Analysis, detection & mitigation of felonious wallet accounts over the Ethereum blockchain network using Machine learning techniques

DISSERTATION

Submitted in partial fulfillment of the requirements of the

MTech Data Science and Engineering Degree programme

By

Anjali Sunder Naik
2019HC04178

Under the supervision of

Dr. Vishnu Prasad V J
Senior Technical Architect
Adjunct Faculty, Department of Engineering Design, IIT Madras

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA

February 2022

Acknowledgement

I express my sincere gratitude to **Dr. Vishnu Prasad V J**, Senior Technical Architect Adjunct Faculty, Department of Engineering Design, IIT Madras for the extended faith in me and for his valuable inputs.

I also express my heartfelt thanks to **Bosch Global Software Technologies**, Bengaluru for providing the perfect nurturing environment to learn and excel in the market relevant technologies.

I am also grateful to the esteemed faculty of **Birla Institute of Technology** for their consistent effort in bringing out the best in their students.

Finally, I would like to thank my **parents, friends and colleagues** who helped me conquer the dual role of an employee and a student.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled "**Analysis, detection & mitigation of felonious wallet accounts over the Ethereum blockchain network using machine learning techniques"** and submitted by Ms. **Anjali Sunder Naik** ID. No. **2019HC04178** in partial fulfillment of the requirements of DSECLZG628T Dissertation, embodies the work done by her under my supervision.

Vishnu Prasad

Digitally signed by Vishnu Prasad
Date: 2022.02.27 23:13:01 +05'30'

(Signature of the Supervisor)

Place: Bengaluru

Date: February,2022

Dr. Vishnu Prasad V J

Senior Technical Architect

Adjunct Faculty, Department of Engineering Design, IIT Madras

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
SECOND SEMESTER 2020-21

DSECLZG628T **DISSERTATION**

Dissertation Title : Analysis, detection & mitigation of felonious wallet accounts over the Ethereum blockchain network using machine learning techniques

Name of Supervisor : Dr. Vishnu Prasad V J

Name of Student : Anjali Sunder Naik

ID No. of Student : 2019HC04178

Abstract

As of 2021, a survey from Coin Market Cap indicates that there are nearly over 6,000 digital coins in the market, a severe increase from just a handful since 2013. However, a large portion of these cryptocurrencies might not be that significant. The total market cap of all the crypto assets, including stable coins and tokens has shown a significant rise from year 2020 and has hit 2.4 trillion. Cryptocurrencies has vast potential of revolutionizing and transforming compliance-free peer-to-peer transactions. However, an end user must overcome certain challenges related to privacy, security, and control. As the transactions are recorded in a publicly distributed ledger known as blockchain, hackers have a large attack surface to gain access to critical and sensitive data. In the rapidly growing crypto currency space, the technological advent of cryptocurrencies and their respective benefits has been veiled with several illicit financing activities operating over the network such as ransomware, terrorist financing, hacking, data manipulation during transaction process, phishing, fraud, money laundering, bribery etc. Chainalysis, a firm that tracks every crypto currency transaction and serves as an advisor to an array of government authorities has published a report that shows that the amount of cryptocurrency spent on dark net markets rose 60% to reach a new high of $1.15billion from July 2020 to June 2021.

In this work, the primary focus is on the Ethereum network, which has seen over 1373 billion transactions since its inception. Propelled with the rise in use of machine learning techniques in the research dimensions of financial domain, this is an attempt to explore the possibility to use various machine learning algorithms to analyze and detect the illicit accounts using the transaction history. Many criminals don't typically transfer funds directly to and from their linked addresses when transacting with regulated exchanges. A vast majority of bad actors will move their funds at least one time. CipherTrace analysts found that a typical cryptocurrency exchange's dark market exposure will typically double at two hops out (transactions once removed from the exchange).

Various machine learning algorithms are evaluated on publicly available accounts flagged by the Ethereum community for their illegal activity coupled with valid accounts. A smart contract deployed on the public blockchain network is further used to track the illicit accounts, and hence proposed as a possible mitigation technique for flagging suspicious wallet addresses. External parties can query this smart contract to validate a blacklisted account and enable the law enforcement agencies take appropriate actions on the stolen coins/Ponzi schemes. The proof of truth data on the blockchain ledger will serve as a benchmark for future analysis.

Key Words: Blockchain, Big data, Fraud-detection, Ethereum, Machine-learning

Vishnu Prasad

Digitally signed by Vishnu Prasad
Date: 2022.02.27 23:13:39 +05'30'

(Signature of the Student) (Signature of the Supervisor)

Anjali Sunder Naik

2019HC04178

Dr Vishnu Prasad V J

Senior Technical Architect, Adjunct Faculty

Department of Engineering Design, IIT Madras

List of Symbols & Abbreviations used

API	Application Programming Interface
CRISP-DM	Cross Industry Standard Process for Data Mining
DApp	Decentralized application
EOA	Externally Owned Accounts
ETH	Ether is the cryptocurrency of the Ethereum network
FN	False Negative
FP	False Positive
KNN	K Nearest Neighbor
SC	Smart Contract
TN	True Negative
TP	True Positive
TPR	True Positive Rate
UI	User Interface

List of Tables

List of Figures

Table of Contents

Introduction

The world wide web has revolutionized information, and the Web2 has revolutionized interactions. The Web3, widely known as the next age of the internet is an idea for a new iteration of the world wide web based on blockchains. The current internet system with its client-server-based architecture and centralized data management has many unique points of failure in terms of privacy of personal data and inefficiencies in the backend operations. Blockchain came into existence with a need to resolve the existing trust issues in the data-intensive transaction systems. [5]

In general terms, a blockchain is an immutable distributed ledger, maintained within a distributed network of nodes, wherein each node maintains a copy of the ledger by applying transactions, validated by a consensus protocol. Blockchain technologies has a very interesting evolution cycle and has come quite far from cryptographically secured chain of blocks to decentralized applications.

In the early 90s, Stuart Haber and Scott Stornetta published their first work on blockchain, which served as a basis for the 2009 Satoshi Nakamoto's popular Bitcoin[2] Whitepaper. The first bitcoin purchase took place in 2010, which further grew into a $1 billion marketplace in 2013. At the same time, Vitalik Buterin released the Ethereum[3] whitepaper. The Ethereum genesis block was further created in 2015, which has grown to house 1.278M transactions at present.

Ethereum is a blockchain framework which lets a person send cryptocurrency to anyone for a minimal fee. It also powers applications that can be consumed across all the participants in the network. It can be simply termed as the "world's programmable blockchain". The introduction of Ethereum as a blockchain platform opened a new world to investors, developers, researchers, bankers, money launderers and hackers. The pseudo-anonymity nature of the actors in the network has paved way to felonious activities without a trace. Hence this work attempts to provide a solution to the growing need of detecting felonious activities in the Ethereum network.

Problem Definition

As of 2021, a survey from Coin Market Cap indicates that there are nearly over 6,000 digital coins in the market, a severe increase from just a handful since 2013. However, a large portion of these cryptocurrencies might not be that significant. The total market cap of all the crypto assets, including stable coins and tokens has shown a significant rise from year 2020 and has hit 2.4 trillion. Cryptocurrencies has vast potential of revolutionizing and transforming compliance-free peer-to-peer transactions. However, an end user must overcome certain challenges[1][4] related to privacy, security, and control. As the transactions are recorded in a publicly distributed ledger known as blockchain, hackers have a large attack surface to gain access to critical and sensitive data. In this rapidly growing crypto currency space, the technological advent of cryptocurrencies and their respective benefits has been veiled with several illicit financing activities operating over the network such as ransomware, terrorist financing, hacking, data manipulation during transaction process, phishing, fraud, money laundering, bribery etc. Chainalysis, a firm that tracks every crypto currency transaction and serves as an advisor to an array of government authorities has published a report that shows that the amount of cryptocurrency spent on dark net markets rose 60% to reach a new high of $1.15 billion from July 2020 to June 2021.

Objective of the project

In this work, the primary focus is on the Ethereum[7] network, which has seen over 1373 million transactions since its inception. Propelled with the rise in use of machine learning techniques in the research dimensions of financial domain, this is an attempt to explore the possibility to use various machine learning algorithms to analyze and detect the felonious accounts using the transaction history. Many criminals don't typically transfer funds directly to and from their linked addresses when transacting with regulated exchanges. A vast majority of bad actors will move their funds at least one time. CipherTrace analysts found that a typical cryptocurrency exchange's dark market exposure will typically double at two hops out (transactions once removed from the exchange).

Various machine learning algorithms are evaluated on publicly available accounts flagged by the Ethereum[7] community for their illegal activity coupled with valid accounts.

Uniqueness of the project

This project will provide a compact comparison of various machine learning techniques to identify fraudulent activities in the Ethereum[7] network.

Benefit to the organization

The analysis would benefit the organization in below ways:

- There would be more trust and worldwide adoption of the distributed ledger technology and possible regulation can be achieved in its usage.
- The felonious wallet accounts can be tracked and validated which would internally enable the law enforcement agencies take appropriate actions on the fraudulent activities
- The analysis data would serve as a benchmark for future analysis.

Background

Ethereum is a blockchain platform with a built-in fully-fledged Turing-complete programming language that can be used to create “smart contracts”, driven by an internal crypto-fuel called as Ether (ETH). These contracts can be used to encode arbitrary state transition functions and allows users create systems by writing up the logic in a few lines of code. This concept provides for a platform with unique potential, intended for a wide array of applications in finance, data storage systems, identity, and reputation systems.

History

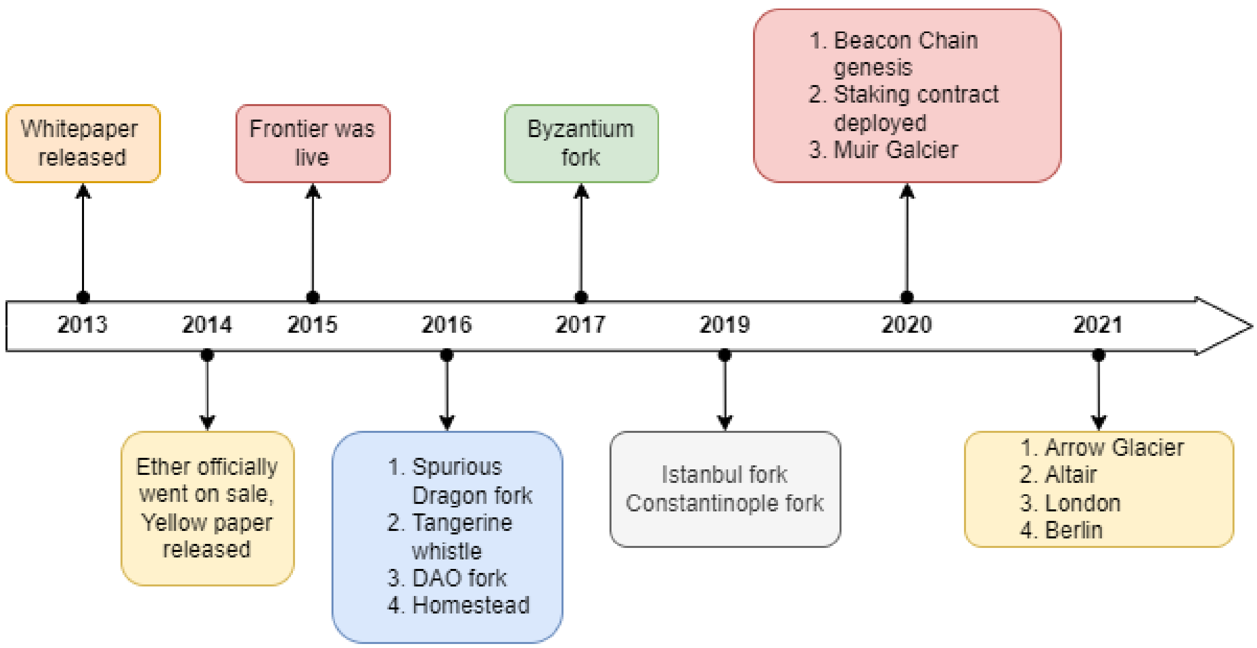

Figure 1: Ethereum timeline

Accounts in Ethereum

Ethereum accounts are 20-byte addresses which contains four fields:

- Nonce
- Ether Balance
- Contract code
- Storage

There are 2 types of accounts:

- Externally Owned Accounts (EOA)
- Smart Contracts (SC)

Table 1: Difference between EOA & SC

EOA	SC
Controlled by Private keys	Controlled by Contract code
Has no code	Code is activated when poked by a message or transaction
Ability to send transactions	Ability to send messages

Ethereum Transactions & Messages

A transaction in the Ethereum is referred to the signed data package that stores a message to be sent from an EOA. Messages are non-serialized virtual objects existing in Ethereum execution environment.

