

# Cryptocurrency Tracing Tools

Project Guide: Dr. M.B.Patil

Team Members:

- 1) Shriraj Vallamdeshi
- 2) Vinay Hipparge
- 3) Rahul Chavan
- 4) Sairam Gudeli

Detail:

Creating a Cryptocurrency Investigation Tool. By which we can Detect Fraud Address in Ethereum Using SVM & and Try to find the Details of that particular address. Using Various API's.



## **Abstract**

### **1. Support Vector Machines (SVM) for Anomaly Detection:**

The SVM (a machine learning algorithm) is used for enhanced anomaly detection. The SVM model is applied to transaction data, allowing for the identification of unusual patterns and potential fraudulent activities within the blockchain network.

### **2. Address Details and Transactions:**

The initial code focuses on fetching and analyzing Ethereum address details and transactions. It computes key metrics such as total sent and received transactions, ether values, and transaction timelines, offering valuable insights into an address's transaction history.

### **3. Transaction Path Visualization:**

This extends the analysis to visualize transaction paths between addresses. Utilizing the vis.js library, it generates an interactive graph illustrating the flow of transactions and identifies potential exchanges associated with specific address patterns.

### **4. Token Balances:**

This provides a comprehensive tool for assessing token balances associated with a given Ethereum address. It fetches balances for various tokens using the Etherscan API and displays the results in a structured table, offering a concise overview of token holdings.

### **5. Suspicious Address Detection:**

Two separate codes contribute to suspicious address detection. The first involves analyzing CSV data to identify senders and recipients with more than ten transactions. The second code visualizes transaction data in a 3D scatter plot, incorporating K-Means clustering to highlight potential clusters of interest. These tools are crucial for identifying anomalous transaction patterns and detecting potential suspicious activities.

### **6. Address clustering**

This focuses on the application of K-Means clustering for the analysis of transaction data within the blockchain network. The code provides a versatile and visually intuitive tool for identifying clusters of transactions based on transaction value, gas usage, and time. By leveraging the Plotly.js library, the code generates an interactive 3D scatter plot, where each data point represents a transaction, and the color-coded clusters are determined by the K-Means algorithm.

## INDEX

Chapter	Table	Page.no
CHAPTER 1	Fraud Detection with SVM	4
	1. Introduction	
	2. Code Overview:	
	2.1 Data Loading and Cleaning	
	2.2 Exploratory Data Analysis (EDA)	
	2.3 Data Splitting & Transformation	
	2.4 Model Training	
	2.5 Model Evaluation	
	2.6 Model Serialization	
	3. Output	
CHAPTER 2	Address Details	6
CHAPTER 3	Visualization Using Graph:	8
	3.1 Fetch Transactions	
	3.2 Visualize Transaction Path	
	3.3 Get Exchange from Address	
CHAPTER 4	Fetching Token Balance	9
CHAPTER 5	Suspicious Address Detection	10
CHAPTER 6	Address Clustering	12
	6.1 Data Processing:	
	6.2 Scatter Plot Creation	
	6.3 Plotly.js Integration	

# CHAPTER 1

## Fraud Detection with SVM

### 1. Introduction:

The provided Python script is focused on building a fraud detection model using Support Vector Machines (SVM). The dataset used is a transaction dataset, and the script involves data preprocessing, model training, evaluation, and saving/loading the trained SVM model using pickle.

### 2. Code Overview:

#### 2.1 Data Loading and Cleaning:

- The script starts by importing necessary libraries, including pandas, scikit-learn, imbalanced-learn, and matplotlib.
- The dataset is loaded using pandas, and irrelevant columns are dropped.
- Data cleaning is performed, including handling missing values, dropping constant columns, and removing specific columns based on the 'drop' list.

#### 2.2 Exploratory Data Analysis (EDA):

- The target variable distribution is visualized using a pie chart to show the proportion of fraud and non-fraud transactions.

