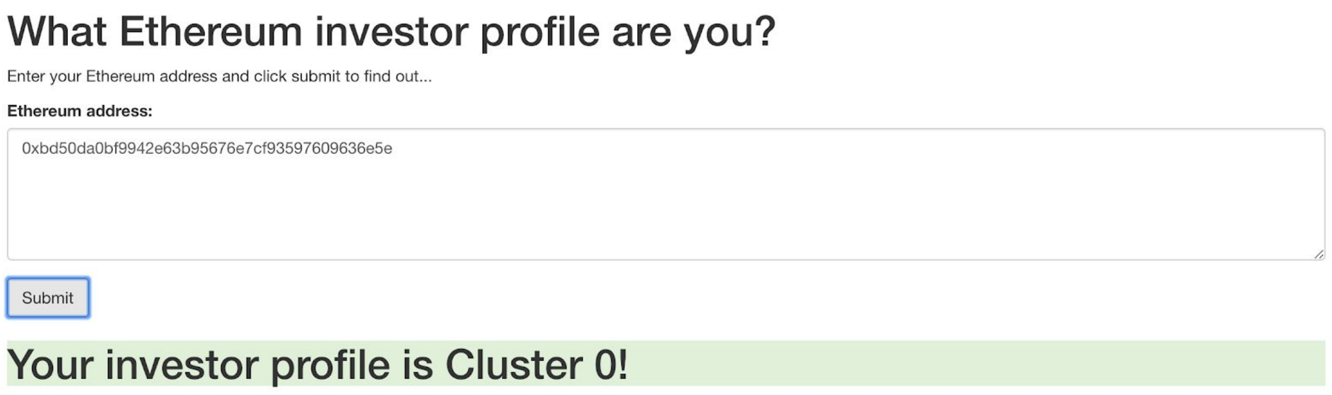
Model Evaluation and Validation

The final result is a web app that returns an investor profile based on the user's provided Ethereum address. The user doesn't need to provide any of their own personal data as all data that is needed can be extracted from the publicly-available Ethereum data source.

The models are evaluated based on the Silhouette score. As there are currently more than 100 million existing Ethereum addresses, it is hard to reliably sample the addresses without making some explicit rules such as filtering on non-trivial balance amounts.

The model is evaluated empirically with the chosen metric. K-means's Silhouette score is 0.38. As there are no ground truth values available there is no way to test this objectively in a scalable manner. On the other hand, the chosen metric quantifies well how different clusters are differentiated in the feature space.
The final solution is tested in a web app where the predicted cluster value is extracted for a sample Ethereum address.

Considering the performance it's clear that there is not enough variance in the data extracted to be able to create interpretable clusters. When looking at different clusters it was not possible to find meaningful naming of clusters.



What Ethereum investor profile are you?

Enter your Ethereum address and click submit to find out...

Ethereum address:

0xbd50da0bf9942e63b95676e7cf93597609636e5e

Your investor profile is Cluster 0!

Justification

As the performance of alternative algorithms are not considerably better than K-means, the simpler solution is used for the final solution. The 0.5 threshold for Silhouette coefficient is not achieved which means that considering this metric we can't describe the models to have a strong grasp of the underlying data. Meanwhile, the main goal of the project was to build a minimum viable solution that can then be improved on.

The biggest bottlenecks of performance are the features that are fed into the models. With a better understanding of the underlying data model we could construct a better feature set.

Besides that, the preprocessing could be improved. Also, the number of clusters was chosen with trial and error with the same values across all algorithms. This could be sped up by using some more sophisticated analysis (Silhouette analysis).

There are also other possible improvements but considering the complexity of the project, the pre-defined goals are definitely satisfied.