A transaction contains the following:

- Recipient
- Sender's signature
- Ether to be transferred to recipient
- Data (optional)
- Maximum number of computational steps the transaction can take, also known as STARTGAS
- The sender's fee, also known as GASPRICE

A message contains the following:

- Sender
- Recipient
- Ether to be transferred to recipient
- Data (optional)
- STARTGAS

Maximum number of computational steps the transaction can take, also known as STARTGAS

Felonies in Ethereum

The major challenge faced by the Ethereum network are the felonious activities occurring in the hindsight, due to its anonymous nature. The scams and frauds have imposed very high costs to the financial system. Some of the major felonies in the Ethereum transactions, as extracted from the Etherscan[6] label cloud are as below.

Table 2: Sample Felonies in Ethereum Network

LABEL	NUMBER OF ACCOUNTS
PHISH/ HACK	5172
SPAM TOKEN	10
BITPOINT HACK	2
CRYPTOPIA HACK	6
ETHERDELTA HACK	1
UPBIT HACK	815

Solution Overview

We use a mix of CRISP_DM and Big Data Lifecycle framework to solve the current problem statement. CRISP-DM has 6 high level phases, and it was used in IBM SPSS Modeler tool. It is an iterative approach to the development of analytical models.

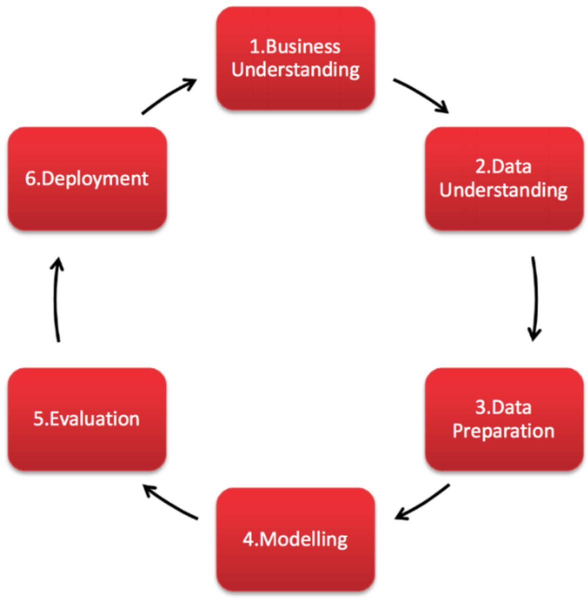

Figure 2: Phases in CRISP DM Methodology

Business Understanding

The primary objective of this project is to detect or predict felonious activities within the Ethereum network through Knowledge Discovery, via the unusual patterns derived from the gathered data. The insights gathered would reflect in the form of increased trust in decentralized applications and benefit the larger crowd remove the middleman in any transaction related use-case.

The scope of work would be as following:

- Collection of blacklisted Ethereum wallet accounts data and its relevant historical transactions.
- Cleansing of data and identification of relevant attributes
- Data preprocessing & raw feature extraction
- Identification of Machine Learning algorithms to be applied on the data set
- Data Analysis using various algorithms and evaluation of key metrics
- Hyperparameter tuning to identify ideal model algorithm
- Visualizations of results

Table 3: Resources utilized for the project

Programming Language	Python, Java, Solidity[9]
Tools	Eclipse, Jupyter-Lab

Hardware+ Cloud Platform	Google Collab+ GPU

Data Understanding

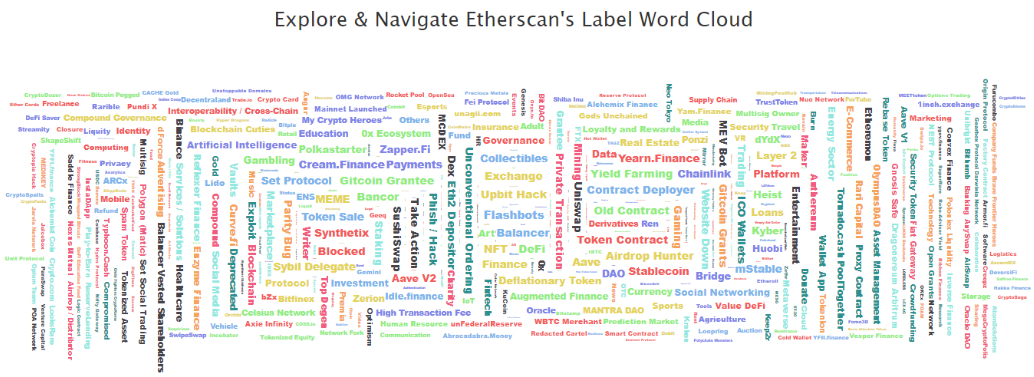

Figure 3: Ether Scan Word Cloud

We collect the accounts & contracts from various open sources such as Ethereum foundation backlisted data, Ether scan, which is a block explorer and analytics platform for Ethereum. The Ether scan provides segregated data based on word cloud, and we filter the felonious data based on key words such as phishing, hack, etc. We also collect data from Harvard verse and GitHub.[1]

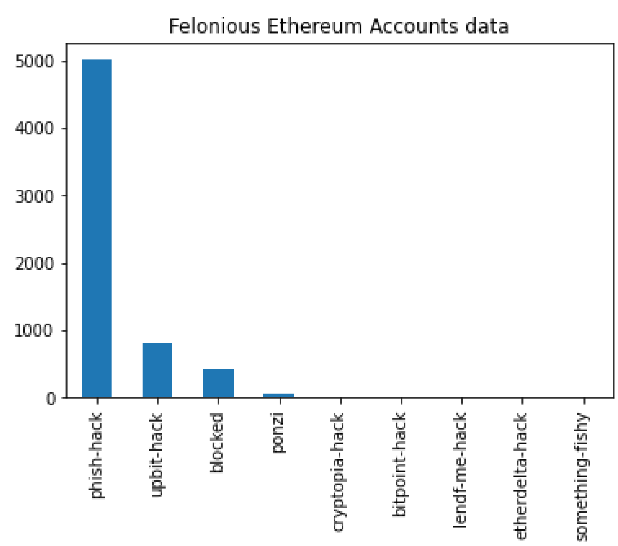

Figure 4: Data Visualization for Felonious Ethereum Accounts

We collect a total of **8476** felonious accounts and a total of **8843** Non-felonious accounts. The collected data is considered to satisfy the requirements of the problem statement.

Etherscan[6] Ethereum Developer[8] APIs provide direct access to Etherscan's block explorer data and services via GET/POST requests. We will use the accounts API to retrieve the list of transactions for the collected Ethereum addresses.

```
https://api.etherscan.io/api
   ?module=account
   &action=txlist
   &address=ETHEREUM-ADDRESS
   &sort=asc
   &apikey=API-KEY-TOKEN
```

The sample response for the API request looks like below:


```
{
    "status": "1",
    "message": "OK",
    "result": [
        {
            "blockNumber": "6127109",
            "timeStamp": "1533974291",
            "hash": "0x1890d018b54fc773ca153701f64b0668d278e15ee9f99abad11635d24ec0babe",
            "nonce": "0",
            "blockHash": "0xa4a5635e484879021678290a785d8ef245c959d6f1613e9ec0f94ce13c088c8c",
            "transactionIndex": "105",
            "from": "0x8ddfdf60aaffe05c623ba193a186abd1f8024946",
            "to": "0xbceaa0040764009fdcff407e82ad1f06465fd2c4",
            "value": "2553361432875840108146O",
            "gas": "21000",
            "gasPrice": "9000000000",
            "isError": "0",
            "txreceipt_status": "1",
            "input": "0x",
            "contractAddress": "",
            "cumulativeGasUsed": "7124989",
            "gasUsed": "21000",
            "confirmations": "7863508"
        },
```


Figure 5: Sample Transactions from the API

Data Preparation

Data Cleansing

In the first phase of data cleansing, we filter the Ethereum addresses with null transaction. We also convert the address to lower case and remove the duplicated records. Once that is achieved, we segregate the accounts as:

- Externally Owned Address (EOA)
- Smart Contract (SC)

The result of this process is labelled data for both the categories of the Ethereum accounts.

Feature Engineering

We identify a total of **44** features from the EOA and **18** features from the SC. We can treat Ethereum transaction data as big data, and hence map reduce is the optimal programming model to efficiently extract the features in parallel over the large dataset in a distributed manner.

MapReduce is central to Apache Hadoop and distributed data processing. By leveraging data locality, MapReduce allows functions to run in parallel across multiple server nodes in a cluster. We use this feature of MapReduce algorithm to efficiently pre-process the input data. We create two pipelines for Felonious as well as non-felonious label.

