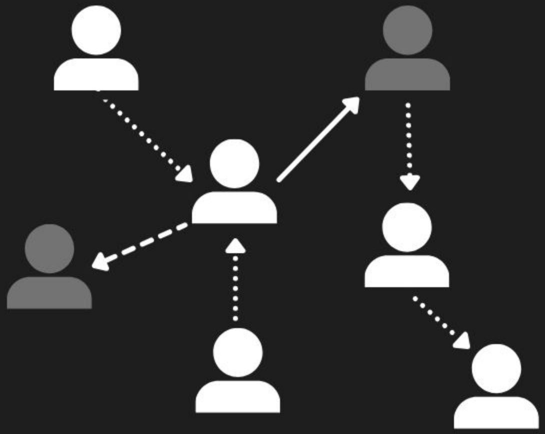
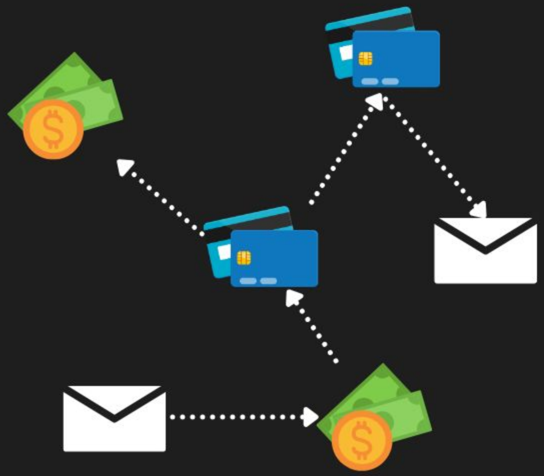
Part 3:

Network Analysis

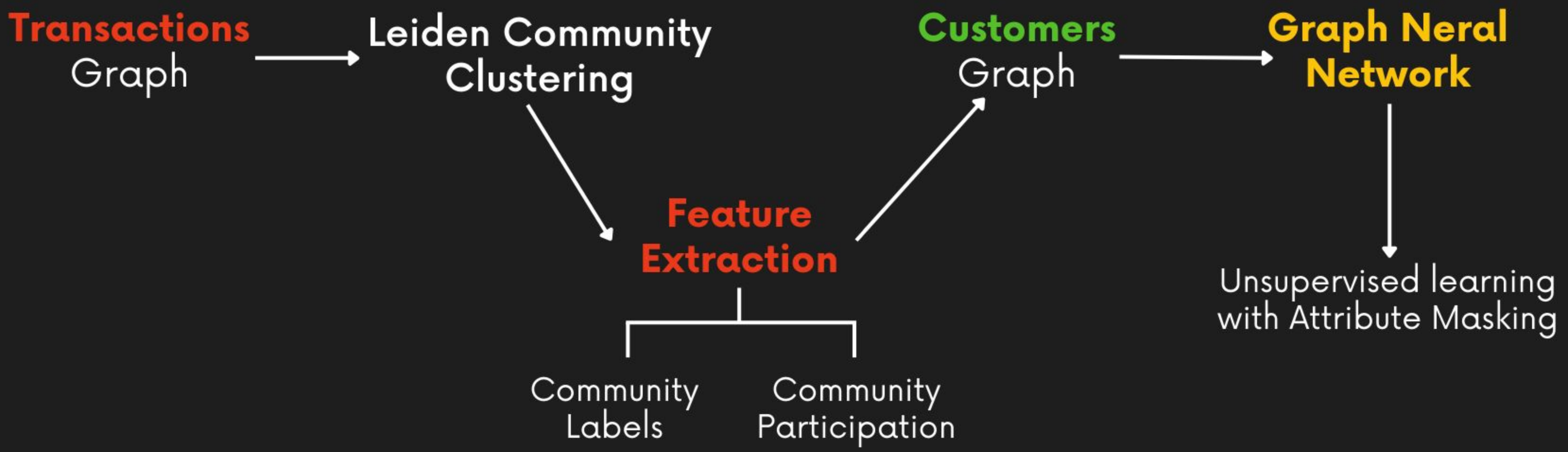


Customers
Graph



Transactions
Graph

Process

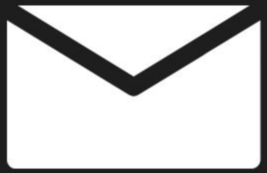

Transactions
Graph
Leiden Community
Clustering
Feature
Extraction
Community
Labels
Community
Participation
Customers
Graph
Graph Neral
Network
Unsupervised learning
with Attribute Masking

Part 3.1:

Transactions Graph

Graph **Nodes**

Transactions



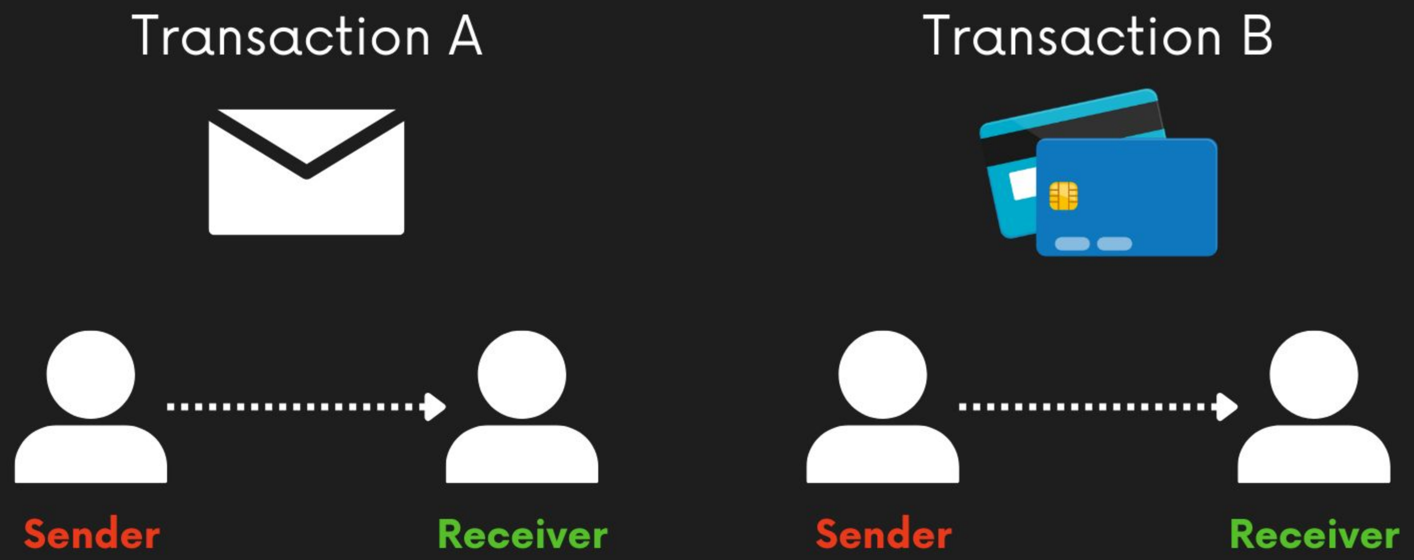
EMT

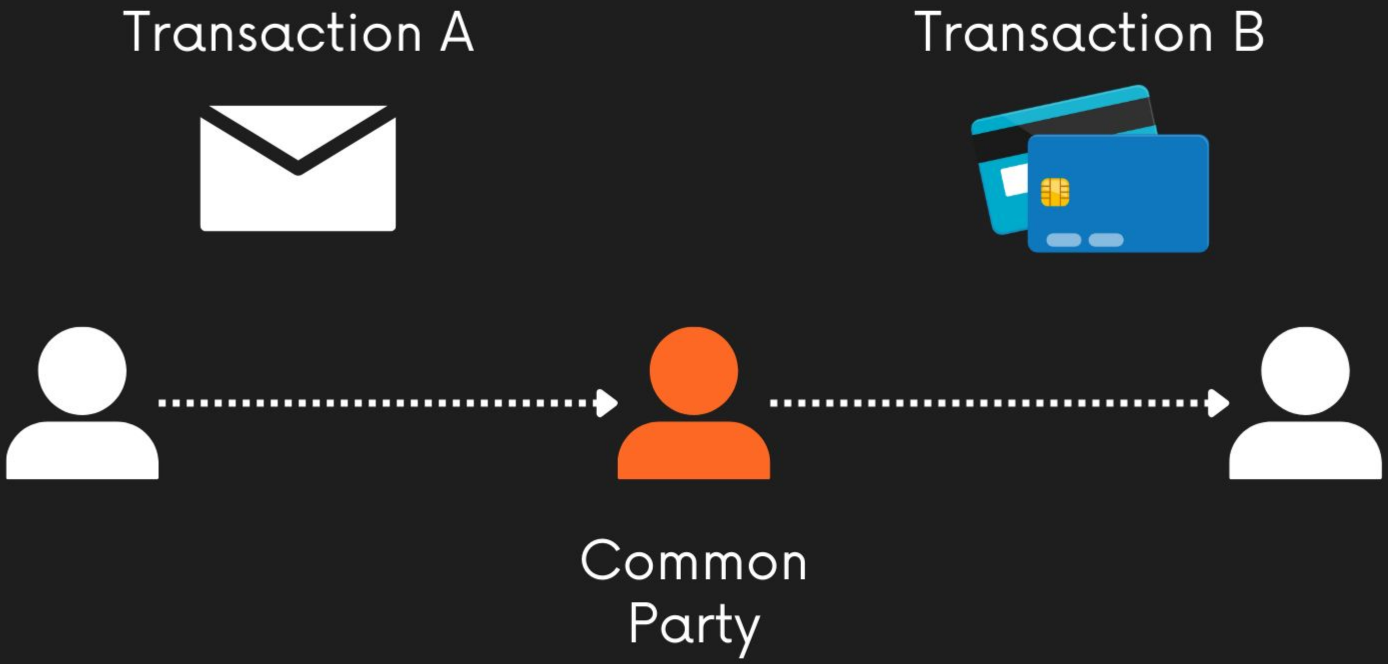
Wire

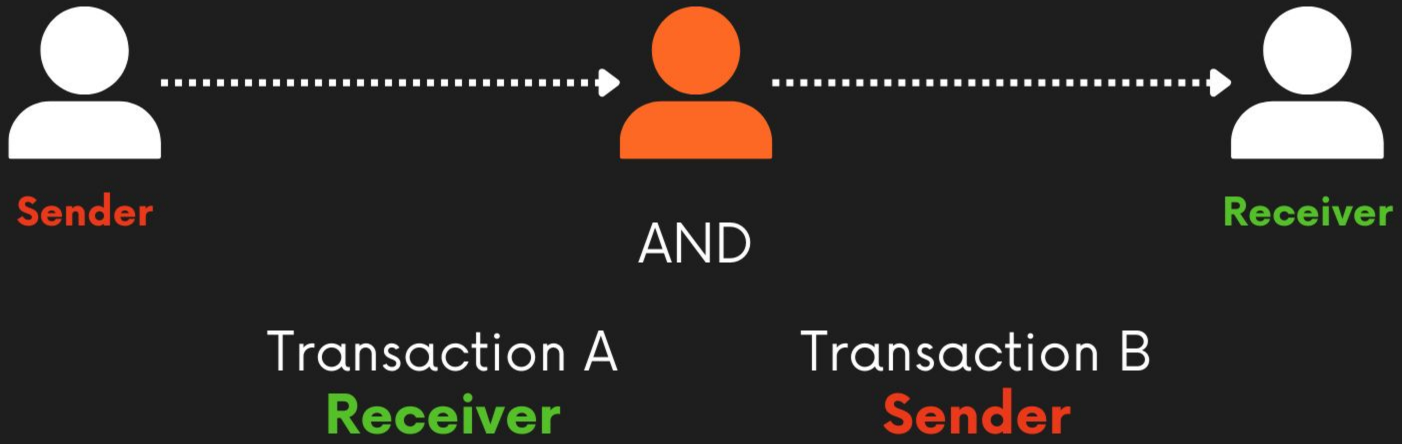
Cash

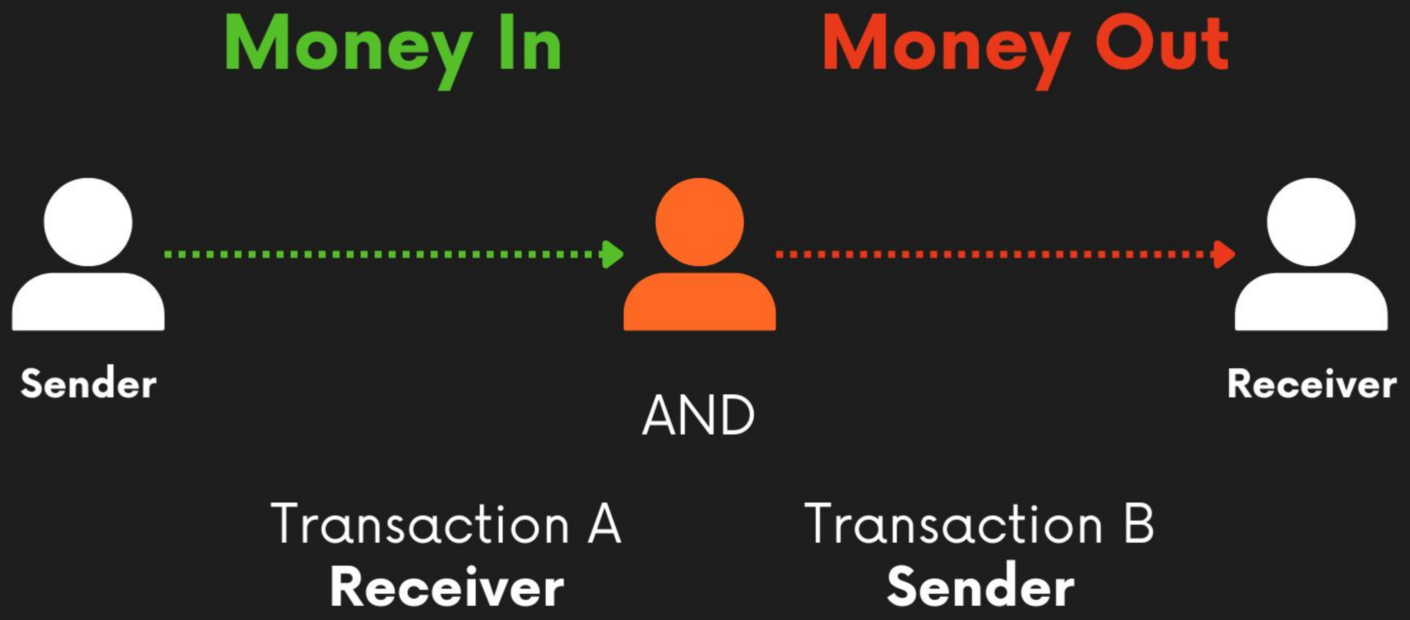
How do you **connect** transactions?

EMT and Wire Transactions


Transaction A
Sender
Receiver
Transaction B
Sender
Receiver


Transaction A
Transaction B
Common
Party


Sender
AND
Receiver
Transaction A
Receiver
Transaction B
Sender


Money In
Money Out
Sender
Receiver
AND
Transaction A
Receiver
Transaction B
Sender

