



Detecting Malicious Accounts in Permissionless Blockchains using Temporal Graph Properties

Rachit Agarwal
Shikhar Barve
Sandeep Kumar Shukla

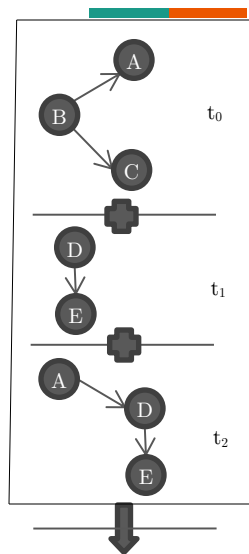
CSE - IITK

Outline

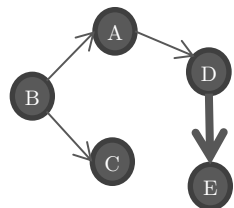


- Motivation
- Related Work
- Our Approach
 - Experiment Settings
 - Feature Extraction
- Machine Learning
 - Supervised Learning
 - Unsupervised Learning
- Behavioural Analysis
- Validation
- Conclusion

Motivation



- Permissionless blockchains mainly deal with crypto-currencies and are **prone to cyber attacks**, scams and ransom payments
- Can we train ML model to detect such activities and generate alerts?
 - Blockchain Transaction graph is a **Temporal Graph**
 - Current existing techniques use aggregated snapshot to perform analysis of malicious activity and thus **neglect temporal aspects** such as **behaviour changes** over time
 - They use graph properties such as inDegree, outDegree and clustering coefficient on top of blockchain specific properties such as transaction amount.



Motivation

- We find that each account behaves differently and features follow a distribution.
- Distribution of inDegree and outDegree follows **power-law** suggesting **bursty behaviour**.

Figure: 1

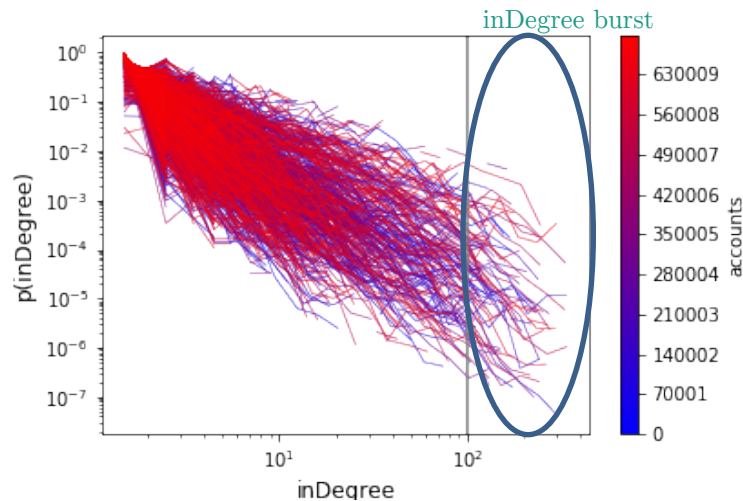
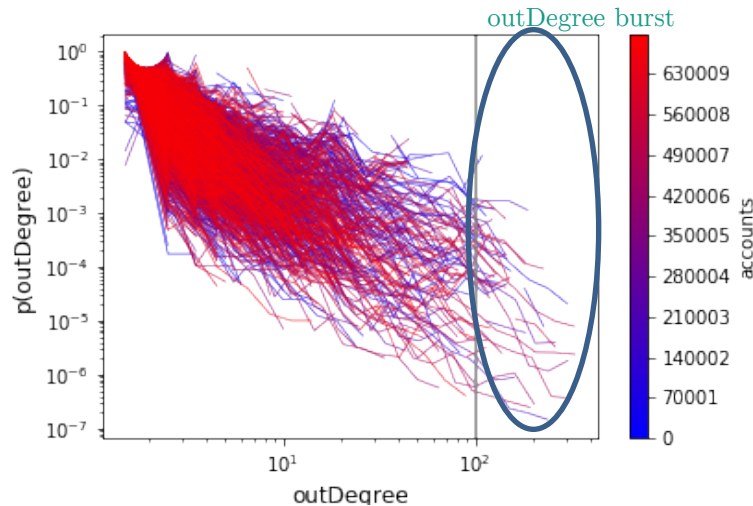
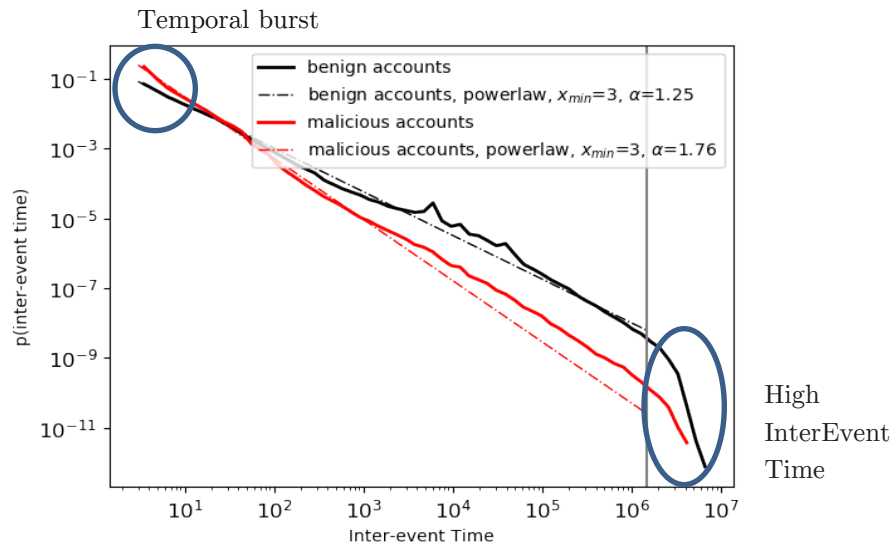