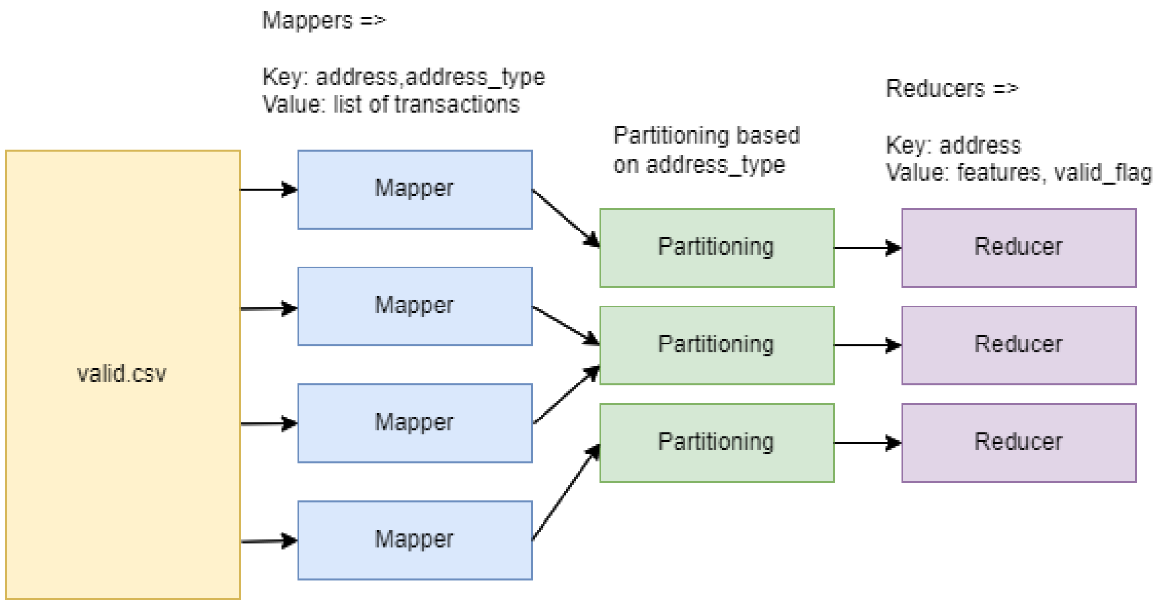

Figure 6: Map Reduce Pipeline for Felonious Data

```
Map-Reduce Framework
        Map input records=8477
        Map output records=6924
        Map output bytes=3118225
        Map output materialized bytes=3144968
        Input split bytes=139
        Combine input records=0
        Combine output records=0
        Reduce input groups=5678
        Reduce shuffle bytes=3144968
        Reduce input records=6924
        Reduce output records=5678
        Spilled Records=13848
        Shuffled Maps =3
        Failed Shuffles=0
        Merged Map outputs=3
        GC time elapsed (ms)=317
        Total committed heap usage (bytes)=4127195136
```

Figure 7: Map Reduce Framework for Felonious Data

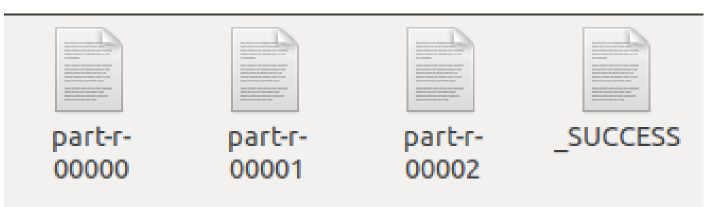

Figure 8: Snapshot of Map-reduce execution

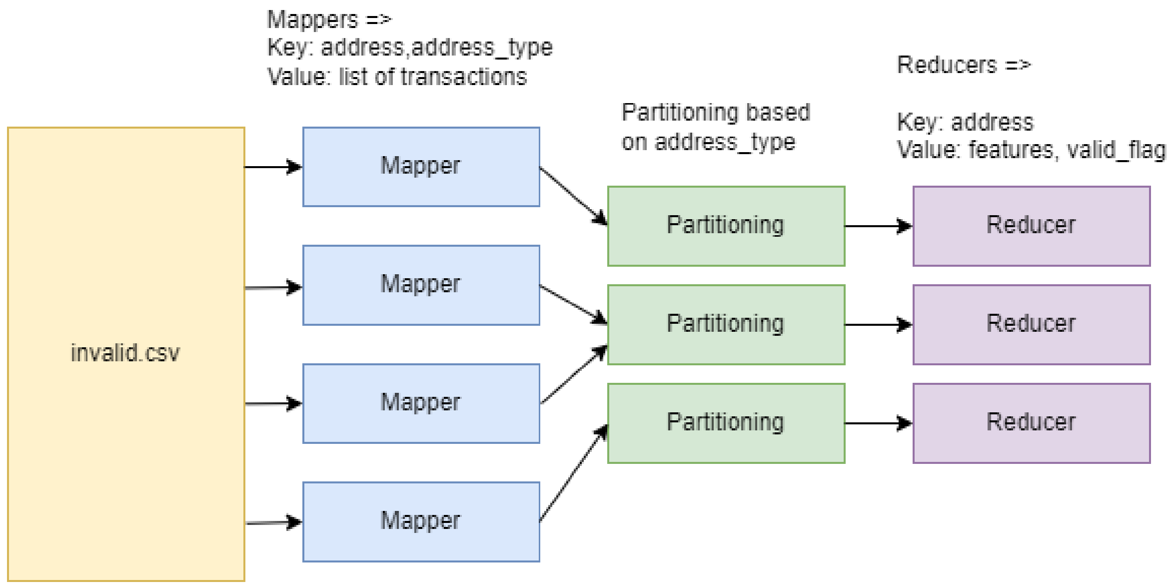

Figure 9: Map Reduce Pipeline for Non-Felonious Data

The **part-r-00000** file consists of all the EOA along with its features, while the **part-r-00001** contains all the Smart contract addresses along with its features. The **part-r-00002** file contains the accounts with null transactions and hence we avoid them in the further analysis.

Once the features are extracted for both EOA & SC, we use information gain as a parameter to obtain the top 10 features relevant for the data analysis. Information gain is a feature selection mechanism, which evaluates the gain of each feature variable against the context of the target variable.

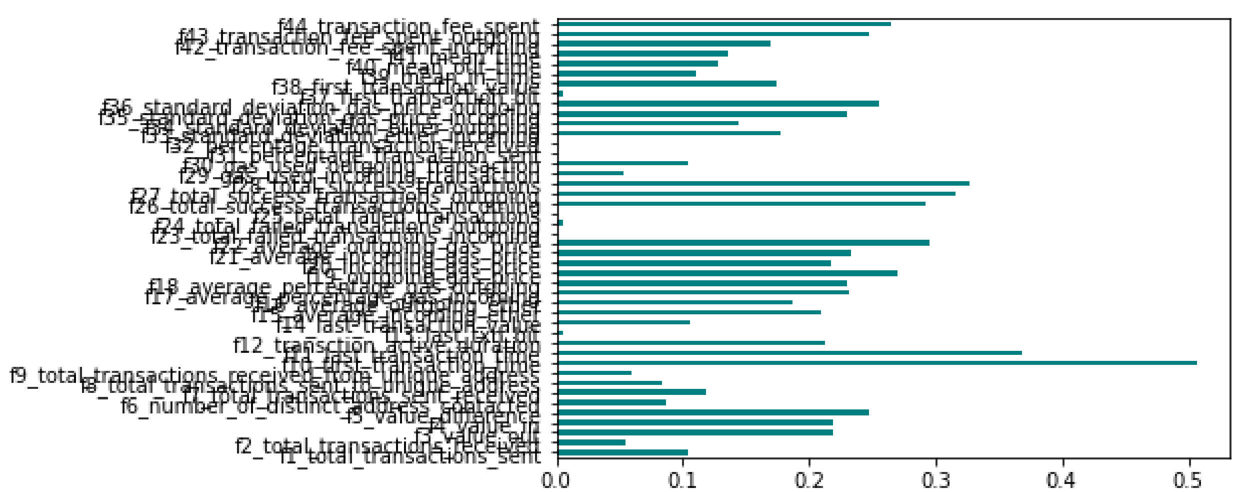

Figure 10: Information gain for EOA features

The top 10 EOA features are identified as follows:

```
f10_first_transaction_time 0.5078283385578866
f11_last_transaction_time 0.3689468220033978
f28_total_success_transactions 0.32750239860581076
f27_total_success_transactions_outgoing 0.3166087114396814
```

```
f22_average_outgoing_gas_price 0.29469023216528467
f26_total_success_transactions_incoming 0.29200763898237847
f19_outgoing_gas_price 0.2691566134985095
f44_transaction_fee_spent 0.26463002109776035
f36_standard_deviation_gas_price_outgoing 0.25541472205622684
f5_value_difference 0.24711574539412395
```

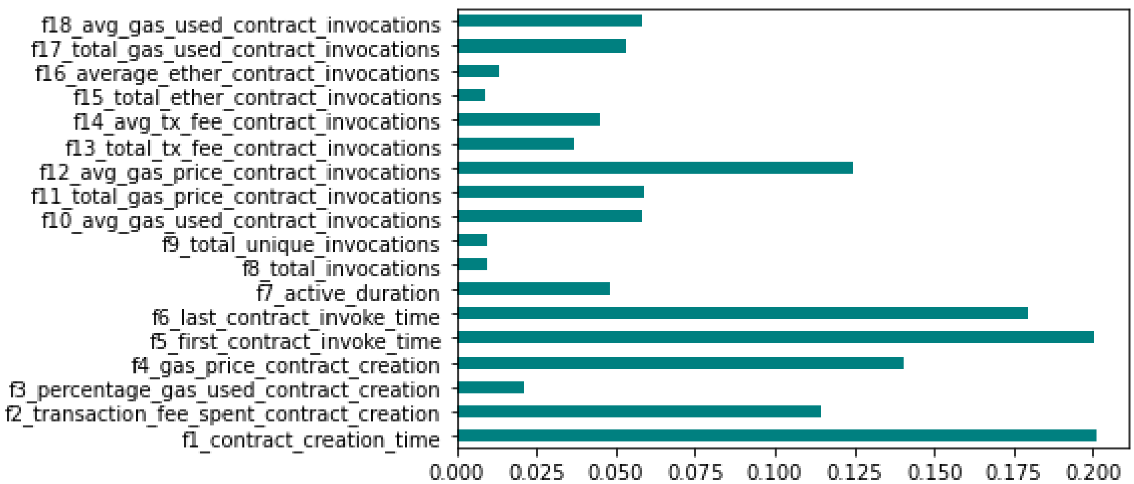

Figure 11: Information gain for SC features

The top 10 SC features are identified as follows:

```
f1_contract_creation_time 0.2012582898157813
f5_first_contract_invoke_time 0.20043490336190617
f6_last_contract_invoke_time 0.179211553031543
f4_gas_price_contract_creation 0.14047210049203573
f12_avg_gas_price_contract_invocations 0.12440789872541136
f2_transaction_fee_spent_contract_creation 0.11480564305480434
f11_total_gas_price_contract_invocations 0.05919944770421259
f18_avg_gas_used_contract_invocations 0.05859698304821159
f10_avg_gas_used_contract_invocations 0.058288528292604
f17_total_gas_used_contract_invocations 0.05342361341653068
```

Modeling

We use the below machine learning classifiers on our pre-processed data set.

1. Decision Tree Classifier
2. K-NN Classifier
3. Random Forest Classifier
4. XGBoost Classifier

Decision Tree Classifier

The decision tree algorithm can be visualized as a tree structure, which includes a root node, branches and leaf nodes. Each node represents a feature, the link between the nodes represents a decision and each leaf node holds a class label. The termination of the classifier is dependent on the attributes of the classifier.

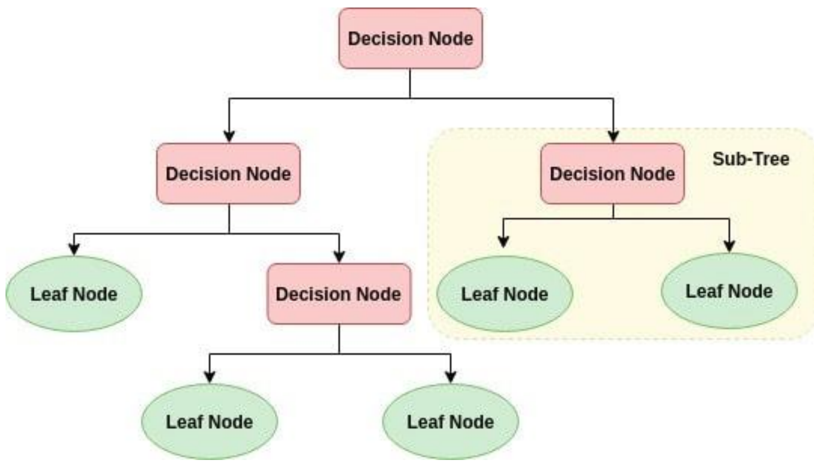

Figure 12: Decision Tree Algorithm

K Nearest Neighbor algorithm

K Nearest Neighbor algorithm belongs to the class of Supervised Learning and is used for classification and regression. It considers K nearest neighbors to predict the class labels.

