Objective: Quantification of degree of similarity of nodes and identification of nodes belonging to the same entity/user using supervised machine learning

In a bitcoin blockchain system, user identity is preserved and there is no certain way of ascertaining the same. In cases where the two public appear as input in a transaction or data revealed in the form of leaks, we can ascertain that some public keys belong to the same users or the same entity/organisation. Unsupervised approaches like k-means clustering are used commonly to cluster user address based on transactional patterns,etc. There is a limited scope of using supervised learning algorithms on bitcoin blockchain as there is no notion of labels. Some of the works on bitcoin blockchain data have provided data of lists of user address that probably belongs to the same user. We can use these as pseudo-labels as we cannot for certainty say whether two address belongs to same person unless revealed by themselves. The objective of this work is to use such pseudo-labels to train a machine learning model such that given a query user address, we can quantify the degree of similarity between two user nodes and can retrieve top-k similar user address existing in the blockchain network .

Dataset: The authors of [1] created a bitcoin data and has provided a list of user address which probably belongs to the same user id. The number of unique user address are 24618958. To reduce the computational complexity, I only consider first 80k user addresses.

Feature Extraction: For each user id, I use the following details:

1. Number of distinct addresses which appears as inputs in transactions where this user id appears as output
2. Number of distinct addresses which appear as outputs in transactions where this address appears.
3. Total amount of transaction value received
4. Mean and standard deviation of transaction value received
5. Max value of amount received in one transaction
6. Min value of amount received in on transaction
7. Total amount of transaction value sent
8. Mean and standard deviation of transaction value sent
9. Max value of amount sent in one transaction
10. Min value of amount sent in on transaction
11. Net balance at the end of 277442 blocks

Creating Training Data: For each user id belonging to the first 80k id’s, make lists of user id belonging to the same person using the data provided. So we will have two distinct lists of the formats as shown below:

1. [[user1,user3232,user24244],[user4,user4342],..........]
2. [4,8,9,6,123,144…]

Row 1 has sublists where each sublist has user id’s belonging to the same person. Row 2 list has independent user id’s

Now, for each sublist in row1, create mutually exclusive pairs. For example : (user1,user2323),(user1,user24244),(user3232,user24244). I call this as positive pairs i.e pairs of user id belonging to the same entity.

Similarly, for each user id in each sublist, I pair that user id with a total of 10 user ids taken from either an element from row2 or an element form a different sublist. I call such pairs as negative pairs. Using this we can generate a huge list of pairs. However, I limit myself to 210608 positive pair and 298090 negative pairs.

The question remains now is how to generate feature vectors using two user ids belonging to a pair. Let’s say we have a positive pair (user1,user2). We extract feature vectors F1,F2 corresponding to user1 and user2. I used cosine distance as a method to generate a unique vector F3 for the pair (F1,F2). The process is illustrated below

Dimension of F1=[1,13]
Dimension of F2=[1,13]

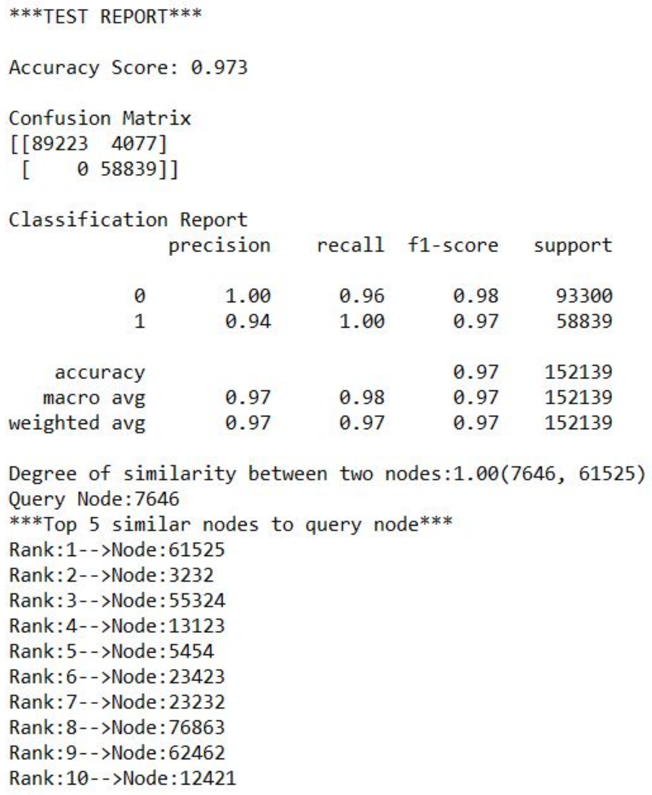
Cosine(F1,F2)=(F1*F2)/||F1||.||F2|| where * indicates element wise product of two vectors and . indicates scalar multiplication of two values. Thus, we have a created a labelled data of the form (F,y) .