Figure: 2



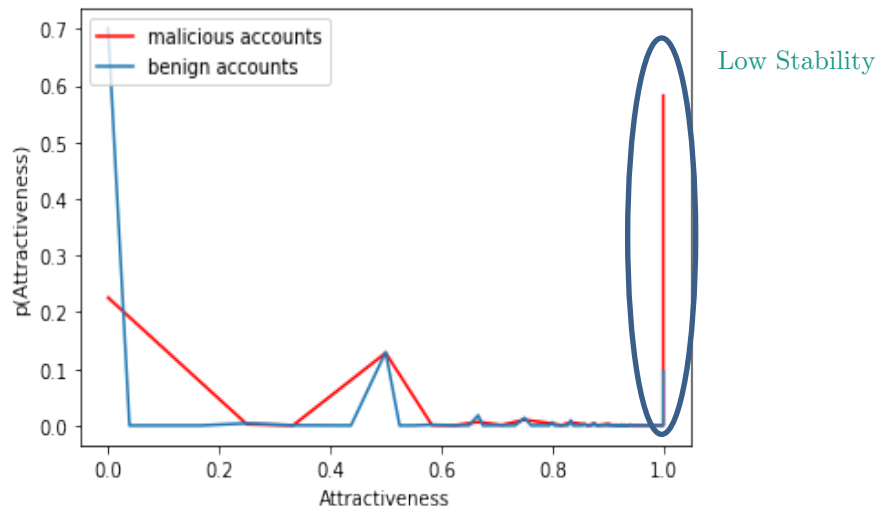
Motivation

- We find that each account behaves differently and features follow a distribution.
- Distribution of Inter-Event Times follows truncated power law suggesting **Temporal Burstiness**.



Motivation

- We find that each account behaves differently and features follow a distribution.
- Most Malicious accounts interact with accounts that they have not interacted in past (Low Stability of Neighbourhood) .