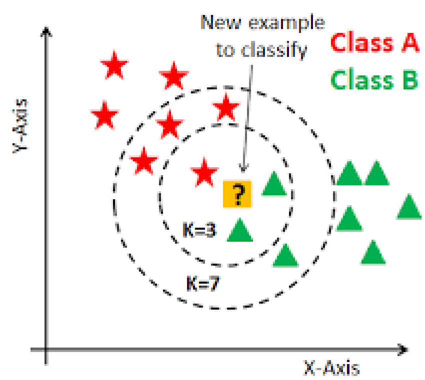

Figure 13: KNN Algorithm

Random Forest Classifier

Random Forest Classifier belongs to ensemble-based learning methods. The approach comprises of construction of many “simple” decision trees in the training stage and majority vote across classification stage. In the training stage, random forest algorithm applies bagging technique to individual trees in the ensemble.

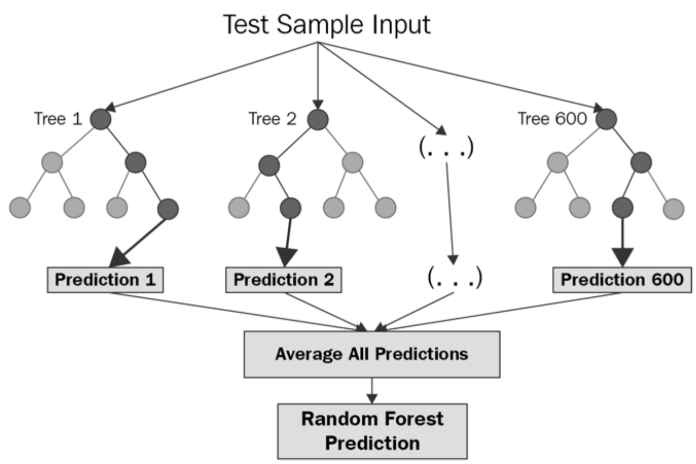

Figure 14: Random Forest Classifier

XG Boost Classifier

XG Boost is an optimized distributed gradient boosting library. It belongs to a family of boosting algorithms and uses boosting framework at its core.

Evaluation

Evaluation Parameters

Confusion Matrix

For any classification model prediction, a confusion matrix can be constructed. It demonstrates the number of test cases classified correctly and incorrectly and is used to describe the performance of the classifier.

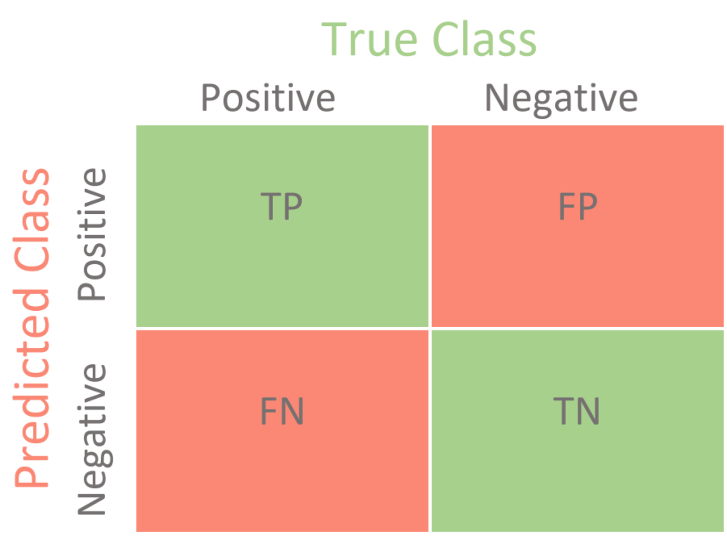

Figure 15: Confusion Matrix for Binary Classification

Metrics of the Confusion Matrix are as following:

- **True Positive (TP)**
  It is the number of predictions where the classifier has correctly predicted the positive class as positive.

- **True Negative (TN)**
  It is the number of predictions where the classifier has correctly predicted the negative class as negative

- **False Positive (FP)**
  It is the number of predictions where the classifier has incorrectly predicted the negative classes as positive

- **False Negative (FN)**
  It Is the number of predictions where the classifier has incorrectly predicted the positive classes as negative

Accuracy

It is the fraction of total samples that were correctly classified by the classifier

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

It is the fraction of predictions indicating positive classes were positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall

It is the fraction of all positive samples that were correctly predicted positive by the classifier. It is also known as True Positive Rate (TPR), Sensitivity, Probability of Detection.

$$Recall = \frac{TP}{TP + FN}$$

F1 Score

It is the harmonic mean of precision and recall.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

AUC-ROC

ROC curve is a plot of TPR against False positive rate. AUC-ROC stands for Area Under the Receiver Operating Characteristics and the higher the area, the better the model performance.

EOA Analysis

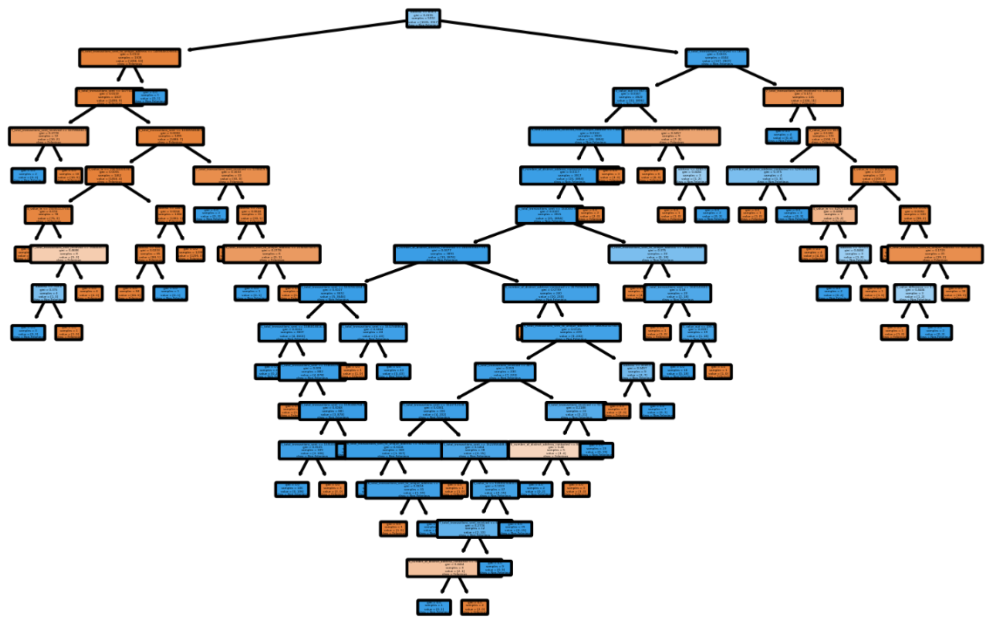

Figure 16: EOA Analysis: Decision Tree

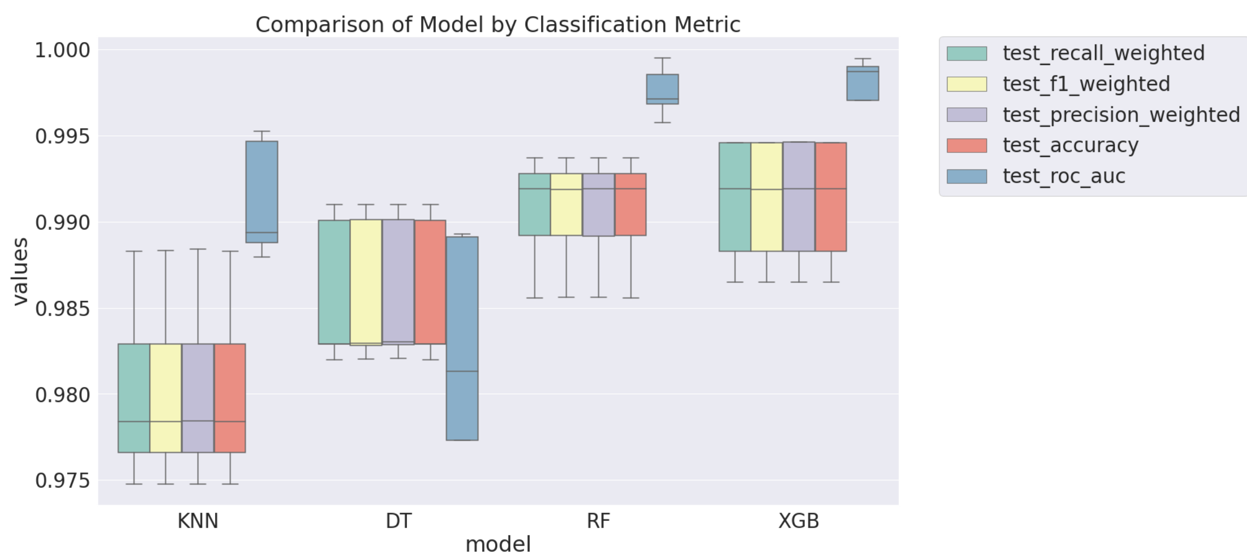

Figure 17: Comparison of models for EOA Analysis by Classification Metric

From the above comparison it is immediately clear that KNN fits our data rather poorly compared to other models, while ensemble decision tree models (Random Forest and XGBoost) fit the data very well.

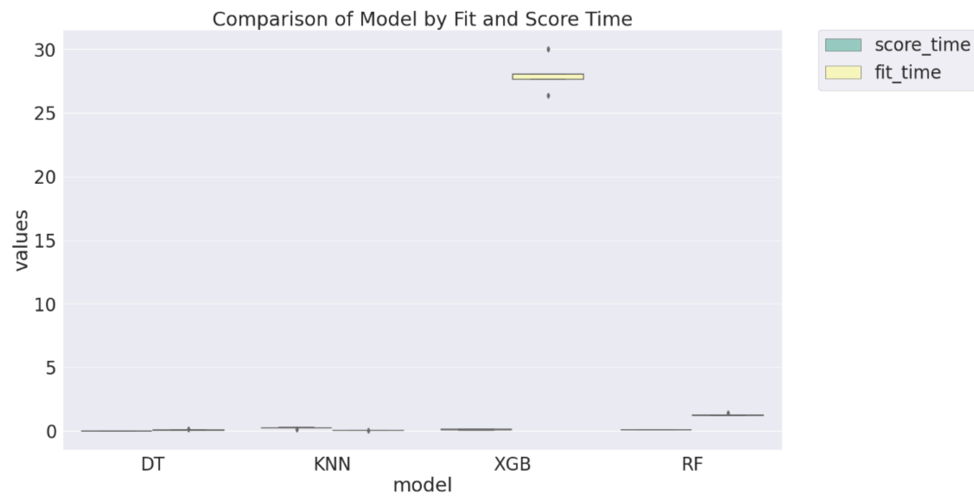

Figure 18: Comparison of Models for EOA Analysis by Fit and Score time

It is interesting to note that XGBoost was far and slowest to train; however, it was the best performing. Random Forest, while relatively slow compared to KNN, Decision tree, had the second-best performance.