#### 2.3 Data Splitting & Transformation:

- The dataset is split into training and testing sets using the train\_test\_split function from scikit-learn.
- Power transformation is applied to normalize the features using the Power Transformer from scikit-learn.
- Synthetic Minority Over-sampling Technique (SMOTE) is used to address class imbalance.

#### 2.4 Model Training:

- An SVM classifier is trained using the radial basis function (RBF) kernel and the oversampled training data.

#### 2.5 Model Evaluation:

- The trained SVM model is evaluated on the test set, and metrics such as classification report, confusion matrix, and accuracy are printed.

#### 2.6 Model Serialization:

- The trained SVM model is saved using pickle for future use.
- The saved SVM model is loaded back into memory using pickle.

- The loaded model is used to make predictions on the entire dataset, and transactions classified as fraud are displayed.

### 3. Output:

- The script effectively demonstrates the process of building a fraud detection model using SVM, addressing issues like class imbalance with SMOTE.
- The model achieves a certain level of **Accuracy of 94%**

### Result

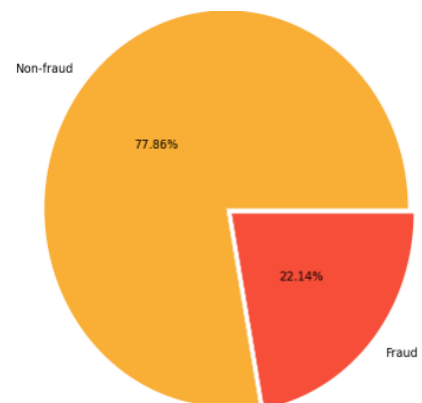
Total number of fraud transactions: 2351

#### List of Fraud Transactions

	FLAG	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	max value received	avg val received	avg val sent	min value sent to contract	total Ether sent	total ether balance	ERC20 total Ether received	ERC20 total ether sent	ERC20 total Ether sent contract	ERC20 unq sent addr	ERC20 unq sent addr.1	ERC20 unq rec token name
13	0	9520.70	5776.32	78197.58	7	2	0	0.265159	0.180079	0.037563	0.0	0.262941	0.097217	8.001000e+00	0.000000e+00	0.0	0.0	0.0	2.0
33	0	2152.95	680.22	22889.93	10	2	0	0.100000	0.075000	0.011073	0.0	0.110734	0.039266	1.000000e-12	1.000000e-11	0.0	1.0	0.0	1.0
36	0	0.00	0.00	4.87	1	1	0	0.003000	0.003000	0.000000	0.0	0.000000	0.003000	3.750000e-12	5.000000e-15	0.0	1.0	0.0	1.0
62	0	621.07	5012.75	93911.63	14	17	4	379.000000	39.759632	48.273105	0.0	675.823470	0.090279	2.407524e-03	0.000000e+00	0.0	0.0	0.0	1.0
80	0	674.61	221.00	11677.73	16	4	0	250.500000	63.377500	15.825000	0.0	253.200000	0.310000	1.846864e+04	9.013333e-01	0.0	4.0	0.0	11.0
114	0	19121.43	11.04	267722.13	14	2	0	6.601200	3.749150	0.677948	0.0	9.491268	-1.992968	4.517041e-01	0.000000e+00	0.0	0.0	0.0	3.0
146	0	6736.68	1019.60	182217.27	16	73	0	1.000000	0.123734	0.564130	0.0	9.026073	0.006507	2.708425e-02	0.000000e+00	0.0	0.0	0.0	1.0
188	0	8691.10	4980.40	32323.40	2	3	0	5.000000	2.086115	3.128731	0.0	6.257462	0.000883	3.580000e-07	0.000000e+00	0.0	0.0	0.0	2.0
235	0	0.00	781.62	1610.60	1	2	0	0.010000	0.009002	0.000000	0.0	0.000000	0.018004	8.823770e-02	1.000000e+01	0.0	1.0	0.0	2.0
253	0	2104.21	5422.52	17157.67	3	2	0	0.133587	0.099100	0.065425	0.0	0.196274	0.001926	1.262260e+02	1.262260e+02	0.0	1.0	0.0	1.0
269	0	0.00	0.00	0.00	0	1	0	0.126419	0.126419	0.000000	0.0	0.000000	0.126419	3.925158e+01	0.000000e+00	0.0	0.0	0.0	5.0
272	0	4222.44	1244.43	68666.87	3	45	0	2.323534	0.133748	2.005772	0.0	6.017317	0.001323	1.566222e-02	0.000000e+00	0.0	0.0	0.0	1.0
286	0	39172.64	52599.21	341341.32	2	5	0	0.520570	0.119699	0.298903	0.0	0.597805	0.000691	4.478120e-02	0.000000e+00	0.0	0.0	0.0	1.0
310	0	14354.08	2854.57	443352.32	11	100	0	0.124775	0.049840	0.452565	0.0	4.978214	0.005739	1.139212e+03	3.886964e+02	0.0	1.0	0.0	7.0
332	0	1714.67	7067.61	158290.03	14	19	0	1.003510	0.649381	0.880469	0.0	12.326565	0.011680	1.927179e-02	0.000000e+00	0.0	0.0	0.0	1.0
362	0	0.00	889.07	4330.22	1	4	0	0.874667	0.432863	0.037635	0.0	0.037635	1.693817	3.241960e+02	0.000000e+00	0.0	0.0	0.0	7.0
396	0	4700.39	4415.20	190001.52	16	26	0	1.008829	0.504703	0.814544	0.0	13.032703	0.089585	6.916904e+01	0.000000e+00	0.0	0.0	0.0	2.0
398	0	0.00	0.00	41008.07	1	1	0	5.000000	5.000000	4.998740	0.0	4.998740	0.001260	3.766315e-01	0.000000e+00	0.0	0.0	0.0	1.0