Money In => Money Out

Sender

Receiver

AND

Transaction A
Receiver

Transaction B
Sender

Cash Transactions

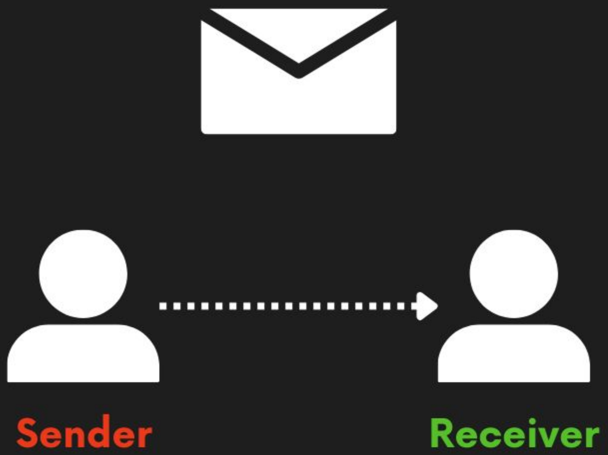
1 Customer

Cash Transactions

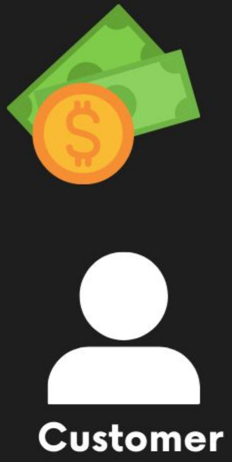
Deposits

Withdrawals

Transaction A



Transaction B

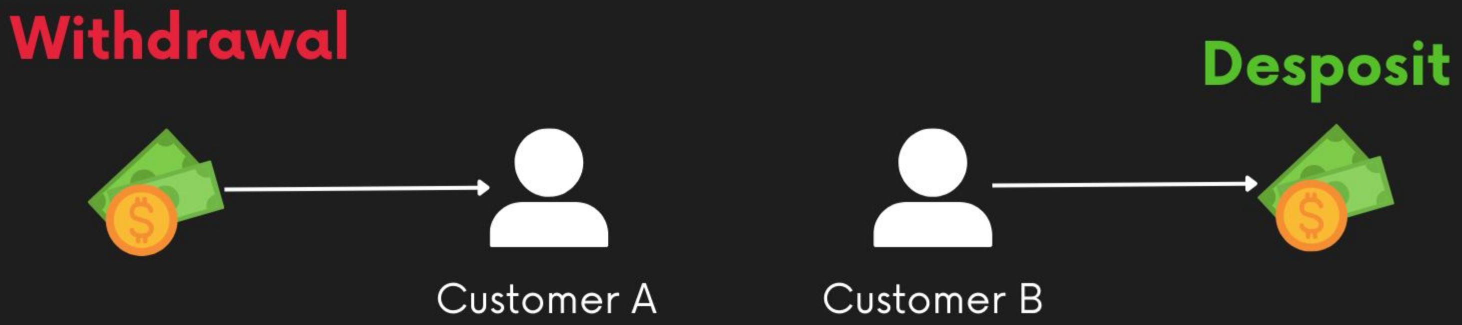


Transaction A
Transaction B
Common
Party

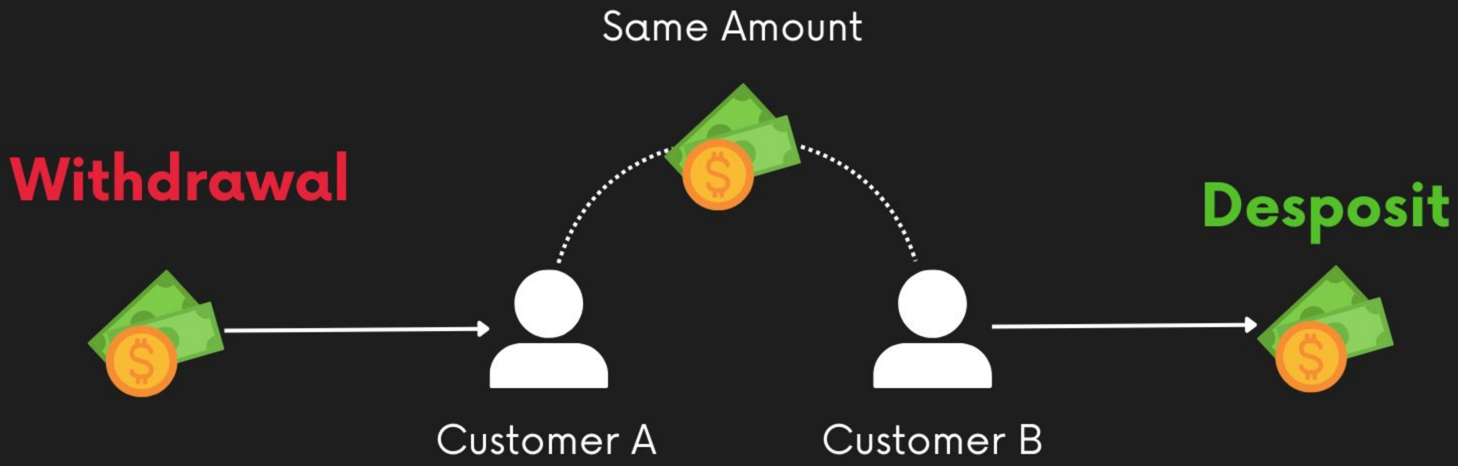
Money Flow


Receive Money
Withdrawal
Sender
Customer Account
Desposit
Send Money
Customer Account
Receiver

Money Flow



Money Flow



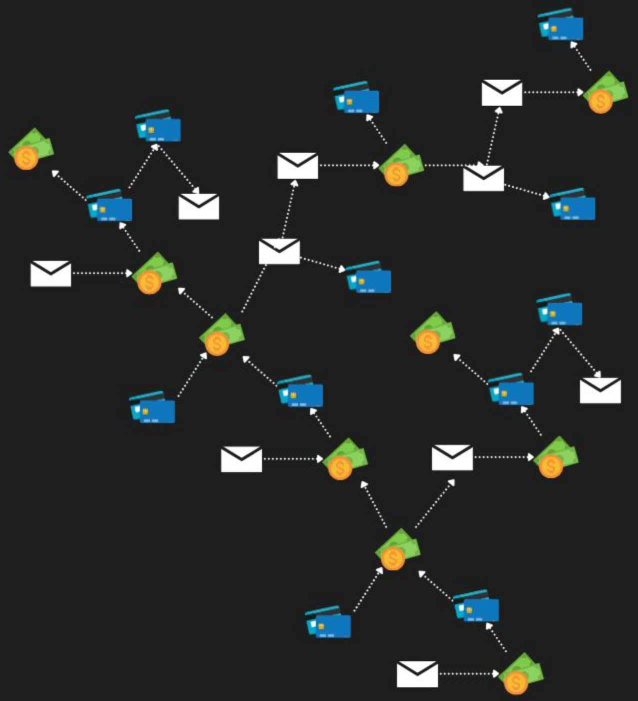
Part 3.2

Community Analysis

Leiden Modularity

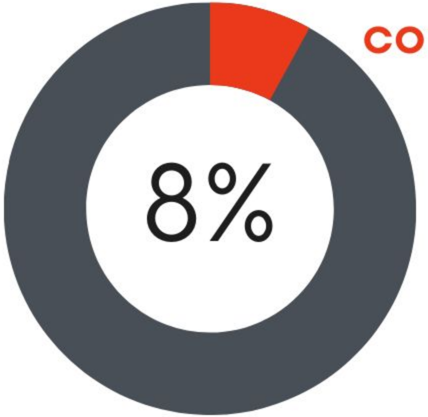
- **Community detection** algorithm
- Compares how **dense** connected nodes are to how connected they would be in a **random network**

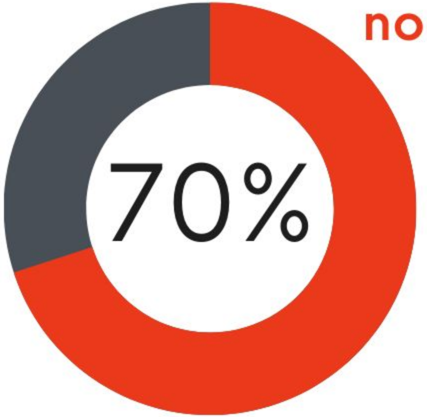
- **Faster + more stable** results vs **Louvain**



Leiden Modularity Communities

- Total Nodes: 457,421
- Total Communities: 213,143





HRIs with Animal-Related Occupation Communities

11

Communities

Community Labels:

[4, 13, 31, 32, 38, 138, 147, 148, 151, 227, 379]