SC Analysis

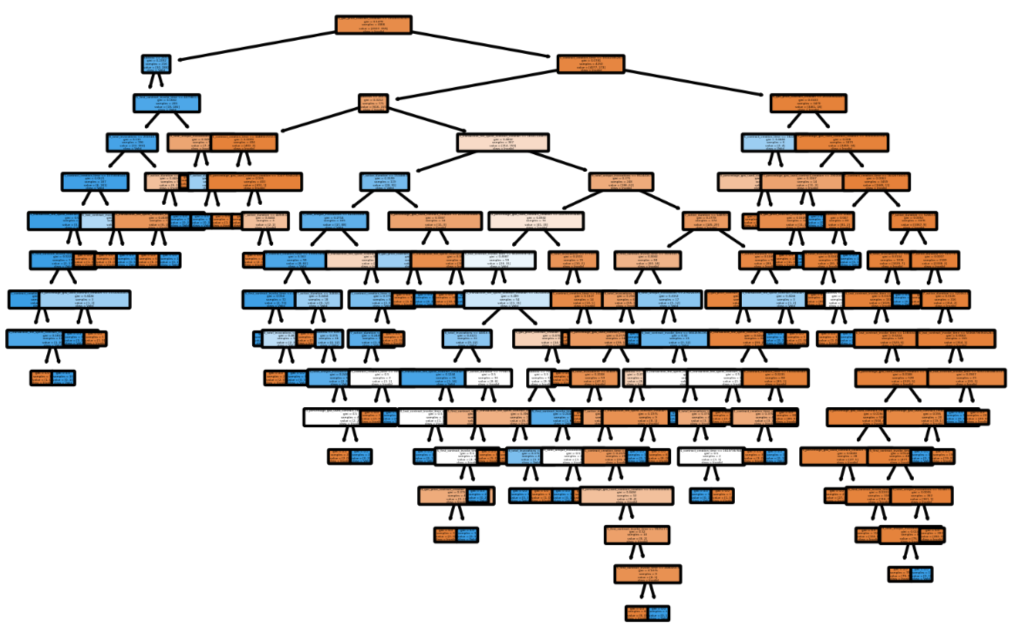

Figure 19: SC Analysis: Decision Tree

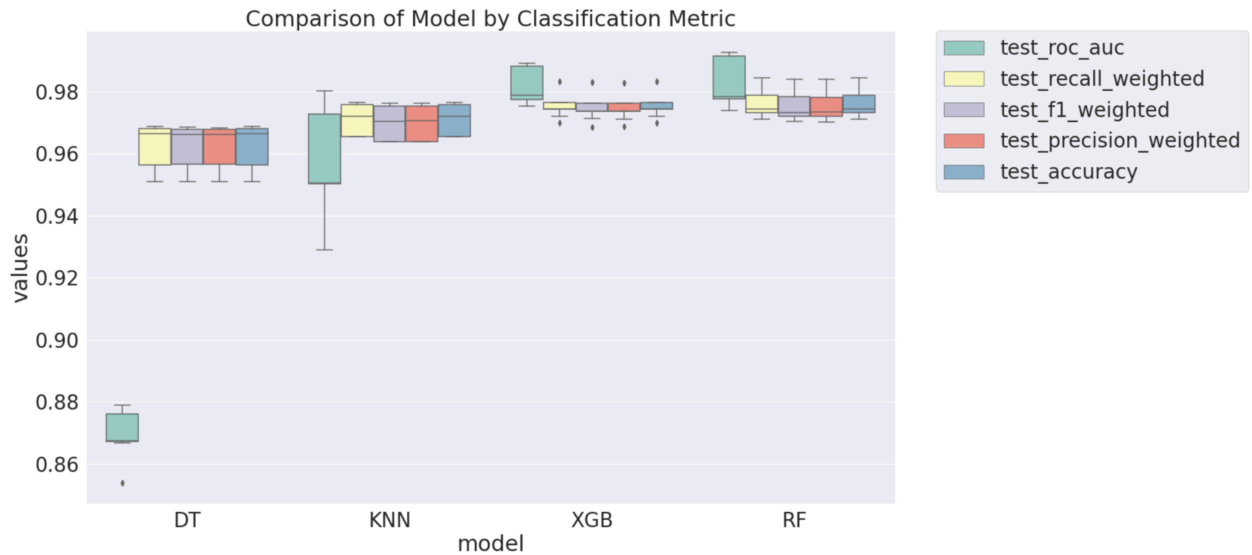

Figure 20: Comparison of models for SC Analysis by Classification Metric

From the above comparison it is immediately clear that DT fits our data rather poorly compared to other models, while ensemble decision tree models (Random Forest and XGBoost) and KNN fit the data very well.

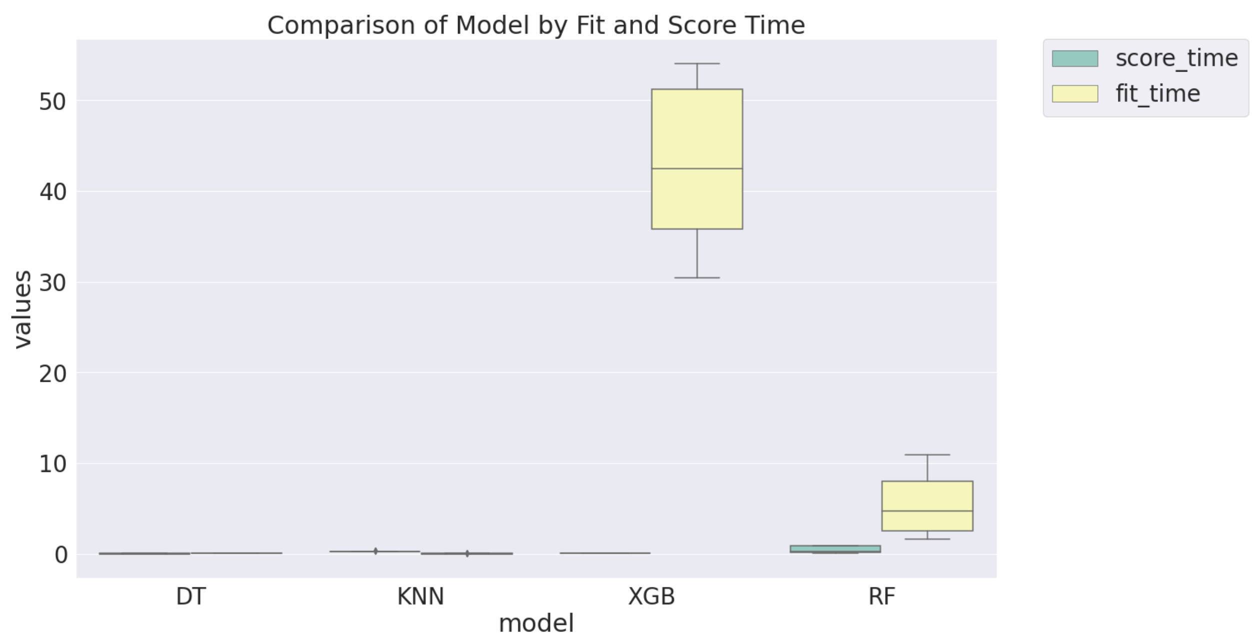

Figure 21: Comparison of Models for SC Analysis by Fit and Score time

A similar observation can be noted for Smart Contract analysis as well, XGBoost was far and slowest to train; however, it was the best performing. Random Forest, while relatively slow compared to KNN, Decision tree, had the second-best performance.

Deployment

Mitigation of Felonious activities in Ethereum network

A simple solidity[9] contract can be deployed on the Ethereum network which would aid users recognize valid felonious accounts.

The smart contract has 2 major functions:

1. Store
   Store takes account address, account type and felonious flag as parameters, and stores the data into the ledger.
2. Retrieve
   Retrieve takes account address as function parameter and returns the status of the account from the ledger

Hence in this way, this blockchain DApp can enable a user to validate the malicious nature of an account by simply querying the ledger.

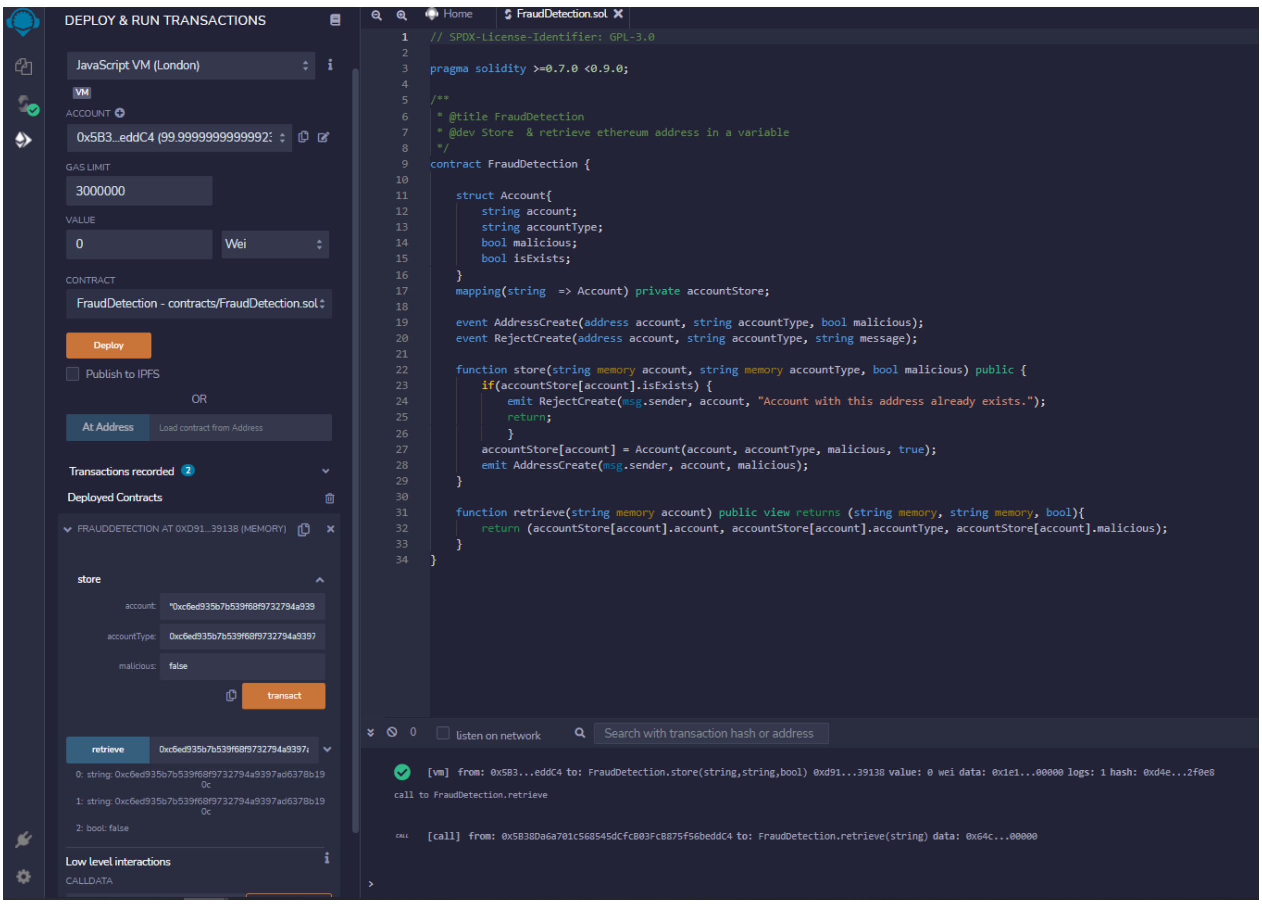

Figure 22: Smart Contract to mitigate felonious activities

User Interface

We deploy the model to a flask server and build a simple UI to classify the new accounts. The user interface takes in the Ethereum address, determines the category as Smart Contract/ Externally Owned Account. It further classifies the account as Felonious or Non-felonious.