	precision	recall	f1-score	support
0	0.99	0.96	0.98	1547
1	0.88	0.96	0.92	422
accuracy			0.96	1969
macro avg	0.93	0.96	0.95	1969
weighted avg	0.96	0.96	0.96	1969

```
[[1490 57]
 [ 17 405]]
Accuracy: 0.9624174707973591
```



## CHAPTER 2

### Address Details

This Function gets an Address of Ethereum & Using API Etherscan API gives all information. Related to Entered Address. Such as

1. Total Sent Transactions
2. Total Received Transactions
3. Total No. Transactions
4. Total Ether Received
5. Total Ether Sent
6. Max Value Received
7. Interacted with Unique Addresses
8. Ether Received Per Transaction
9. Time Difference Between First and Last transaction

Back

Details

### Ethereum Address Information

Enter Ethereum Address:

Get Info

Total Sent Transactions 333	Total Received Transactions 84	Total Transactions 417
Total Ether Received 11.4441 Ether	Total Ether Sent 16.1610 Ether	Maximum Value Received 2.6000 Ether
Unique Addresses Interacted With 7	Ether Received per Transaction 0.1362 Ether	Time Difference (1st to Last Transaction) 994633.32 minutes

And also in Addition we can fetch all transaction Send & Receive done by a Entered Address

Back

### Transaction History In Details

Enter ETH Address:

0xbba9a171847ee8d4a8624e7e59163c79e5e32c52

Submit

Block Number	Timestamp	Value	From	To
14487481	Wed Mar 30 2022 17:40:16 GMT+0530 (India Standard Time)	0.2 ETH	0x3f6249c4d782aab00b5e9ab99e090035b80e0c15	0xbba9a171847ee8d4a8624e7e59163c79e5e32c52
14489862	Thu Mar 31 2022 02:38:50 GMT+0530 (India Standard Time)	0.1 ETH	0xbba9a171847ee8d4a8624e7e59163c79e5e32c52	0x3f6249c4d782aab00b5e9ab99e090035b80e0c15
14490001	Thu Mar 31 2022 03:13:13 GMT+0530 (India Standard Time)	0.09 ETH	0xbba9a171847ee8d4a8624e7e59163c79e5e32c52	0x3f6249c4d782aab00b5e9ab99e090035b80e0c15
14519616	Mon Apr 04 2022 18:17:08 GMT+0530 (India Standard Time)	0.03 ETH	0x3f6249c4d782aab00b5e9ab99e090035b80e0c15	0xbba9a171847ee8d4a8624e7e59163c79e5e32c52
14524802	Tue Apr 05 2022 13:50:29 GMT+0530 (India Standard Time)	0.09 ETH	0x3f6249c4d782aab00b5e9ab99e090035b80e0c15	0xbba9a171847ee8d4a8624e7e59163c79e5e32c52
14531822	Wed Apr 06 2022 16:02:49 GMT+0530 (India Standard Time)	0 ETH	0xbba9a171847ee8d4a8624e7e59163c79e5e32c52	0x75c71673808cb8d54274775b10abd463fc738e82

## CHAPTER 3

### Visualization Using Graph

#### 3.1 Fetch Transactions:

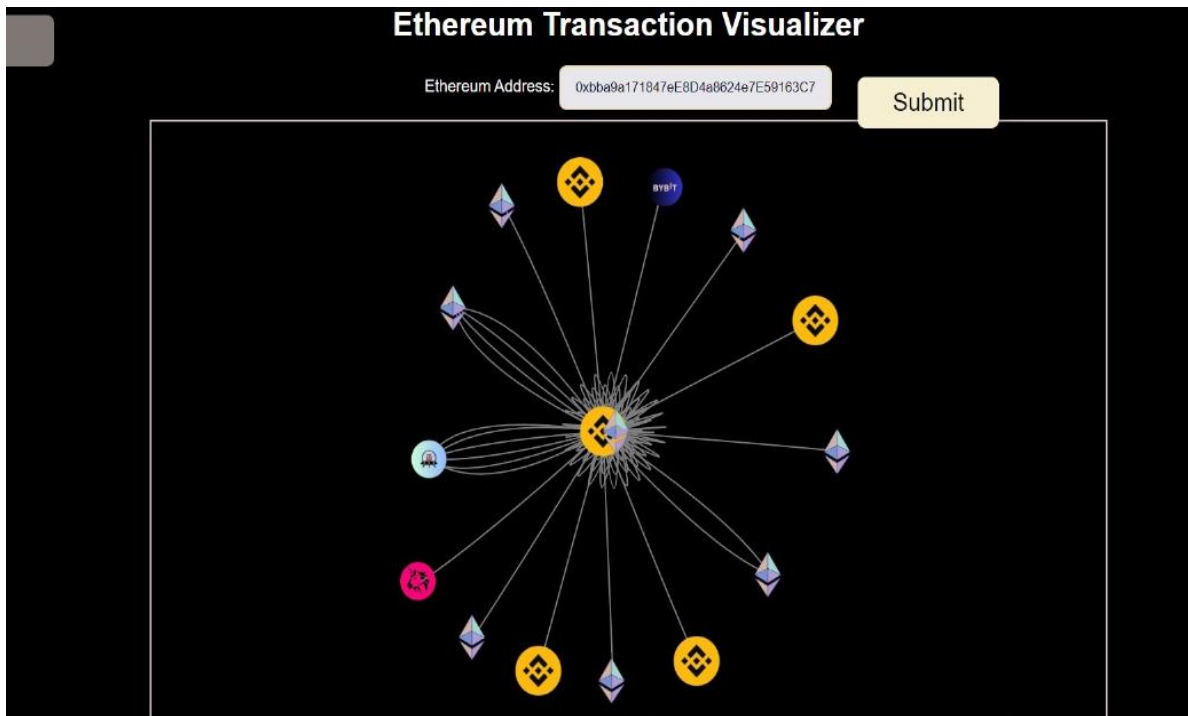
- Retrieves the Ethereum address entered by the user.
- Initiates the process of fetching and visualizing internal Ethereum transactions.
- Constructs the API URL for Etherscan using the provided API key and Ethereum address.
- Sends a request to the Etherscan API to obtain internal transaction data.
- If the status is "1" (success), calls a function to visualize the transaction path.