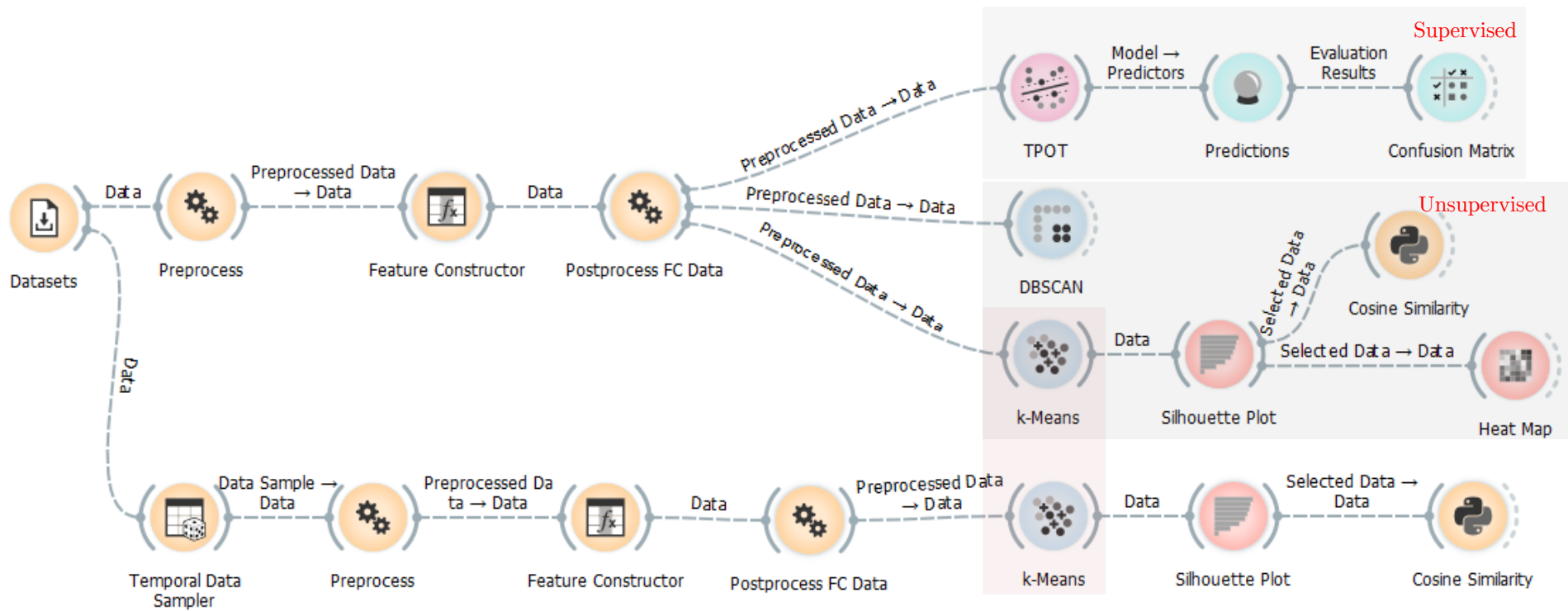


Related Work

		Used features based on													
#	B/C	AS	iD	oD	Bal	TF	BB	A	CC	IFT	ML Algo Used	Dataset	Hyperparameters	Performance	
[17]	B	✓	✓	✓	✓	-	-	-	✓	✓	K-Means	100K ^a	$k \in [1, 14]$	$k_{opt} = 7, 8$	
											Mahalanobis Dis.		\times	0.0256 ^{MDE}	
											ν -SVM		$\nu = 0.005$	0.1441 ^{MDE}	
[18]	B	✓	✓	✓	✓	-	-	-	-	✓	LOF	6.3M ^a	$k = 8$	0.55 ^{MDE}	
[19]	B	-	✓	✓	✓	-	-	-	✓	-	K-Means	1M ^a	$k \in [1, 14]$	$k_{opt} = 8$	
											Trimmed K-Means		$k \in [1, 15], \alpha = 0.01$	$k_{opt} = 8$	
[20]	B	✓	✓	✓	✓	-	-	-	-	✓	RIPPER†	‡6432 ^a	$cost \in [1, 40]$	0.996 ^{ac}	
											Bayes Network		\times	0.983 ^{ac}	
											Random Forest		\times	0.996 ^{ac}	
[21]	E	-	✓	✓	✓	✓	-	-	-	-	XGBoost	‡1382 ^{sc}	\times	0.94 ^p , 0.81 ^r	
[23]	E	✓	✓	✓	-	✓	-	-	-	-	Random Forest	350K ^a	RFPARAM	0.85 ^r , 0.05 ^p	
											SVM		$cost = 1, \gamma = 0.077$	0.87 ^r , 0.02 ^p	
											XGBoost		XGBPARAM	0.8 ^r , 0.07 ^p	
[24]	E	✓	✓	✓	✓	-	-	-	-	-	Decision Tree	300 ^a	\times	0.93 ^{ac}	
											SVM		\times	0.83 ^{ac}	
											KNN		$k = 5$	0.91 ^{ac}	
											MLP		\times	0.86 ^{ac}	
											NaiveBayes		\times	0.89 ^{ac}	
											Random Forest		\times	0.99 ^{ac}	
[25]	E	✓	-	-	✓	✓	-	-	-	-	Decision Tree	9375 ^a	\times	0.92 ^{ac}	
											KNN		\times	0.92 ^{ac}	
											XGBoost		\times	0.96 ^{ac}	
											Random Forest		\times	0.95 ^{ac}	
[26]	B	-	✓	✓	✓	✓	-	-	-	-	Adaboost	1000M ^a	$estimators = 50, rate = 1$	$> 0.2^r$	
											Random Forest		$estimators = 10$	$> 0.85^r$	
											Gradient boosting		$estimators = 100, rate = 0.1$ $depth = 3$	$> 0.93^r$	

B/C Blockchain, ^B Bitcoin, ^E Ethereum, ^a accounts, ^{sc} smart contracts, ^{MDE} Dual Evaluation Metric, ^{ac} accuracy, ^p Precision, ^r Recall, † it learns classification rules via sequential covering, ‡ Ponzi scheme data, ^{RFPARM} features = 3, leaf samples = 10, threshold probability = 0.99, ^{XGBPARAM} depth = 3, child weight = 8, subsample = 1, probability = 0.99, \times not provided.