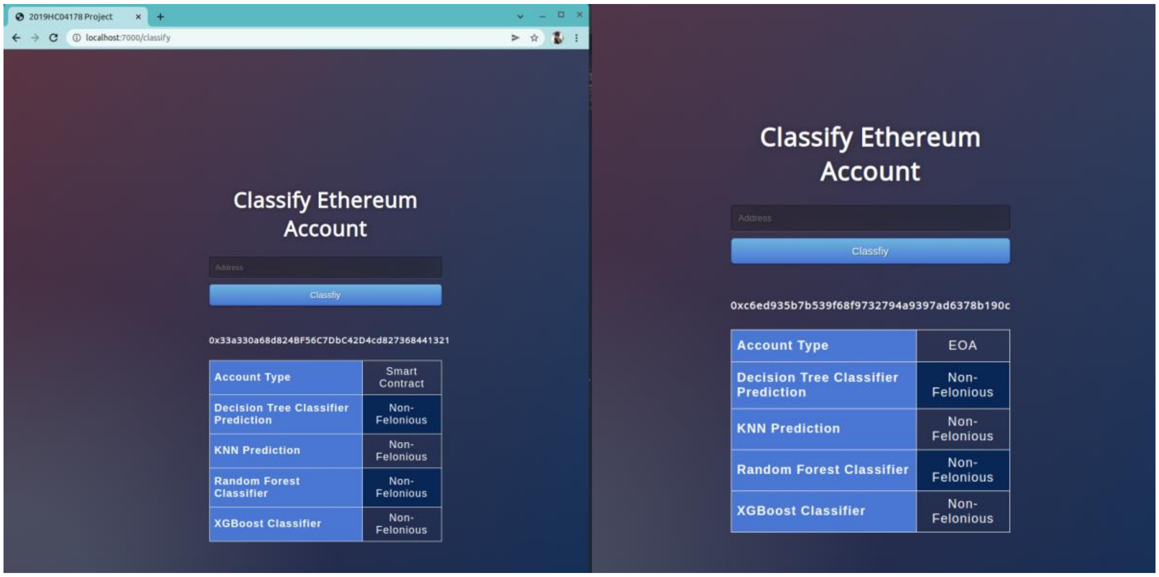

Figure 23: Flask UI to classify Ethereum accounts

Conclusions/ Recommendations

In this work, we introduced a methodology to extract features from the different accounts available in the Ethereum network. We explored the possibility of using MapReduce algorithm on the big data set to extract the relevant features. We identified the classes as “Felonious” and “Non-Felonious” and applied the classification algorithms such as Decision Tree, k-NN, Random forest and XG-Boost on 2 different categories of the Ethereum accounts. The machine learning algorithms were applied on the top 10 features, selected based on information gain. We finally attempt to identify the best performing model by comparing them using resampling methods like cross validation technique using scikit-learn[10] package of python. Model fit statistics such as accuracy, precision, recall etc. are calculated for comparison, and ROCs are used to analyze the best performing algorithm for the given dataset.

We identify that the XG Boost Classifier takes the most time to train, however has the best performance among other classifiers.

Finally, we propose a mitigation technique in the form of smart contract, to identify valid felonious accounts. We also deploy the models to classify any new account address using a flask server and user interface.

Directions for future work

Some of the key challenges faced during the implementation of the project are as follows:

- Map-reduce execution fails due to irregular data in the dataset.
- Accounts will null transactions/lack of account information, though relevant to use case is avoided in the analysis.
- Feature extraction becomes difficult for data which has less than 2 transactions

The above challenges can be accommodated for future work.

- We can also perform real time analytics with big data using Microsoft Azure Synapse Analytics Service
- The developed blockchain DApp is not integrated with the current architecture, hence we can integrate with the help of web3 APIs and ensure trust in the felonious account detection.
- We can repeat the work on a larger data set and validate the accuracy rate against the current feature set.

Bibliography/ References

[1] Shlomi Dolev, Vladimir Kolesnikov, Sachin Lodha, Gera Weiss (Eds.) Cyber Security Cryptography and Machine Learning, Fourth International Symposium, CSCML 2020 Be'er Sheva, Israel, July 2–3, 2020 Proceedings

[2] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).

[3] Vitalik Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. (2014)

[4] Farrugiaa, Ellul, Azzopardi.(2020). Detection of illicit accounts over the Ethereum blockchain. 0957-4174/2020 Elsevier Ltd.

[5] Rahmeh Fawaz Ibrahim, Aseel Mohammad Elian, Mohammed Ababneh. Illicit Account Detection in the Ethereum Blockchain Using Machine Learning. 2021 International Conference on Information Technology (ICIT)

[6] https://etherscan.io/

[7] https://ethereum.org/en/

[8] https://docs.etherscan.io/

[9] https://docs.soliditylang.org/en/v0.8.12/

[10] Jake VanderPlas, O'Reilly Media, Inc.,ISBN 9781491912058, Python Data Science Handbook

Appendices

Appendix A

Table 4: Data attributes and its definition

Key	Definition
blockNumber	The length of the blockchain in blocks
timeStamp	The time when the block was mined
hash	The cryptographic hash that represents the block header
nonce	A counter that indicates the number of transactions sent from the account.
blockHash	It is used to query the hash of the past 256 blocks
transactionIndex	Index of the transaction in the block
from	The address of the account that submitted the transaction.
to	The address of the recipient or smart contract that the transaction interacts with.
value	Amount of ETH to transfer from sender to recipient (wei)
gas	Measures the amount of computational effort required to execute specific operations
gasPrice	Cost necessary to perform a transaction on the network
isError	If *isError* is 0 then transaction was successful else failed.
txreceipt_status	0 when transaction if execution failed, 1 if succeeded.
input	Extra data attached to transaction
contractAddress	Given when a contract is deployed to the Ethereum Blockchain.
cumulativeGasUsed	The total amount of gas used when this transaction was executed in the block.
gasUsed	The total units of gas used by the transactions in the block
confirmations	Number of blocks created since the block that included your transaction.

Appendix B

Table 5: Features extracted from Externally owned accounts

	Feature	Description	Unit
1	$f1\_total\_transactions\_sent$	The total number of transactions sent from the given address	Integer
2	$f2\_total\_transactions\_received$	The total number of transactions received from the given address	Integer
3	$f3\_value\_out$	The total ether sent from the given address	Big Integer

4	$f4\_value\_in$	The total ether received from the given address	Big Integer
5	$f5\_value\_difference$	Absolute difference $f3_{value_{out}} - f4_{value_{in}}$	Big Integer
6	$f6\_number\_of\_distinct\_address\_contacted$	The number of distinct addresses contacted	Integer
7	$f7\_total\_transactions\_sent\_received$	The total number of transactions performed by the address	Integer
8	$f8\_total\_transactions\_sent\_to\_unique\_address$	The total number of transactions sent to a unique address	Integer
9	$f9\_total\_transactions\_received\_from\_unique\_address$	The total number of transactions received from a unique address	Integer
10	$f10\_first\_transaction\_time$	The block timestamp wherein the first transaction was performed	Long
11	$f11\_last\_transaction\_time$	The block timestamp wherein the last transaction was performed	Long
12	$f12\_transaction\_active\_duration\ (seconds)$	$\mathrm{f}11_{\mathrm{last}_{\mathrm{transaction}_{\mathrm{time}}}} - \mathrm{f}10_{first_{transaction_{time}}}$	Long
13	$f13\_last\_txn\_bit$	0 if last transaction is incoming transaction else 1	Integer
14	$f14\_last\_transaction\_value$	Total ether transferred in last transaction	Big Integer
15	$f15\_average\_incoming\_ether$	Average value of ether in the incoming transactions	Big Integer
16	$f16\_average\_outgoing\_ether$	Average value of ether in the outgoing transactions	Big Integer
17	$f17\_average\_percentage\_gas\_incoming$	Average % of gas used in the incoming transactions	Double

18	$f18\_average\_percentage\_gas\_outgoing$	Average % of gas used in the outgoing transactions	Dou ble
19	$f19\_outgoing\_gas\_price$	Total gas price in the outgoing transaction	Long
20	$f20\_incoming\_gas\_price$	Total gas price in the incoming transaction	Long
21	$f21\_average\_incoming\_gas\_price$	Average gas price in the outgoing transaction	Dou ble
22	$f22\_average\_outgoing\_gas\_price$	Average gas price in the incoming transaction	Dou ble
23	$f23\_total\_failed\_transactions\_incoming$	Total failed transactions in the incoming transaction	Inte ger
24	$f24\_total\_failed\_transactions\_outgoing$	Total failed transactions in the outgoing transaction	Inte ger
25	$f25\_total\_failed\_transactions$	Total failed transactions	Inte ger
26	$f26\_total\_success\_transactions\_incoming$	Total successful transactions in the incoming transaction	Inte ger
27	$f27\_total\_success\_transactions\_outgoing$	Total successful transactions in the outgoing transaction	Inte ger
28	$f28\_total\_success\_transactions$	Total successful transactions	Inte ger
29	$f29\_gas\_used\_incoming\_transaction$	Total gas used in the incoming transaction	Long
30	$f30\_gas\_used\_outgoing\_transaction$	Total gas used in the outgoing transaction	Long
31	$f31\_percentage\_transaction\_sent$	Percentage of transactions sent	Dou ble
32	$f32\_percentage\_transaction\_received$	Percentage of transactions received	Dou ble
33	$f33\_standard\_deviation\_ether\_incoming$	Standard deviation of ether in incoming transaction	Dou ble
34	$f34\_standard\_deviation\_ether\_outgoing$	Standard deviation of ether in outgoing transaction	Dou ble
35	$f35\_standard\_deviation\_gas\_price\_incoming$	Standard deviation of gas price in incoming transaction	Dou ble

36	$f36\_standard\_deviation\_gas\_price\_outgoing$	Standard deviation of gas price in outgoing transaction	Double
37	$f37\_first\_transaction\_bit$	0/1 (0 if last transaction is incoming transaction else 1)	Bit
38	$f38\_first\_transaction\_value$	Ether value in first transaction	Long
39	$f39\_mean\_in\_time\ (seconds)$	Mean time difference between incoming transaction	Double
40	$f40\_mean\_out\_time$	Mean time difference between outgoing transaction	Double
41	$f41\_mean\_time$	Mean time difference between all transaction	Double
42	$f42\_transaction\_fee\_spent\_incoming$	Total fee spent in incoming transaction	Long
43	$f43\_transaction\_fee\_spent\_outgoing$	Total fee spent in outgoing transaction	Long
44	$f44\_transaction\_fee\_spent$	Total fee spent	Long

Appendix C

Table 6: Features extracted from Smart Contract Address

	Feature	Description	Unit
1	$f1\_contract\_creation\_time$	Contract creation time	Long
2	$f2\_transaction\_fee\_spent\_contract\_creation$	Transaction fee spent in contract creation	Long
3	$f3\_percentage\_gas\_used\_contract\_creation$	Gas price used in contract creation	Double
4	$f4\_gas\_price\_contract\_creation$	Block timestamp for contract creation	Long
5	$f5\_first\_contract\_invoke\_time$	First contract invoke timestamp	Long
6	$f6\_last\_contract\_invoke\_time$	Last contract invoke timestamp	Long
7	$f7\_active\_duration\ (seconds)$	Active duration of the contract address	Long
8	$f8\_total\_invocations$	Total invocations from the contract address	Integer
9	$f9\_total\_unique\_invocations$	Total unique invocations from the contract address	Integer
10	$f10\_avg\_gas\_used\_contract\_invocations$	Average percentage of gas used in contract invocations	Double