321

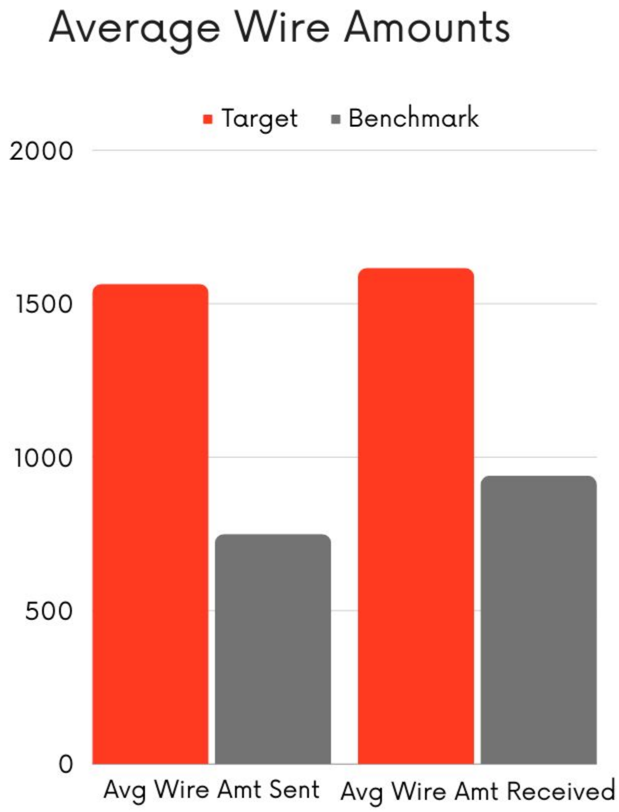
Customers

on average

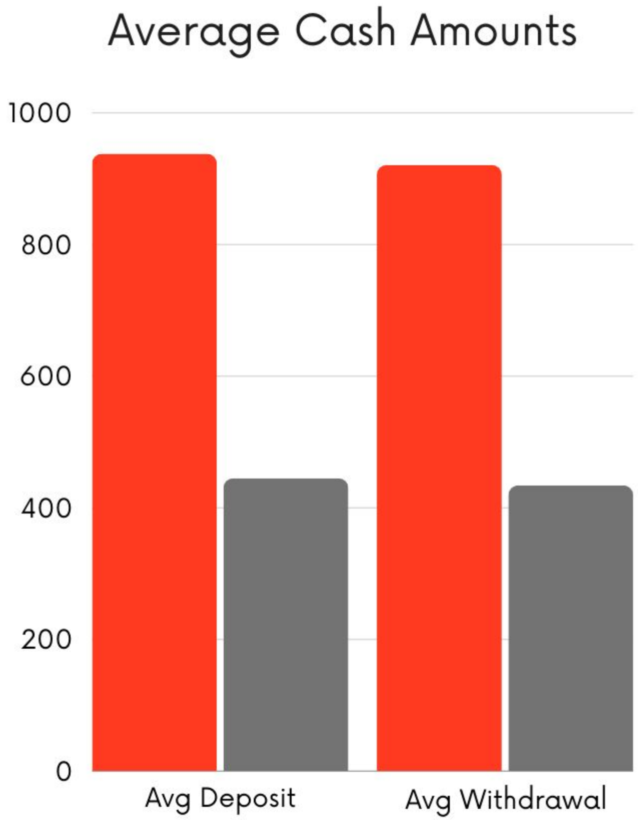
33

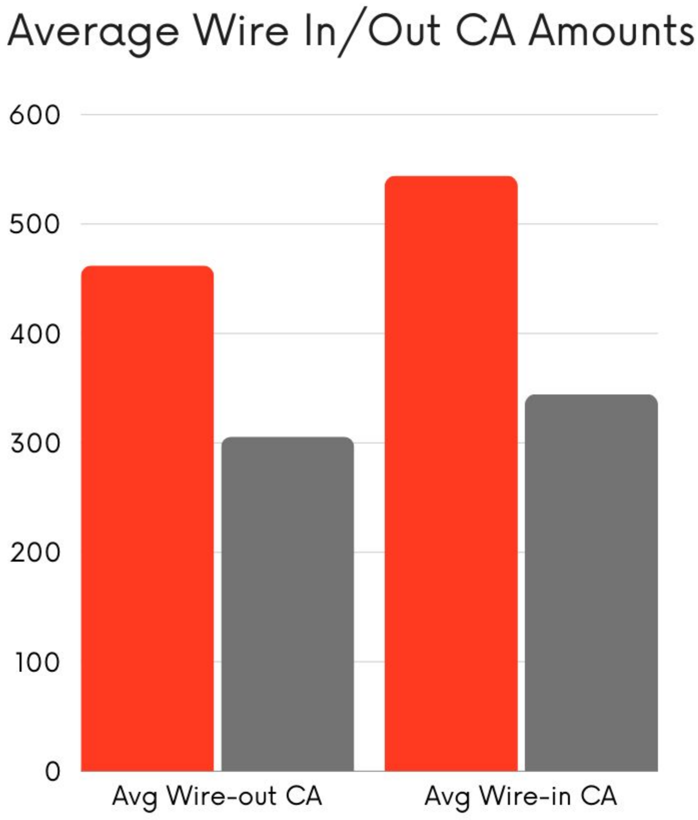
High-Risk Individuals (HRIs)