Our Approach



Experiment Setting




Datasets

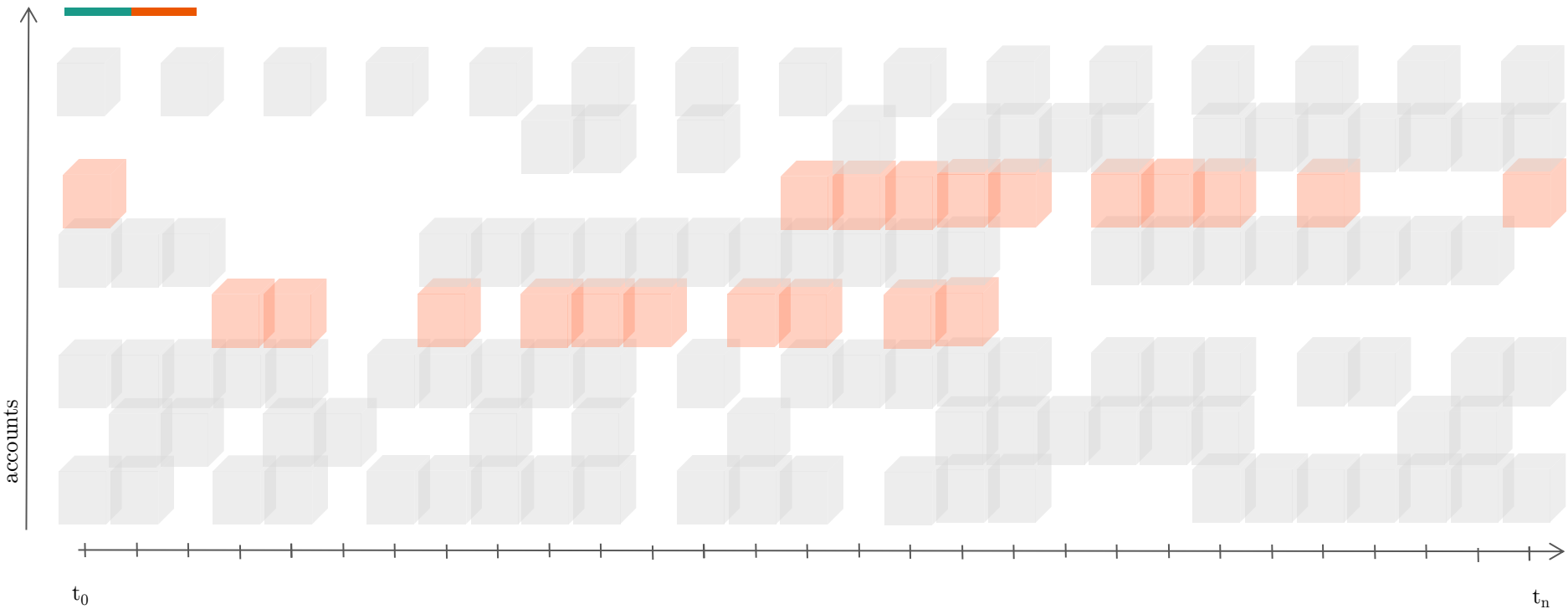
- Ethereum Mainnet dataset (under-sampled from 1:23500 to 1:233).
 - 700k (approx 697k benign + 3k malicious) accounts.
 - Two types of accounts : EOAs and SC.
 - Malicious accounts those involved in Phishing, Scams, Hacks etc.
 - Entire transaction history of these accounts from genesis block until 7th December 2019
- Preprocessing stage involves data cleaning and construction of temporal graphs.



Preprocess

Representation of accounts wrt time

 Block in which an account transacted





Features Extraction/Construction

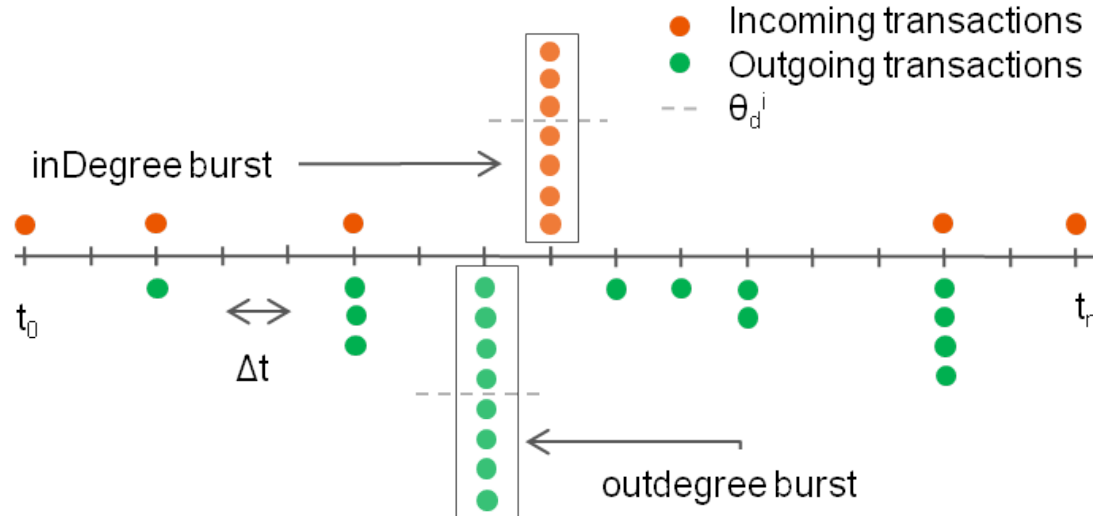
- Features motivated from previous attacks, scams and ransom payments.
- Burst based
 - Degree, Temporal, Txn. Amount, GasPrice
- Time Series based
 - In/Out Degree, BalanceIn/Out, MaxIn/OutPayment, GasPrice, Attractiveness and Inter-event time
- Non Time Series based
- Existing feature set :
 - TransactedFirst, TransactedLast, ActiveDuration, LastActiveSince, ZeroTransactions, UniqueIn, Clustering Coefficient

Burstiness: Degree



Feature Constructor

- More than Θ_d events happening at the same time.
- Attacks such as All-in-vain theft uses such feature.