11	$f11\_total\_gas\_price\_contract\_invocations$	Total gas price used in contract invocations	Long
12	$f12\_avg\_gas\_price\_contract\_invocations$	Average gas price used in contract invocations	Double
13	$f13\_total\_tx\_fee\_contract\_invocations$	Total transaction fee used in contract invocations	Long
14	$f14\_avg\_tx\_fee\_contract\_invocations$	Average transaction fee used in contract invocations	Double
15	$f15\_total\_ether\_contract\_invocations$	Total ether in contract invocations	Big Integer
16	$f16\_average\_ether\_contract\_invocations$	Average ether in contract invocations	Big Integer
17	$f17\_total\_gas\_used\_contract\_invocations$	Total gas used in contract invocations	Long
18	$f18\_avg\_gas\_used\_contract\_invocations$	Average gas used in contract invocations	Double

Appendix D

Get Account type categorizes the account as EOA or SC

```
public Account getAccountType(EthereumTransaction firstTx) {
    if (firstTx.getInput().equals("0x") && firstTx.getContractAddress().length()
== 0) {
        return Account.EOA;
    }

    if (firstTx.getTo().length() == 0 && firstTx.getContractAddress().length() !=
0) {
        return Account.SMARTCONTRACT;
    }

    return Account.NONE;
}
```

Appendix E

Get Ethereum Transactions

```
public EthereumTransactions getEthereumTransactions(String address) throws IO-
Exception {
    EthereumTransactions data = new EthereumTransactions();
    StringBuffer content = null;
    String endpoint = ENDPOINT + address.toLowerCase() +
API_QUERY_STRING+this.ApiKey;
    URL url = new URL(endpoint);
    HttpURLConnection con = (HttpURLConnection) url.openConnection();
    try {
        con.setRequestMethod("GET");
        int status = con.getResponseCode();
        if (status == 200) {
            BufferedReader in = new BufferedReader(new InputStreamRead-
er(con.getInputStream()));
            String inputLine;
            content = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
            }
            in.close();
        }
        data = new Gson().fromJson(content.toString(), EthereumTransac-
tions.class);
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("Error occured for this address: "+address+"\n"+ con-
tent.toString());
        return data;
    } finally {
        con.disconnect();
    }

    if (data.getResult()!=null && data.getResult().size() != 0) {
        Account type = getAccountType(data.getResult().get(0));
        data.setType(type);
    } else {
        data.setType(Account.NONE);
    }
    return data;

}
```

Appendix F

```
public int f1_total_transactions_sent() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            count += 1;
        }
    }
    return count;
}

public int f2_total_transactions_received() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            count += 1;
        }
    }
    return count;
}

public BigInteger f3_value_out() {
    BigInteger value = new BigInteger("0");
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            BigInteger t = new BigInteger(tx.getValue());
            value = value.add(t);
        }
    }
    return value;
}

public BigInteger f4_value_in() {
    BigInteger value = new BigInteger("0");
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            BigInteger t = new BigInteger(tx.getValue());
            value = value.add(t);
        }
    }
    return value;
}
```

```java
public BigInteger f5_value_difference() {
    return (this.f3_value_out().subtract(this.f4_value_in())).abs();
}

public int f6_number_of_distinct_address_contacted() {
    Set<String> seen = new HashSet<String>();
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            seen.add(tx.getTo());
        }

        if (tx.getTo().equals(this.address)) {
            seen.add(tx.getFrom());
        }
    }
    return seen.size();
}

public int f7_total_transactions_sent_received() {
    return this.f1_total_transactions_sent() +
this.f2_total_transactions_received();
}

public int f8_total_transactions_sent_to_unique_address() {
    Set<String> seen = new HashSet<String>();
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            seen.add(tx.getTo());
        }
    }
    return seen.size();
}

public int f9_total_transactions_received_from_unique_address() {
    Set<String> seen = new HashSet<String>();
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            seen.add(tx.getFrom());
        }
    }
    return seen.size();
}
```

```java
public Long f10_first_transaction_time() {
    return Long.parseLong(this.getFirstTransaction().getTimeStamp());
}

public Long f11_last_transaction_time() {
    return Long.parseLong(this.getLastTransaction().getTimeStamp());
}

public Long f12_transaction_active_duration() {
    return
Duration.between(Instant.ofEpochSecond(this.f10_first_transaction_time()),
            Instant.ofEpochSecond(this.f11_last_transaction_time())).toSeconds();
}

public int f13_last_txn_bit() {
    if (this.getLastTransaction().getFrom().equals(this.address)) {
        return 1;
    }
    if (this.getLastTransaction().getTo().equals(this.address)) {
        return 0;
    }
    return 0;
}

public BigInteger f14_last_transaction_value() {
    BigInteger value = new BigInteger(this.getLastTransaction().getValue());
    return value;
}

public BigInteger f15_average_incoming_ether() {
    BigInteger value = new BigInteger("0");
    BigInteger count = new BigInteger("0");
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            BigInteger t = new BigInteger(tx.getValue());
            value = value.add(t);
            count = count.add(new BigInteger("1"));
        }
    }
    if (!count.equals(new BigInteger("0"))) {
        return value.divide(count);

    } else {
        return new BigInteger("0");
    }
```

```java
public BigInteger f16_average_outgoing_ether() {
    BigInteger value = new BigInteger("0");
    BigInteger count = new BigInteger("0");
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            BigInteger t = new BigInteger(tx.getValue());
            value = value.add(t);
            count = count.add(new BigInteger("1"));
        }
    }
    if (!count.equals(new BigInteger("0"))) {
        return value.divide(count);

    } else {
        return new BigInteger("0");
    }

}

public double f17_average_percentage_gas_incoming() {
    long gas = 0;
    long total_gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGas());

        }
        total_gas = total_gas + Long.parseLong(tx.getGas());
    }
    return ((double) gas / total_gas) * 100;
}

public double f18_average_percentage_gas_outgoing() {
    long gas = 0;
    long total_gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGas());

        }
        total_gas = total_gas + Long.parseLong(tx.getGas());
    }
    return ((double) gas / total_gas) * 100;
}
```

```java
public long f19_outgoing_gas_price() {
    long gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGasPrice());

        }
    }
    return gas;
}

public long f20_incoming_gas_price() {
    long gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGasPrice());

        }
    }
    return gas;
}

public double f21_average_incoming_gas_price() {
    long gas = 0;
    long total = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGasPrice());
            total = total + 1;
        }
    }
    return ((double) gas / total);
}

public double f22_average_outgoing_gas_price() {
    long gas = 0;
    long total = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGasPrice());
            total = total + 1;
        }
    }
    return ((double) gas / total);
}
```

```java
public int f23_total_failed_transactions_incoming() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address) &&
tx.getTxreceipt_status().equals("0")) {
            count = count + 1;
        }
    }
    return count;
}

public int f24_total_failed_transactions_outgoing() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address) &&
tx.getTxreceipt_status().equals("0")) {
            count = count + 1;
        }
    }
    return count;
}

public int f25_total_failed_transactions() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTxreceipt_status().equals("0")) {
            count = count + 1;
        }
    }
    return count;
}

public int f26_total_success_transactions_incoming() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTxreceipt_status().equals("1") &&
tx.getTo().equals(this.address)) {
            count = count + 1;
        }
    }
    return count;
}
```

```java
public int f27_total_success_transactions_outgoing() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTxreceipt_status().equals("1") &&
tx.getFrom().equals(this.address)) {
            count = count + 1;
        }
    }
    return count;
}

public double f28_total_success_transactions() {
    int count = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTxreceipt_status().equals("1")) {
            count = count + 1;
        }
    }
    return count;
}

public long f29_gas_used_incoming_transaction() {
    long gasUsed = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            gasUsed = gasUsed + Long.parseLong(tx.getGasUsed());
        }
    }
    return gasUsed;
}

public long f30_gas_used_outgoing_transaction() {
    long gasUsed = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            gasUsed = gasUsed + Long.parseLong(tx.getGasUsed());
        }
    }
    return gasUsed;
}
```

```java
public double f31_percentage_transaction_sent() {
    int total = (this.f1_total_transactions_sent() +
this.f2_total_transactions_received());
    if (total != 0) {
        return ((this.f1_total_transactions_sent() / total) * 100);
    }
    return 0;
}

public double f32_percentage_transaction_received() {
    int total = (this.f1_total_transactions_sent() +
this.f2_total_transactions_received());
    if (total != 0) {
        return ((this.f2_total_transactions_received() / total) * 100);
    }
    return 0;
}

public double f33_standard_deviation_ether_incoming() {
    int max = this.transactions.size();
    double[] values = new double[max];
    int index = 0;
    for (EthereumTransaction tx : this.transactions) {

        if (tx.getTo().equals(this.address)) {
            values[index] = Double.parseDouble(tx.getValue());
            index++;
        }
    }
    StandardDeviation sd = new StandardDeviation(false);
    return sd.evaluate(values);
}

public double f34_standard_deviation_ether_outgoing() {
    int max = this.transactions.size();
    double[] values = new double[max];
    int index = 0;
    for (EthereumTransaction tx : this.transactions) {

        if (tx.getFrom().equals(this.address)) {
            values[index] = Double.parseDouble(tx.getValue());
            index++;
        }
    }
    StandardDeviation sd = new StandardDeviation(false);
```

```java
public double f35_standard_deviation_gas_price_incoming() {
    int max = this.transactions.size();
    double[] values = new double[max];
    int index = 0;
    for (EthereumTransaction tx : this.transactions) {

        if (tx.getTo().equals(this.address)) {
            values[index] = Double.parseDouble(tx.getGasPrice());
            index++;
        }
    }
    StandardDeviation sd = new StandardDeviation(false);
    return sd.evaluate(values);
}

public double f36_standard_deviation_gas_price_outgoing() {
    int max = this.transactions.size();
    double[] values = new double[max];
    int index = 0;
    for (EthereumTransaction tx : this.transactions) {

        if (tx.getFrom().equals(this.address)) {
            values[index] = Double.parseDouble(tx.getGasPrice());
            index++;
        }
    }
    StandardDeviation sd = new StandardDeviation(false);
    return sd.evaluate(values);
}

public int f37_first_transaction_bit() {
    if (this.getFirstTransaction().getFrom().equals(this.address)) {
        return 1;
    }
    if (this.getFirstTransaction().getTo().equals(this.address)) {
        return 0;
    }
    return 0;
}

public BigInteger f38_first_transaction_value() {
    BigInteger value = new BigInteger(this.getFirstTransaction().getValue());
    return value;
}
```

```java
public double f39_mean_in_time() {
    List<Long> all_time = new ArrayList<Long>();
    List<Long> times = new ArrayList<Long>();
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            all_time.add(Long.parseLong(tx.getTimeStamp()));

        }
    }
    for (Long time : all_time) {
        int currentIndex = all_time.indexOf(time);
        if (currentIndex + 1 < all_time.size()) {
            Duration timeElapsed =
Duration.between(Instant.ofEpochSecond(all_time.get(currentIndex)),
                    Instant.ofEpochSecond(all_time.get(currentIndex + 1)));
            times.add(timeElapsed.toSeconds());
        }
    }
    if (times.size() == 0) {
        return 0;
    } else {
        double sum = 0;
        for (int i = 0; i < times.size(); i++) {
            sum += (double) times.get(i) / (double) times.size();
        }
        return sum;
    }
}

public double f40_mean_out_time() {
    List<Long> all_time = new ArrayList<Long>();
    List<Long> times = new ArrayList<Long>();
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            all_time.add(Long.parseLong(tx.getTimeStamp()));

        }
    }
    for (Long time : all_time) {
        int currentIndex = all_time.indexOf(time);
        if (currentIndex + 1 < all_time.size()) {
            Duration timeElapsed =
Duration.between(Instant.ofEpochSecond(all_time.get(currentIndex)),
                    Instant.ofEpochSecond(all_time.get(currentIndex + 1)));
            times.add(timeElapsed.toSeconds());
```

```java
public double f41_mean_time() {
    List<Long> all_time = new ArrayList<Long>();
    List<Long> times = new ArrayList<Long>();
    for (EthereumTransaction tx : this.transactions) {
        all_time.add(Long.parseLong(tx.getTimeStamp()));
    }
    for (Long time : all_time) {
        int currentIndex = all_time.indexOf(time);
        if (currentIndex + 1 <= all_time.size() - 1) {
            Duration timeElapsed =
Duration.between(Instant.ofEpochSecond(all_time.get(currentIndex)),
                    Instant.ofEpochSecond(all_time.get(currentIndex + 1)));
            times.add(timeElapsed.toSeconds());
        }
    }
    if (times.size() == 0) {
        return 0;
    } else {
        double sum = 0;
        for (int i = 0; i < times.size(); i++) {
            sum += (double) times.get(i) / (double) times.size();
        }
        return sum;
    }
}

public long f42_transaction_fee_spent_incoming() {
    long gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getTo().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGas());
        }
    }
    return gas;
}

public long f43_transaction_fee_spent_outgoing() {
    long gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getFrom().equals(this.address)) {
            gas = gas + Long.parseLong(tx.getGas());
        }
    }
    return gas;
```