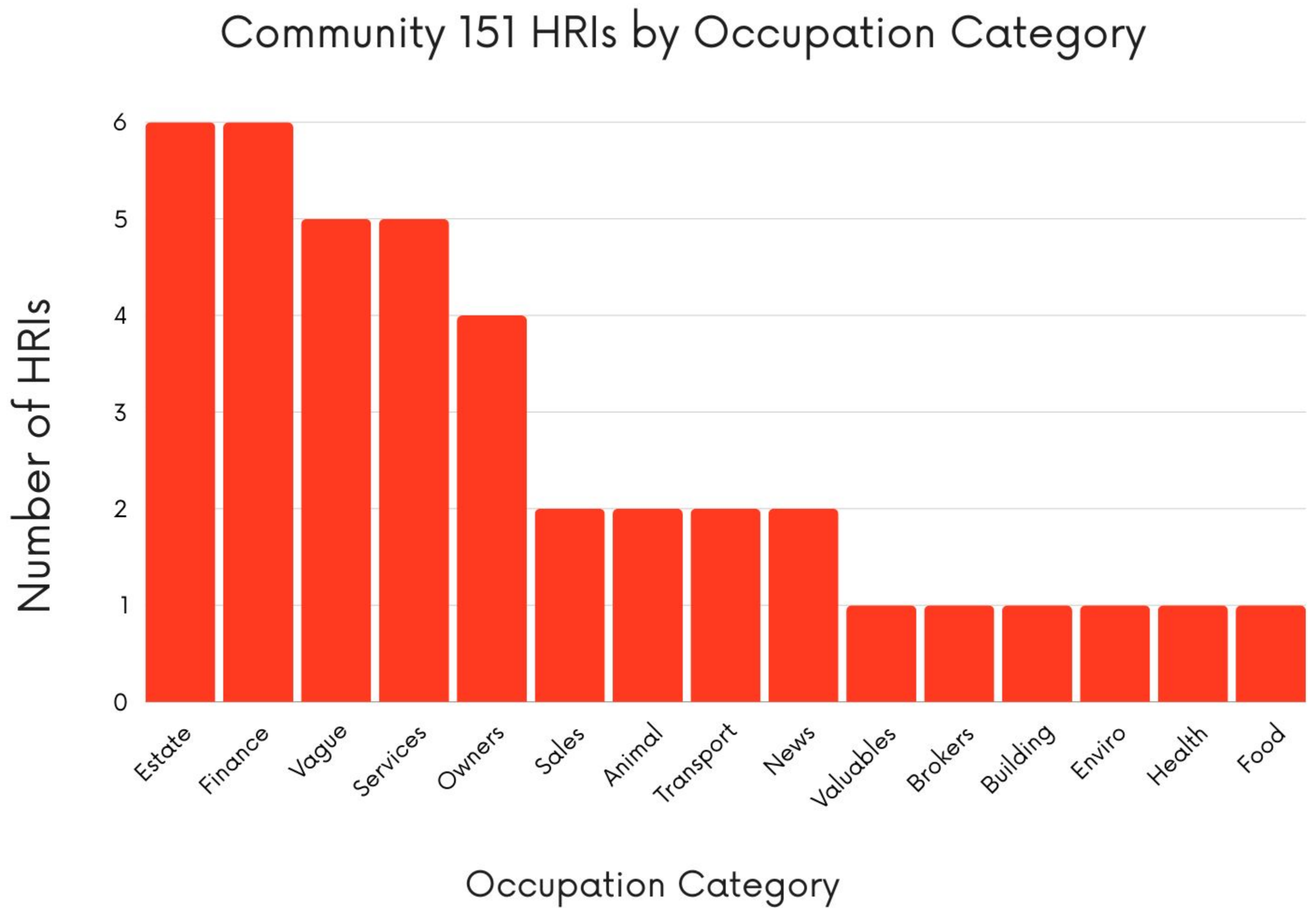
on average


Average Wire Amounts
Target
Benchmark
2000
1500
1000
500
0
Avg Wire Amt Sent
Avg Wire Amt Received


Average EMT Amounts
Target
Benchmark
600
500
400
300
200
100
0
Avg Emt Amt Sent
Avg Emt Amt Received


Average Cash Amounts
1000
800
600
400
200
0
Avg Deposit
Avg Withdrawal


Average Wire In/Out CA Amounts
600
500
400
300
200
100
0
Avg Wire-out CA
Avg Wire-in CA


Community 151 HRIs by Occupation Category
Number of HRIs
6
5
4
3
2
1
0
Estate
Finance
Vague
Services
Owners
Sales
Animal
Transport
News
Valuables
Brokers
Building
Enviro
Health
Food
Occupation Category

Transaction Flow Analysis

Most common edge type:

Cash Deposits - To - EMT

Runner-up:

Wire - To - EMT

Least common edge type:

Cash Withdrawals - To - Cash Deposits

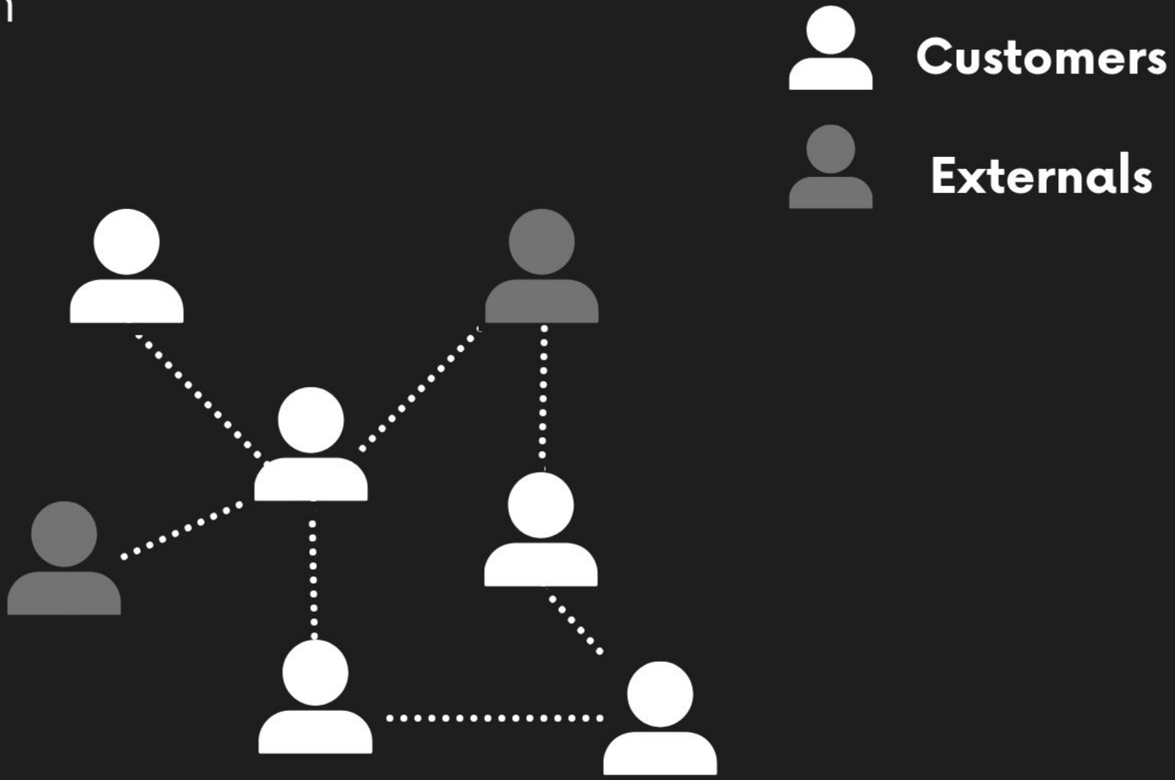
Community 151 Transactions

	Edge Type	Source Counts	Destination Counts
0	wsw_edges	94	93
1	wse_edges	444	442
2	wsc_edges	47	47
3	ese_edges	108	109
4	esc_edges	30	29
5	esw_edges	34	34
6	csc_edges	6	3
7	csw_edges	132	133
8	cse_edges	511	511