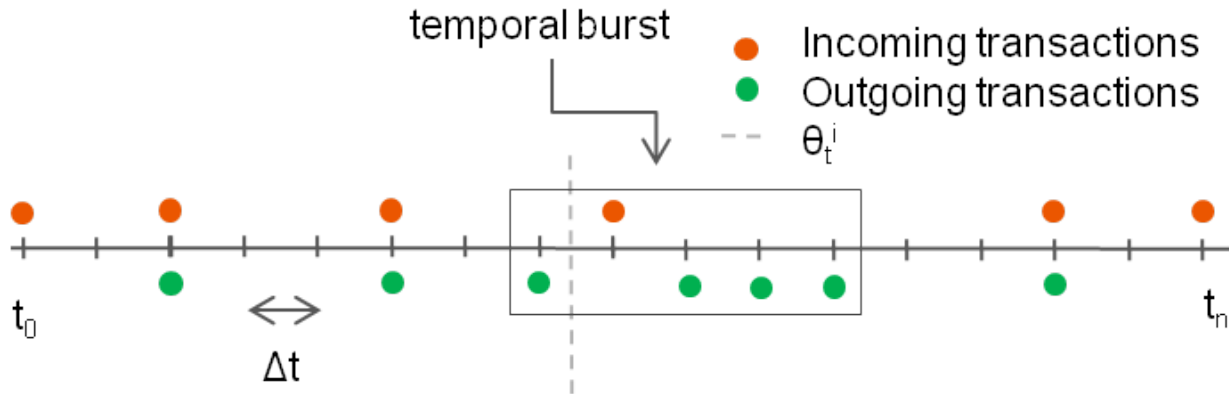


Burstiness: Temporal



Feature Constructor

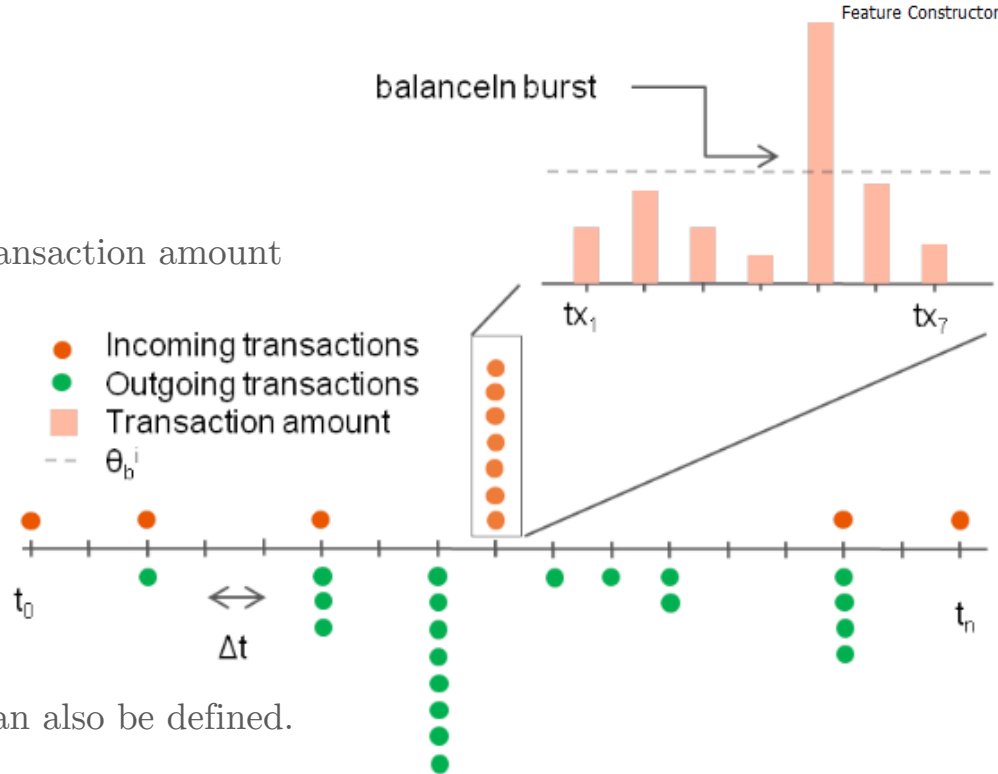
- Events happening for at least Θ_t consecutive time instances.
- Accounts involved in Gambling show such behaviour.



Burstiness: Txn. Amount



- Transaction amount more than Θ_b .
- Silk Road accounts showed such transaction amount patterns.



- Similarly, Burstiness for gasPrice can also be defined.
 - $\text{Txn_Fee} = \text{gasUsed} * \text{gasPrice}$

Attractiveness

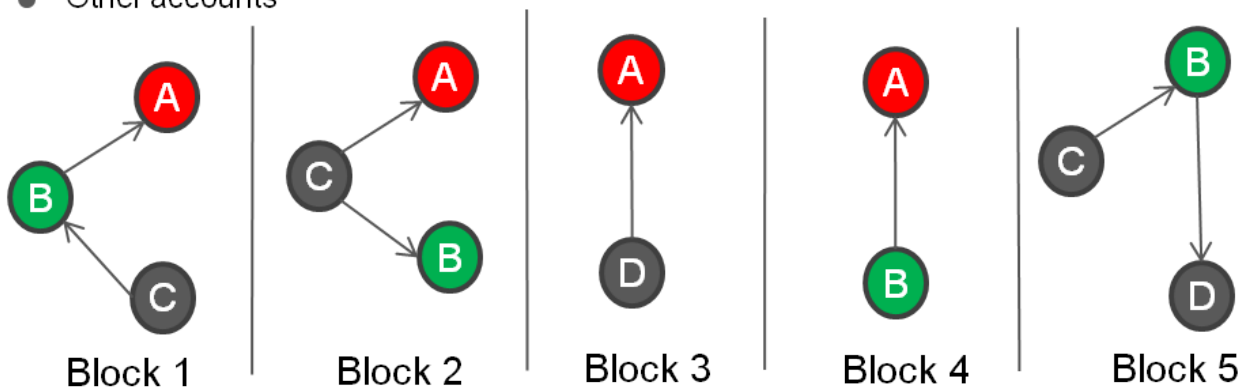


Feature Constructor

- Use notion of stability of neighbourhood.
- N_i^t neighbourhood of account i at time t .
- $T = \{t, t-1, \dots, t - \Theta_a\}$; Θ_a = duration of dataset