Appendix F

Smart Contract Feature extraction

```
public long f1_contract_creation_time() {
    return Long.parseLong(getFirstTransaction().getTimeStamp());
}

public long f2_transaction_fee_spent_contract_creation() {
    return Long.parseLong(getFirstTransaction().getGas());
}

public double f3_percentage_gas_used_contract_creation() {
    return 100 * (Double.parseDouble(this.getFirstTransaction().getGasUsed())
            / Double.parseDouble(this.getFirstTransaction().getGas()));
}

public long f4_gas_price_contract_creation() {
    return Long.parseLong(getFirstTransaction().getGasPrice());
}

public long f5_first_contract_invoke_time() {
    return Long.parseLong(this.getSecondTransaction().getTimeStamp());
}

public long f6_last_contract_invoke_time() {
    return Long.parseLong(this.getLastTransaction().getTimeStamp());
}

public long f7_active_duration() {
    return
Duration.between(Instant.ofEpochSecond(this.f5_first_contract_invoke_time()),
            Instant.ofEpochSecond(this.f6_last_contract_invoke_time())).toSeconds
();
}

public int f8_total_invocations() {
    return this.transactions.size() - 1;
}

public int f9_total_unique_invocations() {
    List<String> gasList = new ArrayList<String>();
    for (EthereumTransaction tx : this.transactions) {
        gasList.add(tx.getFrom() + "-" + tx.getTo());
    }
    Set<String> uniqueGas = new HashSet<String>(gasList);
    return uniqueGas.size();
}
```

```java
public double f10_avg_gas_used_contract_invocations() {
    long gasUsed = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gasUsed = gasUsed + Long.parseLong(tx.getGasUsed());
        }
    }

    if (this.f8_total_invocations() != 0) {
        return gasUsed / this.f8_total_invocations();
    } else {
        return 0;
    }
}

public long f11_total_gas_price_contract_invocations() {
    long gasPrice = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gasPrice = gasPrice + Long.parseLong(tx.getGasPrice());
        }
    }
    return gasPrice;
}

public double f12_avg_gas_price_contract_invocations() {
    long gasPrice = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gasPrice = gasPrice + Long.parseLong(tx.getGasPrice());
        }
    }
    if (this.f8_total_invocations() != 0) {
        return gasPrice / this.f8_total_invocations();
    } else {
        return 0;
    }
}

public long f13_total_tx_fee_contract_invocations() {
    long gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gas = gas + Long.parseLong(tx.getGas());
        }
```

```
public double f14_avg_tx_fee_contract_invocations() {
    long gas = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gas = gas + Long.parseLong(tx.getGas());
        }
    }
    if (this.f8_total_invocations() != 0) {
        return gas / this.f8_total_invocations();
    } else {
        return 0;
    }
}

public BigInteger f15_total_ether_contract_invocations() {
    BigInteger value = new BigInteger("0");
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            BigInteger t = new BigInteger(tx.getValue());
            value = value.add(t);
        }
    }
    return value;
}

public BigInteger f16_average_ether_contract_invocations() {
    BigInteger value = new BigInteger("0");
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            BigInteger t = new BigInteger(tx.getValue());
            value = value.add(t);
        }
    }
    BigInteger t = new BigInteger(String.valueOf(this.f8_total_invocations()));
    if (this.f8_total_invocations() != 0) {
        return value.divide(t);
    } else {
        return new BigInteger("0");
    }
}
```

```java
public long f17_total_gas_used_contract_invocations() {
    long gasUsed = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gasUsed = gasUsed + Long.parseLong(tx.getGasUsed());
        }
    }
    return gasUsed;
}

public double f18_avg_gas_used_contract_invocations() {
    long gasUsed = 0;
    for (EthereumTransaction tx : this.transactions) {
        if (tx.getContractAddress().isEmpty()) {
            gasUsed = gasUsed + Long.parseLong(tx.getGasUsed());
        }
    }
    if (this.f8_total_invocations() != 0) {
        return gasUsed / this.f8_total_invocations();
    } else {
        return 0;
    }

}
```

Appendix F

Map Reduce: Partitioner

```java
public int getPartition(Text key, Text arg1, int numReduceTasks) {
    String[] str = key.toString().split(",");
    Account accountType = null;
    if (str.length > 1) {
        accountType = Account.valueOf(str[1]);
    } else {
        accountType = Account.NONE;
    }
    if (numReduceTasks == 0) {
        return 0;
    }
    if (accountType.equals(Account.EOA)) {
        return 0;
    } else if (accountType.equals(Account.SMARTCONTRACT)) {
        return 1 % numReduceTasks;
    } else {
        return 2 % numReduceTasks;
    }
}
```

Map Reduce: Mapper for Felonious Data

```
public void map(LongWritable key, Text values, Context context) throws
IOException, InterruptedException {
    Text mapKey = new Text();
    Text accountType = new Text();
    Text output = new Text();
    if (key.get() == 0 && values.toString().contains("Address"))
        return;
    else {
        String line = values.toString();
        String[] data = line.split(",");

        EtherScan s = new EtherScan(API_KEY);
        EthereumTransactions t =
s.getEthereumTransactions(data[0].toLowerCase());
        if (t.getResult().size() > 1) {
            if
(s.getAccountType(t.getResult().get(0)).equals(EtherScan.Account.EOA)) {
                accountType.set(EtherScan.Account.EOA.toString());
                EOA eoa = new EOA(data[0], t.getResult());
                output.set(eoa.getAllFeatures());

            }

            if
(s.getAccountType(t.getResult().get(0)).equals(Account.SMARTCONTRACT)) {
                accountType.set(EtherScan.Account.SMARTCONTRACT.toString());
                SmartContract sc = new SmartContract(data[0], t.getResult());
                output.set(sc.getAllFeatures());
            }

            mapKey.set(data[0] + "," + accountType.toString());

            context.write(mapKey, output);
        }
    }

}
```

Map Reduce: Mapper for Non- Felonious Data

```
public void map(LongWritable key, Text values, Context context) throws
IOException, InterruptedException {
    Text mapKey = new Text();
    Text accountType = new Text();
    Text output = new Text();
    if (key.get() == 0 && values.toString().contains("Address"))
        return;
    else {
        String line = values.toString();
        String[] data = line.split(",");

        EtherScan s = new EtherScan(API_KEY);
        EthereumTransactions t =
s.getEthereumTransactions(data[0].toLowerCase());
        List<String> invalidAccounts = NonFelonious.getInvalidAccounts();

        if (t.getResult().size() > 1 &&
NonFelonious.validateNonFeloniousAccount(invalidAccounts, t.getResult())) {
            if
(s.getAccountType(t.getResult().get(0)).equals(EtherScan.Account.EOA)) {
                accountType.set(EtherScan.Account.EOA.toString());
                EOA eoa = new EOA(data[0], t.getResult());
                output.set(eoa.getAllFeatures());

            }

            if
(s.getAccountType(t.getResult().get(0)).equals(Account.SMARTCONTRACT)) {
                accountType.set(EtherScan.Account.SMARTCONTRACT.toString());
                SmartContract sc = new SmartContract(data[0], t.getResult());
                output.set(sc.getAllFeatures());
            }

            mapKey.set(data[0] + "," + accountType.toString());

            context.write(mapKey, output);
        }
    }

}
```

Map Reduce: Reducer

```
public void reduce(Text t_key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
Text output = new Text();
Text mapKey = new Text();
String address = t_key.toString().split(",")[0];

for (Text tx : values) {
output.set(tx+","+"1");
}
mapKey.set(address);
context.write(mapKey, new Text(output));
}
```

Non -Felonious Helper Functions

```
public static List<String> getInvalidAccounts() throws IOException {
    List<String> records = new ArrayList<String>();
    try (BufferedReader br = new BufferedReader(new FileReader(INVAID_CSV_PATH)))
{
        String line;
        while ((line = br.readLine()) != null) {
            String[] values = line.split(COMMA_DELIMITER);
            if (!values[0].equals("Address")) {
                records.add(values[0]);
            }
        }
    }
    return records;
}

//Returns true if the account is a valid non felonious account
public static boolean validateNonFeloniousAccount(List<String> invalidAccounts,
        List<EthereumTransaction> transactions) {
    List<String> accountsInteractedWith = new ArrayList<String>();

    for (EthereumTransaction tx : transactions) {
        accountsInteractedWith.add(tx.getFrom().toLowerCase());
        accountsInteractedWith.add(tx.getTo().toLowerCase());
    }

    Set<String> uniqueAccountsInteractedWith = new
HashSet<String>(accountsInteractedWith);
    return Collections.disjoint(uniqueAccountsInteractedWith, invalidAccounts);

}
```

List of Publications/Conference Presentations, if any.

- The project has been published in the inter departmental Technical Paper Writing Hackathon: Tech-Writeathon 2022

Duly Completed Checklist

a.	Is the Cover page in proper format?	**Y** / N
b.	Is the Title page in proper format?	**Y** / N
c.	Is the Certificate from the Supervisor in proper format? Has it been signed?	**Y** / N
d.	Is Abstract included in the Report? Is it properly written?	**Y** / N
e.	Does the Table of Contents page include chapter page numbers?	**Y** / N
f.	Does the Report contain a summary of the literature survey?	**Y** / N
i.	Are the Pages numbered properly?	**Y** / N
ii.	Are the Figures numbered properly?	**Y** / N
iii.	Are the Tables numbered properly?	**Y** / N
iv.	Are the Captions for the Figures and Tables proper?	**Y** / N
v.	Are the Appendices numbered?	**Y** / N
g.	Does the Report have Conclusion / Recommendations of the work?	**Y** / N
h.	Are References/Bibliography given in the Report?	**Y** / N
i.	Have the References been cited in the Report?	**Y** / N
j.	Is the citation of References / Bibliography in proper format?	**Y** / N

Vishnu Prasad

Digitally signed by Vishnu Prasad
Date: 2022.02.27 23:14:54 +05'30'

(Signature of Student)

Date: February,2022

(Signature of Supervisor)

Date: February,2022