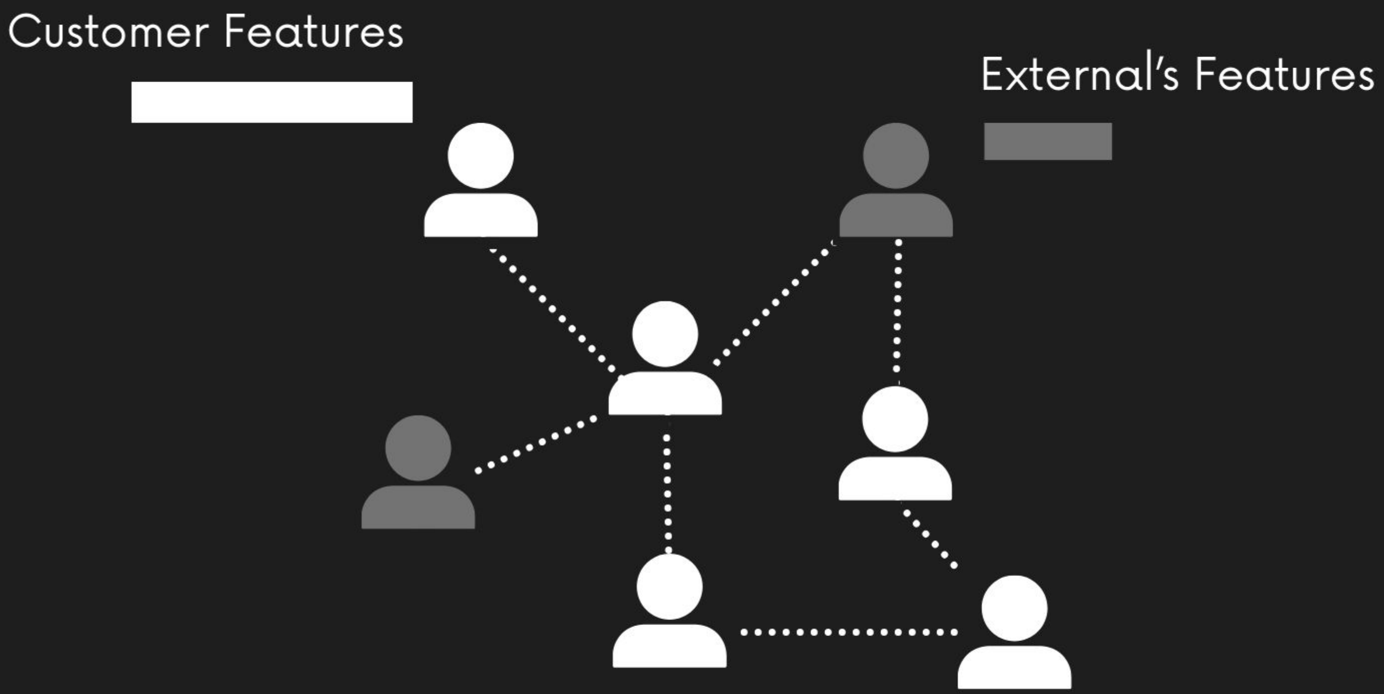
Part 3.3

Customers Graph

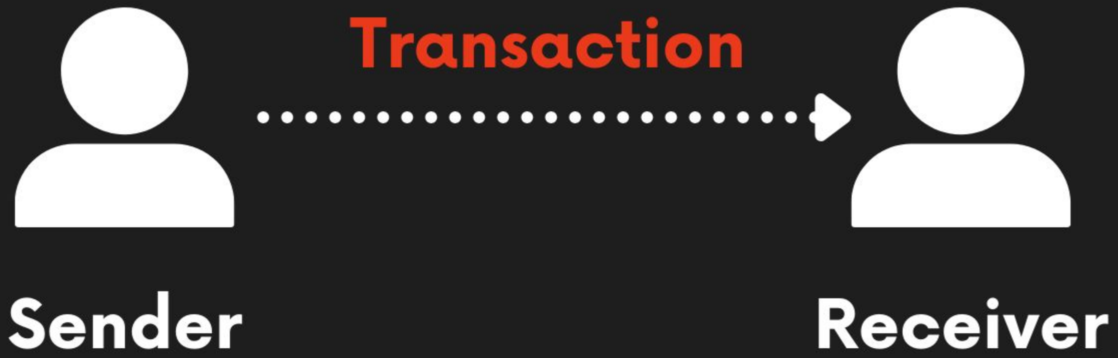
Customers Graph



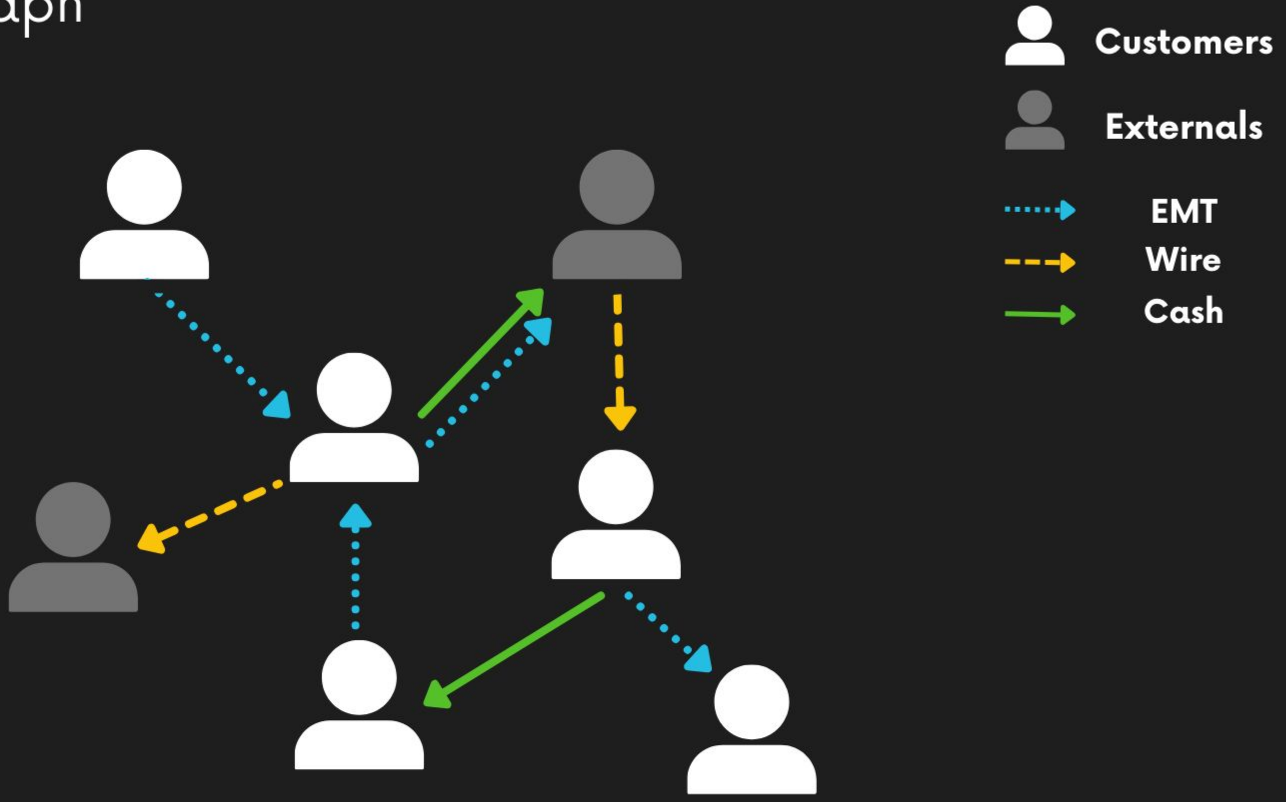
Heterogeneous Graph



Directed Graph



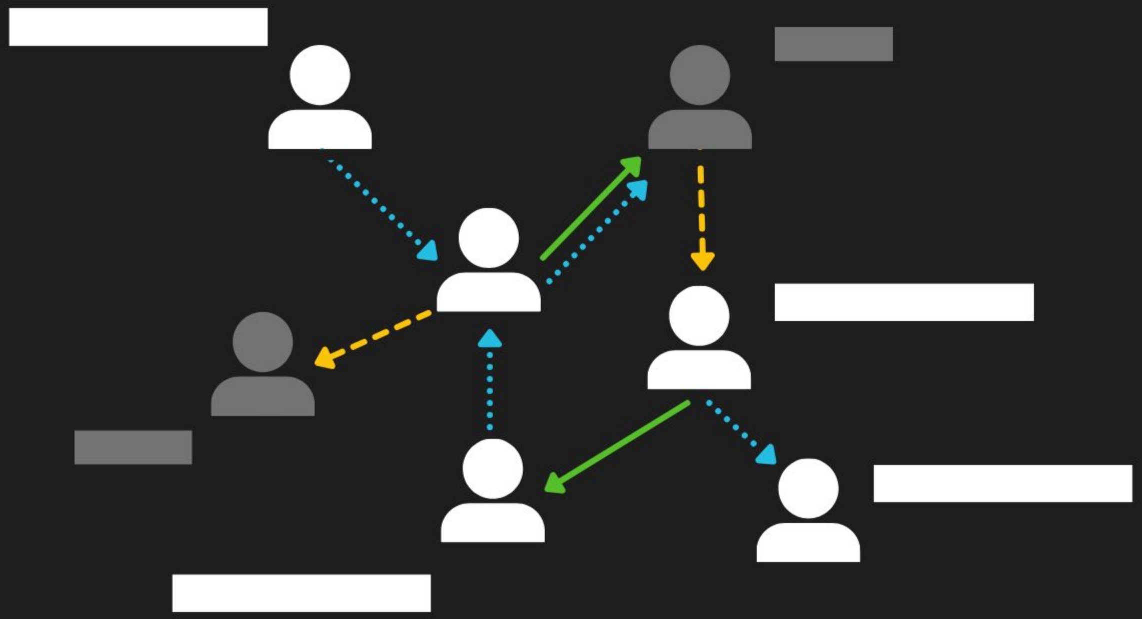
Customers Graph



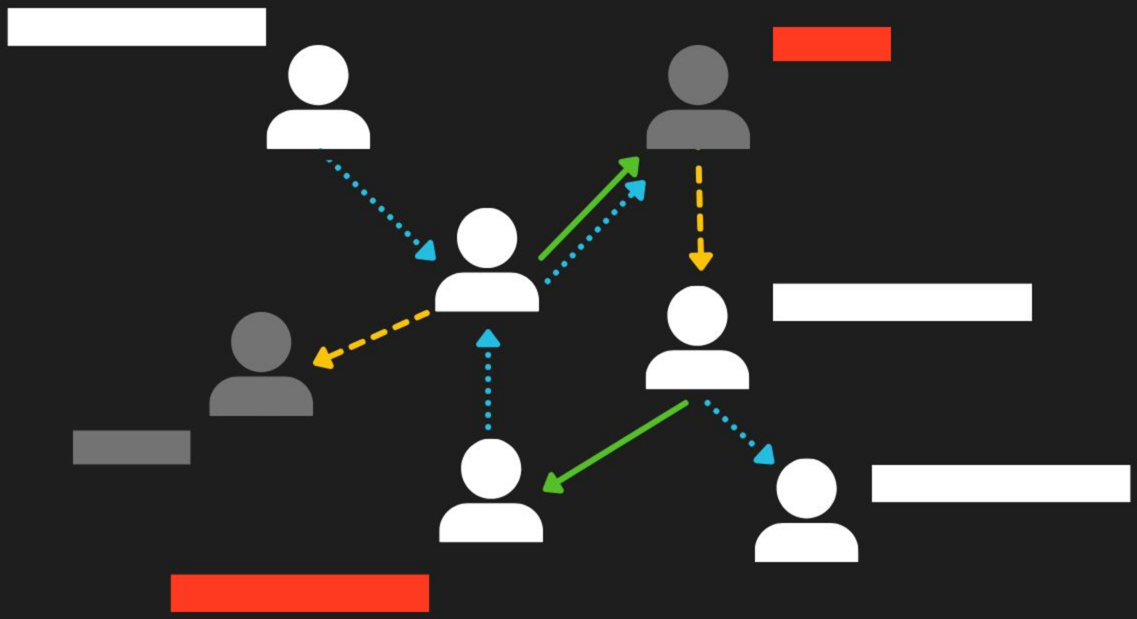
Part 3.4

Unsupervised Learning with GNNs

Unsupervised Learning: Attribute Masking



Unsupervised Learning: Attribute Masking



Selected Model:

HeteroGNN

Capable of taking in graph data with **multiple node and edge types**

Model Inputs:

Customers Graph

Number of Nodes: 196,058
Number of Edges: 563,250

Model Architecture

- Node Feature Transformation

- **2 GNN Layers**
  - HeteroGraphConv Layers
    - **SAGEConv**
      - **agg_type='lstm'**
      - **feat_drop=0.1**
- **Train-test split: 80-20**

In Progress:

- Model tuning + normalization feature layer

Next Steps:

- Incorporating Edge Features
- Testing Other Conv Layers:
    - **EGATConv:** Graph Attention Layer
    - **GINEConv:** Graph Isomorphism Network with Edge Features
    - **HGTConv:** Heterogeneous Graph Transformer convolution

Future Implementations

In addition, we can explore additional changes to our analysis process. Instead of clustering the nodes in the Transactions graph with Leiden Modularity, we can input the graph into a GNN to extract embeddings that can capture more information about the relationships. We can then cluster the nodes based on those embeddings and continue the community analysis and feature extraction for the customers graph. After retrieving the embedded outputs for the GNN that uses the customer graph as input, we can cluster the customer nodes in to subgraphs, then further implement another GNN to produce embeddings for these subgraphs, which we can then further cluster into communities of subgraphs. Once we have these communities, we can then analyze the communities of subgraphs to try and identify a community of illegal wildlife traffickers and those involved in the process.

Thank you for reading!