$$A_i^t = \begin{cases} 1 - \frac{|N_i^{t,in} \cap (\bigcup_{j \in T - \{t\}} N_i^{j,in})|}{|\bigcup_{j \in T} N_i^{j,in}|}, & \text{when } j \geq 0 \text{ and } N_i^{t,in} \neq \emptyset \\ 0, & \text{otherwise.} \end{cases}$$

- Most Attractive accounts
- Least Attractive accounts
- Other accounts



Features Post-processing



Postprocess FC Data

- Time Series based features such as In/Out Degree need to be further characterized into features.
- Tsfresh is used to extract features from such time series.
- It provides more than 400 features for each time series.
- We use top 3 features that have high gini coefficient.
- Many of the features were correlated.
- This resulted into a total of 59 features, again out of which some features were correlated. Only 36 features remain after the reduction.
 - We also used PCA to identify 28 principal component conserving nearly 98 % variance.

Supervised Learning



- Using AutoML tool called TPOT
- Configured to techniques used in related work and more.
- Applied to different subsets of the datasets to get best classifiers in each case.
- Used balanced accuracy as a performance measure.
- Extra Tree Classifier performs best with respect to balanced accuracy.

Features	Data Segment	TPOT identified Classifier	Balanced Accuracy	Precision		Recall		F1 score	
				Mal	Ben	Mal	Ben	Mal	Ben
28 (PCA)	Only EOA	ExtraTrees	0.872	0.38	1.00	0.75	0.99	0.50	1.00
	EOA and SC	ExtraTrees	0.873	0.22	1.00	0.76	0.99	0.34	0.99
36	Only EOA	ExtraTrees	0.876	0.11	1.00	0.78	0.97	0.19	0.99
	EOA and SC	ExtraTrees	0.882	0.24	1.00	0.78	0.99	0.37	0.99
59	Only EOA	ExtraTrees	0.881	0.26	1.00	0.77	0.99	0.38	0.99
	EOA and SC	ExtraTrees	0.887	0.29	1.00	0.78	0.99	0.42	1.00

28 (PCA) EOA ExtraTreesClassifier (class_weight = 'balanced', max_features = 0.4, max_samples = 0.3, min_samples_leaf = 11, min_samples_split = 19, n_estimators = 600)

28 (PCA) EOA and SC ExtraTreesClassifier (class_weight = 'balanced', criterion = 'entropy', max_features = 0.25, max_samples = 0.15, min_samples_leaf = 13, min_samples_split = 4, n_estimators = 800, n_jobs = 20, random_state = 100)

36 EOA ExtraTreesClassifier (bootstrap = true, class_weight = 'balanced', max_features = 0.15, max_samples = 0.7, min_samples_leaf = 8, min_samples_split = 18, n_estimators = 200, n_jobs = 10, random_state = 100)

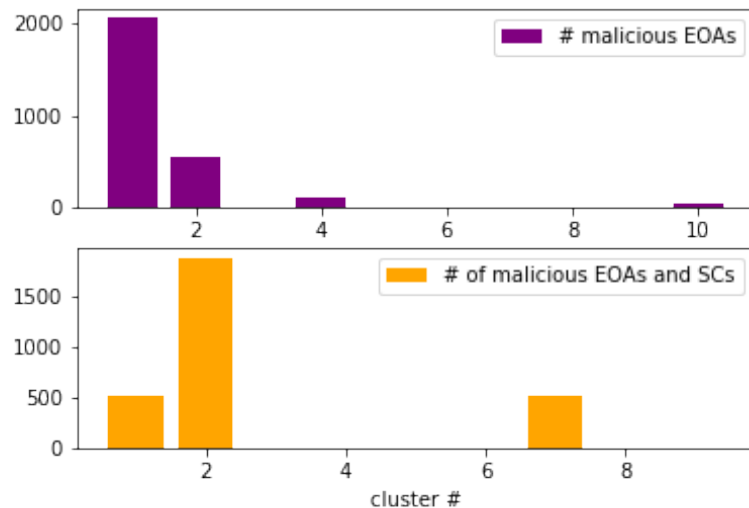
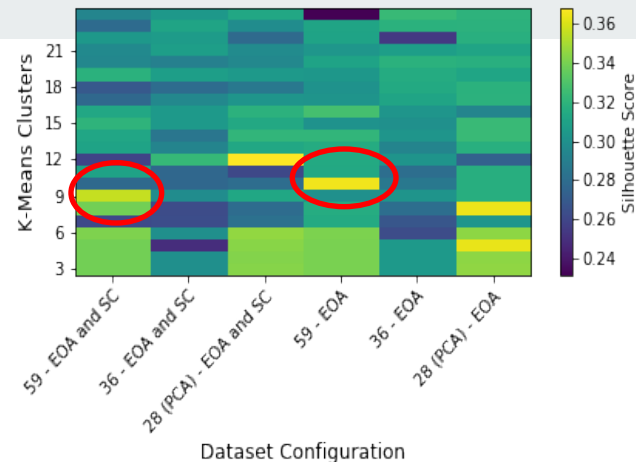
36 EOA and SC ExtraTreesClassifier (class_weight = 'balanced', criterion = 'entropy', max_features = 0.45, max_samples = 0.75, min_samples_leaf = 18, min_samples_split = 6, n_estimators = 200)