I labeled all the positive pairs as 1 and all the negative pairs as 0. The intuition behind this scheme is simple: each key is paired with one positive and 10 negatives thus inherently forcing the network to learn parameters in a corresponding manner such that if a pair (key1,key2) are same then the probability that a given key pair is similar is close to 1 else close to 0. The benefit of such a learning approach is that let’s say a new node is added to a network , we can pair the node with other existing nodes, pass is through the trained model and sort the output probabilities and retrieve top-K similar nodes in the network.

Training Details: After creating a well shuffled data, I split the data into two fractions 70:30 for training and test. For the training split, I use 10-fold Cross validation for hyperparameter optimization and prevent over-fitting. I used RandomForestClassifier with number of estimators equal to 100 to fit the training data. After fitting a classifier model, I used sklearn

CalibratedClassifierCV to calibrate the already fitted model to output probability scores for each class.

Results: I achieved an accuracy of 97.4% for binary classification indicating the indeed this method could work. To test the model, I a query address from test set and some random address which might and might not be similar. I retrieve the top 10 addresses which are similar to the query address indicating that the retrieved nodes are similar.



```
***TEST REPORT***

Accuracy Score: 0.973

Confusion Matrix
[[89223  4077]
 [    0 58839]]

Classification Report
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     93300
           1       0.94      1.00      0.97     58839

    accuracy                           0.97    152139
   macro avg       0.97      0.98      0.97    152139
weighted avg       0.97      0.97      0.97    152139

Degree of similarity between two nodes:1.00(7646, 61525)
Query Node:7646
***Top 5 similar nodes to query node***
Rank:1-->Node:61525
Rank:2-->Node:3232
Rank:3-->Node:55324
Rank:4-->Node:13123
Rank:5-->Node:5454
Rank:6-->Node:23423
Rank:7-->Node:23232
Rank:8-->Node:76863
Rank:9-->Node:62462
Rank:10-->Node:12421
```

Interpreting the results of trained model: The final performance of the model depends on how well the features have been extracted. Since, blockchain data is extremely huge requiring good computational resources,feature extraction from transactions is tough and limits my submission. However, it can be concluded that indeed the degree of similarity between two nodes can be found out which may help immensely in taking decisions and further strengthening the bitcoin blockchain system.As you can see, the degree of similarity is 1.0, which means the two node is highly similar. However, it can be interpreted in a different way. Since, I use only 13 features, which is not discriminative to the very best, hence with limited features the model is naturally bound to output a probability of 1.0 if he finds two nodes as extremely similar and vice versa, given that it has been trained in such a manner. So, what happens at present at that even if the nodes might not be similar to a degree of <0.6 , my model would output a value close to 0.9 if the features match to a great extent. If there had been more discriminative features, I would expect to not get an exact similarity score as around 1.0 . With a more proper feature engineering on a big scale, it’s safe to say that we can use this method to say whether two nodes belong to the same organisation/user/etc.

References:

[1]:https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0086197
[2]:http://www.vo.elte.hu/bitcoin/zipdescription.htm (DATASET LINK)