**Ethereum dataset**

1. Loading and understanding the dataset (attached notebook)
2. Visualization tasks (attached notebook)
3. Calculate and Compare Different Centrality Measures:

* Task: Compute various centrality measures for all nodes in the network and compare their distributions and rankings.
* Steps:
  + Use the networkx library to calculate measures like:
    - Degree Centrality: Measures the number of direct connections a node has.
    - Betweenness Centrality: Measures how often a node lies on the shortest path between other nodes.
    - Closeness Centrality: Measures how quickly a node can spread information through the network.
    - Eigenvector Centrality: Measures the influence of a node based on the centrality of its neighbors .
  + Store these centrality values in a pandas DataFrame or dictionary, associated with the transaction addresses.
  + Visualize the distribution of each centrality measure (e.g., using histograms).
  + Identify the top-k most central nodes based on each measure and compare these lists. Discuss why different measures might identify different nodes as most central.

1. Community detection:

* Task: Apply various community detection algorithms to the network and compare the resulting partitions. This can reveal groups of addresses that belong to the same entity (e.g., a large exchange, a decentralized application, a collection of wallets owned by the same individual or group). It can also help identify tightly-knit groups that might be coordinating activities, potentially including illicit ones.
* Steps:
  + Use libraries like networkx such as:
    - Louvain method: An iterative algorithm that optimizes modularity.
    - Leiden method(optional)
  + Run several of these algorithms on your network.
  + Obtain the community assignments for each node.
* Task: Analyze the Properties of Identified Communities: such as their size, density, and internal structure.
* Steps:
  + Calculate the number of nodes and edges in each community.
  + Compute the density of each community (ratio of existing edges to possible edges).
  + Visualize individual communities or the network with nodes colored by their community membership
  + Analyze the distribution of community sizes.

* Task: Use metrics to assess the quality of the community structures found by different algorithms.
* Steps:
  + Calculate modularity: A widely used metric that measures how well a network is partitioned into communities. Higher modularity generally indicates a better partition.
  + Calculate NMI
  + Calculate ARI
* Task: Discover communities that are predominantly composed of phishing addresses or are heavily involved in transactions with phishing addresses. This can help isolate and analyze the structure of phishing operations within the network. Understanding how phishing addresses cluster and interact can aid in developing more targeted detection and prevention strategies.
* Steps:
  + Apply community detection algorithms and then analyze the composition of the detected communities in terms of the proportion of phishing addresses within them
  + incorporate node attributes (like the phishing labels) during the community detection process.

1. Link Prediction:

Predicting the Existence of Future Transactions:

* Task: Given the current network, predict which pairs of addresses will engage in a transaction in a future time period.
* Steps:
  + Pre-process the dataset into train-test data.
  + Use link prediction methods based on network structure (e.g., common neighbors, Jaccard coefficient, Adamic-Adar index, preferential attachment).
  + Evaluate the performance of your prediction model using appropriate metrics (e.g., Area Under the ROC Curve (AUC), Average Precision (AP)).

1. Graphlets:

* Task: Identify graphlets.
  + Focus on smaller graphlets initially, such as 2-node graphlets (an edge), 3-node graphlets (a path and a triangle), and potentially some 4-node graphlets(using ORCA tool).
* Task: Compute Graphlet Frequency and Distribution. Count the occurrences of different graphlets in your network to understand its local structure.
  + Count the number of times each defined graphlet appears in your network.
  + Calculate the frequency of each graphlet type.
  + Visualize the distribution of graphlet counts.
* Task: Use Graphlet Features for Downstream Tasks:
  + Once you have the graphlet orbit count vectors for each node, use these vectors as features in a machine learning model.
  + For example, train a classifier to predict phishing users (as in node classification) using the graphlet features instead of or in addition to sequence-based features (after you have studied GNNs).
  + Apply clustering algorithms to the nodes using their graphlet feature vectors to see if similar graphlet patterns correspond to functional or structural groupings.

1. GNNs: Classify each address (node) as either "phishing" or "non-phishing."

* Steps:
  + Use the 'FromIsPhi' and 'ToIsPhi' labels to create node-level labels. (code already provided)
  + Split the nodes into training, validation, and test sets.
  + Build a simple GNN model (e.g., a Graph Convolutional Network - GCN) that takes node features and the graph structure as input.
  + Train the GNN model to predict the target variable for each node.
  + Evaluate the model's performance using metrics appropriate for classification.

1. Perform the above classification task using:
   * Graph Attention Networks
   * Graph Transformers