59 EOA ExtraTreesClassifier (class_weight = 'balanced', max_features = 0.2, max_samples = 0.75, min_samples_leaf = 13, min_samples_split = 19)

59 EOA and SC ExtraTreesClassifier (class_weight = 'balanced', criterion = 'entropy', max_features = 0.3, max_samples = 0.3, min_samples_leaf = 14, min_samples_split = 20, n_estimators = 200)

Unsupervised Learning

- K-Means performed best and identified 10 clusters for one configuration (9 for another).
- We chose the cluster with most malicious nodes among identified clusters and identify behavioural similarity between malicious and benign accounts in that cluster.
 - Use cosine similarity score (s_{ij}) between malicious node i and benign node j
 - If $s_{ij} \geq 1 - \epsilon$. Account j is Malicious
 - Here $\epsilon = 10^{-7}$

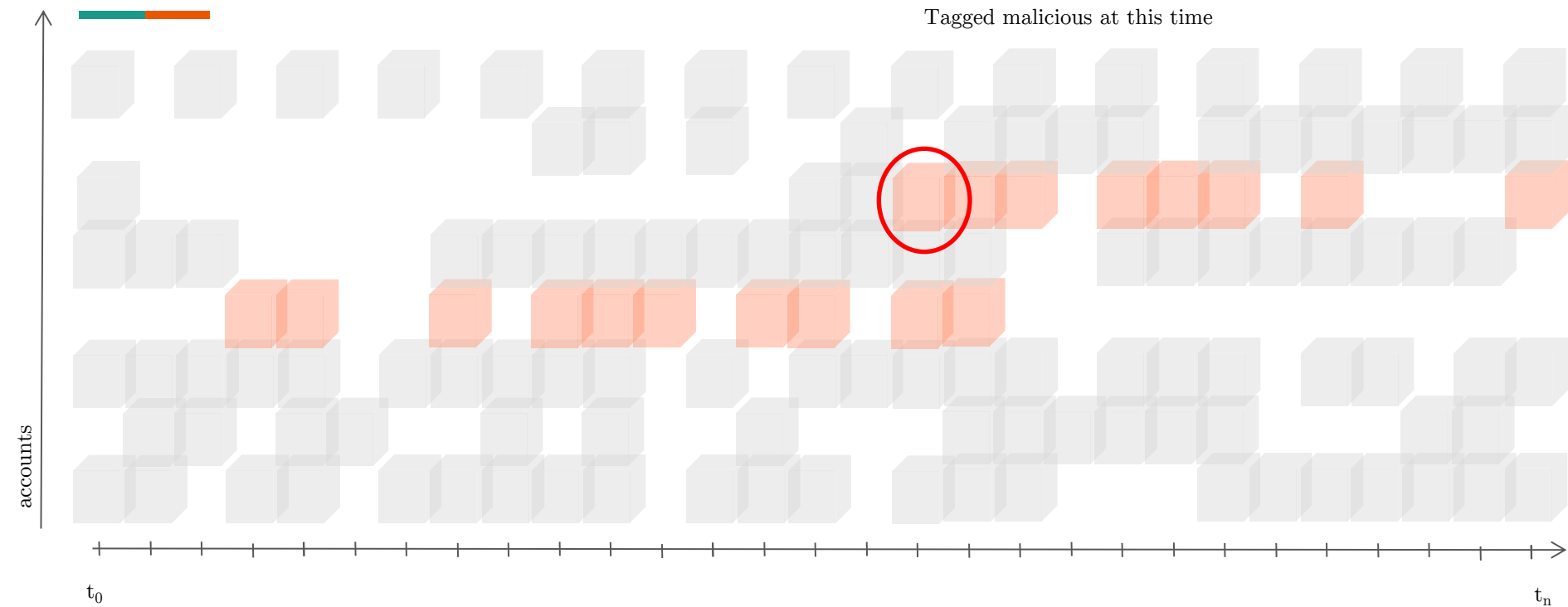


Results



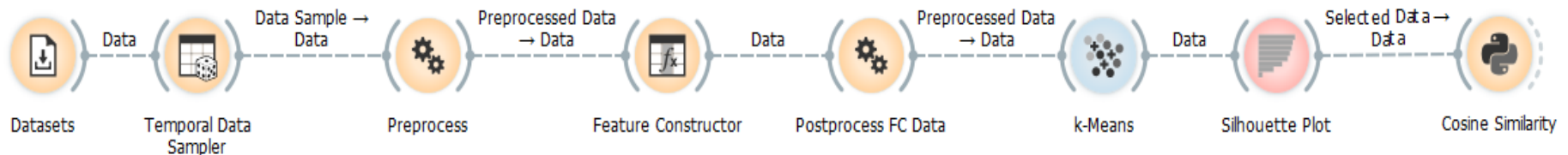
- We have found 554 and 293 accounts in only EOAs and All accounts settings respectively. 160 accounts are common in both of them.
- Most of these accounts
 - Have not transacted since a very long time.
 - Have *only incoming transactions* with less stable neighbourhood.