#### 3.2 Visualize Transaction Path:

- Processes the fetched transaction data to create nodes and edges for visualization.
- Each transaction involves two nodes (sender and receiver) connected by an edge.
- Nodes contain information such as address, exchange, time, and value.
- Renders the transaction path using the vis.js library.
- Converts wei to ether using the conversion factor (1 Ether =  $10^{18}$  Wei).

#### 3.3 Get Exchange from Address:

- Determines the associated exchange based on the prefix of a given Ethereum address.
- Returns a string representing the exchange or "Unknown" if no match is found.





## CHAPTER 4

### Fetching Token Balances

- The 'getBalance' function is triggered when the user clicks the "Get Balances" button.
- It retrieves the Ethereum address entered by the user.
- A predefined list of token contracts with their corresponding contract addresses and decimals is provided.
- The code iterates through each token, making API requests to Etherscan using the Ethereum address and the token contract address to retrieve the token balance.
- The total balance is calculated by summing up individual token balances.

Back

### Token Balances

Enter Ethereum address:  Get Balances

Token	Balance
DAI	33.01
USDC	0
USDT	27.719377
BNB	0
WBTC	0.00356335
Total	60.73294035

## CHAPTER 5

### Suspicious Address Detection

- **Sender and Recipient Analysis:**

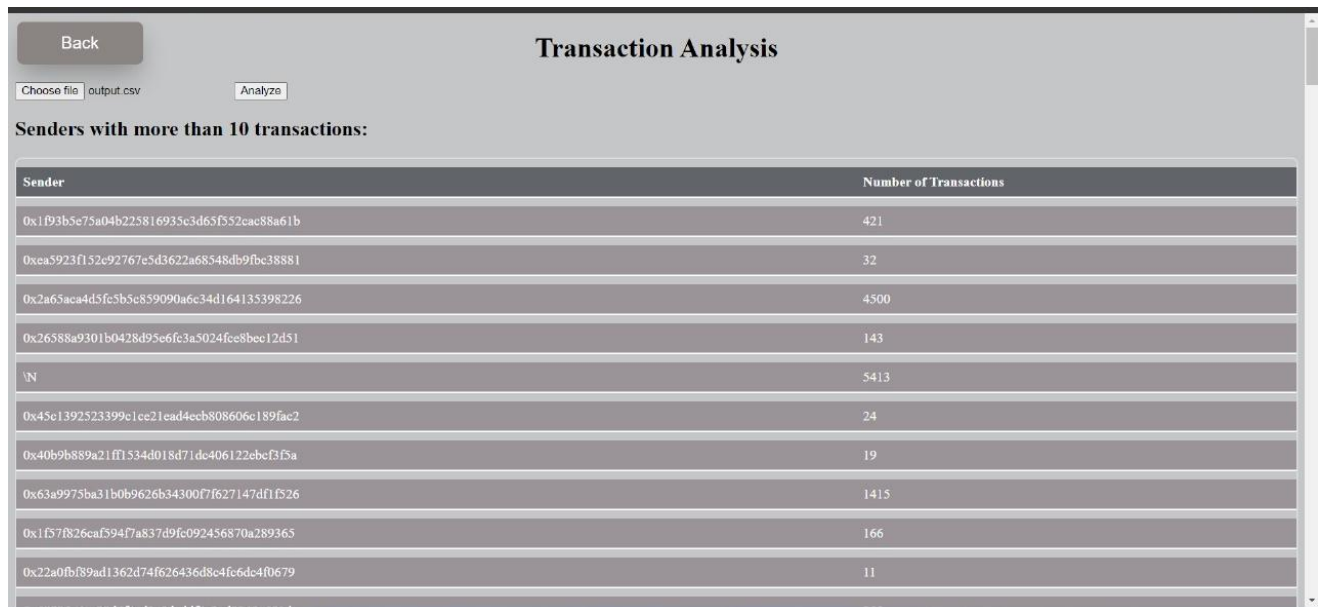
- The code aims to identify senders and recipients involved in a considerable number of transactions.
- Addresses with a high transaction count might be considered suspicious, potentially indicating abnormal or malicious behavior.