Representation of accounts wrt time



Behavioural Analysis

- Segment data into Sub-Datasets(SD) of different granularity (Tg).
- 1531 Day (6000 blocks) datasets, 219 Weeks (7 Days) datasets, 52 Months (30 Days) datasets, 18 Quarter (3 Months) datasets, 9 Half yearly (6 Months) datasets, 5 Yearly (365 Days) datasets, and finally Full dataset.
- Preprocessing, Feature construction and post-processing steps again performed for each granularity.
- Apply K-Means with previously identified hyperparameters on each 1800+ datasets, identify cluster with most malicious accounts, find cosine similarity between benign and malicious accounts in best identified cluster.



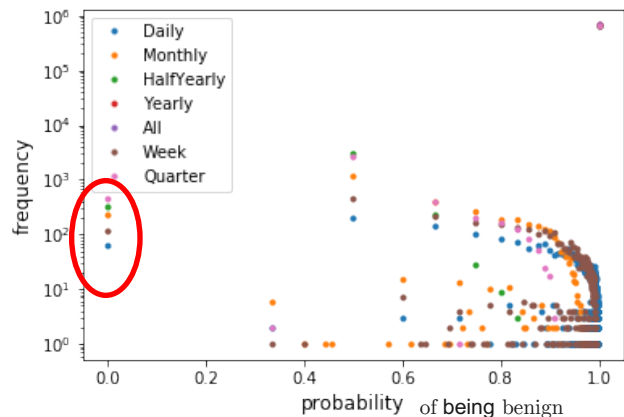
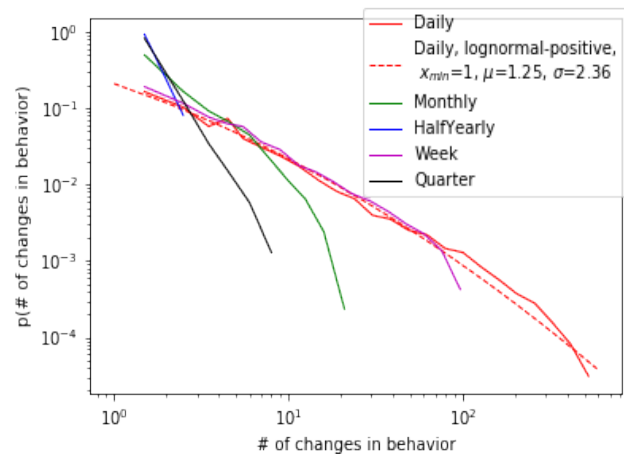
Behavioural Analysis

- Find number of time an account changed its behaviour (Malicious \leftrightarrow Benign).
- 9254 unique benign accounts showed unstable behavior.
- Probability of an account being malicious in a given Tg is

$$p_m^i = \frac{\sum_{j \in SD(T_g)^i} M_j}{n_i}$$

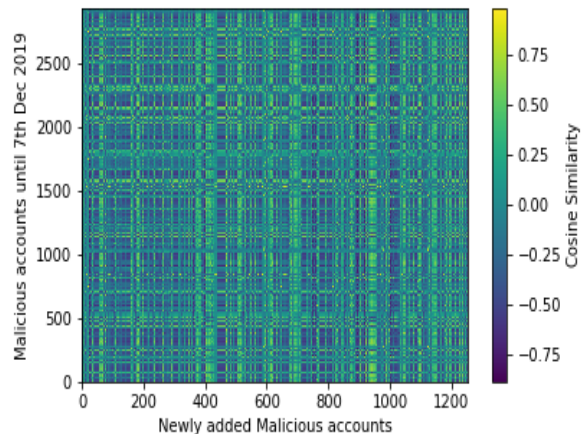
where n_i is the number of times in a particular Tg the account i appears, M_j is label given by our analysis to the account i in j^{th} SD

- For 814 unique accounts probability of being malicious was very close to 1.



Validation

- We notice new 1252 malicious added until 25th May 2020.
- These account have different behavior from previously marked accounts
- We applied ExtraTreesClassifier with identified hyperparameters, but, it failed to detect newly tagged malicious accounts as their behaviour is different.
- However if 10% of the new malicious accounts are added to training set, it successfully detects all other 90% of them.
- We do not apply unsupervised learning as we do not have sufficient data for them.
- Most accounts are newly created



Conclusion+Future Work



- We present a way to detect malicious accounts considering the temporal nature of the blockchains.
- We have introduced graph-based temporal features (such as **Burst** and **Attractiveness**) that are inspired by the existing attacks in the blockchain.
- We performed behavioural analysis considering temporal graphs rather than only using aggregated graph. Through which we also identified potential malicious accounts.
- In Future, we can also try Reinforcement Learning so that our model can continuously learn and modify itself learning from newly tagged malicious accounts.

Thank You