- **Threshold Setting:**

- The threshold of 10 transactions is an arbitrary value and can be adjusted based on the context and requirements.
- A higher threshold may help in identifying more significant and potentially more suspicious transaction patterns.

- **Visual Representation:**

- The results are presented in an HTML table, making it visually accessible and providing a clear breakdown of senders and recipients with elevated transaction counts.



The screenshot shows a web application interface for 'Transaction Analysis'. At the top, there is a 'Back' button and a title 'Transaction Analysis'. Below the title, there are two buttons: 'Choose file' and 'output.csv', followed by an 'Analyze' button. The main content area is titled 'Senders with more than 10 transactions:'. Below this title is a table with two columns: 'Sender' and 'Number of Transactions'. The table lists several hexadecimal addresses and their corresponding transaction counts.

Sender	Number of Transactions
0x1f93b5e75a04b225816935c3d65f552eac88a61b	421
0xea3923f152c92767e5d3622a68548db9fbc38881	32
0x2a65aca4d5fc5b5c859090a6c34d164135398226	4500
0x26588a9301b0428d95e6fc3a5024fce8bec12d51	143
\N	5413
0x45e1392523399c1ce21ead4ecb808606c189fac2	24
0x40b9b889a21ff1534d018d71dc406122ebcf3f5a	19
0x63a9975ba31b0b9626b34300f7f627147df1f526	1415
0x1f57f826caf594f7a837d9fc092456870a289365	166
0x22a0fbf89ad1362d74f626436d8c4fc6dc4f0679	11

**Recipients with more than 10 transactions:**

Recipient	Number of Transactions
0x9af09991ad63814e53ffc1bccf213ee74027608b	840
0x3375ec30428b2a71c428afa5e89e427905f95f7e	143
0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5	615
0xc7696b27830dd8aa4823a1cba8440c27c36adec4	24
0x2a65aca4d5fc5b5c859090a6e34d164135398226	2414
0xfbb1b73c4f0bda4f67dca266ce6cf42f520fbb98	1068
0x790b8a3ce86e707ed0ed32bfb9b3269692a23ce1	63
0x1f57f826caf594f7a837d9fc092456870a289365	818
0xf8b483dba2c3b7176a3da549ad41a48bb3121069	593
0x738db714c08b8a32a29e0e68af00215079aa9e5e	168
0x0c729be7c39543c3d549282a40395299d987ccc2	78
0xd34da389374caad1a048fbd4569aac33fd5a375	16

## CHAPTER 6

### Address Clustering:

The plot includes transaction value, gas used, and time, with additional coloring indicating clusters obtained through K-Means clustering.

#### 6.1 Data Processing:

- The **processData** function parses the CSV data into separate arrays for transaction value, gas used, time, and labels.
- Each line of the CSV file is split, and relevant data is extracted assuming a specific column structure.

#### 6.2 Scatter Plot Creation:

- A 3D scatter plot is created using Plotly.js, with transaction value on the x-axis, gas used on the y-axis, and time on the z-axis.
- Data points are color-coded based on labels, representing clusters obtained from K-Means clustering.

#### 6.3 Plotly.js Integration:

- The **Plotly.newPlot** function is used to generate the 3D scatter plot. The layout of the plot is defined, including title and axis labels.

Transaction Value vs. Gas Used vs. Time with K-Means Clusters

