TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Lucky Chigozie Ogogo 245206IVSM

COMPARATIVE ANALYSIS OF GRAPH-BASED METHODS FOR DETECTING COORDINATED CRYPTOCURRENCY SCAM NETWORKS ON TWITTER/X

Master's thesis

Supervisor: Dimitrios Symeonidis

MSc.

Gert Kanter

PhD

Tallinn 2026

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Lucky Chigozie Ogogo 245206IVSM

GRAAFIPÕHISTE MEETODITE VÕRDLEV ANALÜÜS KOORDINEERITUD KRÜPTORAHAPETTUSTE VÕRGUSTIKE AVASTAMISEKS TWITTERIS/X-IS

Magistritöö

Juhendaja: Dimitrios Symeonidis

MSc.

Gert Kanter

PhD.

Tallinn 2026

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Lucky Chigozie Ogogo

Date: 18.05.2026

Abstract

Cryptocurrency scams on Twitter/X are rarely the work of one account. Fake giveaways, impersonation scams, and pump-and-dump schemes usually depend on a small group of accounts that mention, retweet, and follow each other to manufacture an impression of legitimacy. This thesis asks whether the relational structure that such coordination leaves on the platform can be exploited by graph-based detection methods, and whether those methods improve on a strong content-only baseline.

Using the TwiBot-22 benchmark, the study runs a controlled comparison of nine method configurations grouped into four families: a Random Forest baseline trained on eleven behavioural tweet features; Louvain community detection; Node2Vec random-walk embeddings; and Simple Graph Convolution (SGC) as a lightweight graph neural network surrogate. All methods share the same 5,000-user stratified sample, the same feature set, and the same five-fold stratified cross-validation protocol. A calibrated synthetic graph and a small two-phase cryptocurrency-scam case study are added to isolate graph density as a confounding variable.

On the real subgraph the nine methods compress into a narrow F1 band of 0.480 to 0.503 and the non-graph baseline holds the best AUC-ROC of 0.691. On the denser synthetic graph SGC reaches F1 = 0.945 and the all-features fusion reaches F1 = 0.950, while the feature-only baseline drops to F1 = 0.460. The reversal is decisive: graph methods help when there is enough graph to work with and provide essentially no lift below a density threshold somewhere between 1.25×10^{-5} and 7.23×10^{-4}. SGC is the most resilient method across both regimes because its self-loop term means isolated nodes degrade gracefully to the content-only baseline. This study argues that the sampling strategy, not the algorithm choice, is the binding constraint for graph-based scam detection on large social platforms.

Lühikokkuvõte

Graafipõhiste meetodite võrdlev analüüs koordineeritud krüptorahapettuste võrgustike avastamiseks Twitteris/X-is

Krüptorahapettused Twitteris/X-is on harva ühe konto töö. Võltsitud meened, kehastuspetuskeemid ja pump-ja-pump skeemid sõltuvad tavaliselt väikesest kontode grupist, mis mainivad, retweebivad ja järgivad üksteist, et valmistada legitiimsuse muljet. Käesolevas lõputöös küsitakse, kas sellist relatsioonilist struktuuri, mida selline koordinatsioon platvormile jätab, on võimalik graafikupõhiste tuvastusmeetoditega ära kasutada ning kas need meetodid paranevad ainult tugeval sisupõhisel lähtetasemel.

Kasutades TwiBot-22 võrdlusalust, toimub uuringus üheksa meetodi konfiguratsiooni kontrollitud võrdlus, mis on grupeeritud nelja perekonda: Random Forest algtase, mis on treenitud üheteistkümnele käitumuslikule säutsule; Louvain kogukonna tuvastamine; Node2Vec random-walk embeddings; ja Simple Graph Convolution (SGC) kui kerge graafik Neural Network surrogaat. Kõik meetodid jagavad sama 5000-kasutaja kihistunud proovi, sama funktsioonikomplekti ja sama viiekordset kihistunud ristvalidatsiooniprotokolli. Graafiku tiheduse isoleerimiseks lisatakse segava muutujana kalibreeritud sünteetiline graafik ja väike kahefaasiline krüptoraha-kelmuse juhtumiuuring.

Reaalsel alamgraafil surutakse üheksa meetodit kokku kitsaks F1 ribaks 0,480 kuni 0,503 ja mittegraafilisel lähtetasemel on parim AUC-ROC 0,691. Tihedamal sünteetilisel graafikul jõuab SGC F1 = 0,945-ni ja kõigi funktsioonide fusioonini F1 = 0,950, samas kui ainult featuring langeb F1 = 0,460-ni. Ümberpööramine on määrav: graafikumeetodid aitavad, kui on piisavalt graafikut, millega töötada, ja ei paku sisuliselt ühtegi tõstet alla tihedusläve kusagil 1.25 × 105 kuni 7.23 × 104. SGC on mõlema režiimi kõige vastupidavam meetod, sest selle enesekehtestamise termin tähendab isoleeritud sõlmede graatsilist degradeerumist ainult sisupõhiseks lähtealuseks. Lõputöös väidetakse, et proovivõtustrateegia, mitte algoritmivalik, on graafikupõhise petuskeemi tuvastamise siduvaks piiranguks suurtel sotsiaalsetel platvormidel.

List of abbreviations and terms

API	Application Programming Interface
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
CIB	Coordinated Inauthentic Behaviour
CV	Cross-Validation
DFS	Depth-First Search
BFS	Breadth-First Search
F1	F1-score, the harmonic mean of precision and recall
GCN	Graph Convolutional Network
GAT	Graph Attention Network
GNN	Graph Neural Network
LR	Logistic Regression
PR	PageRank
SGC	Simple Graph Convolution
TalTech	Tallinn University of Technology
TwiBot-22	Twitter Bot benchmark dataset released in 2022
VGAE	Variational Graph Auto-Encoder

Table of contents

List of figures

List of tables

1 Introduction

Cryptocurrency scams on Twitter/X have grown into a persistent and well-documented category of online financial fraud. Fake giveaways promising to double the amount of cryptocurrency sent to an address, impersonations of well-known exchanges or founders, and coordinated pump-and-dump campaigns are typically not the work of a single visibly malicious account. They depend on small networks of accounts that mention each other, retweet the same misleading message and follow one another to create the impression of an organic community. From a detection perspective, this matters a great deal. A single deceptive post is easy to flag; a coordinated cluster of accounts using slightly different wording, posting on overlapping schedules and redirecting traffic through shared links is much harder to catch with content rules applied account by account.

Graph-based methods are an obvious candidate for this kind of detection problem. If the coordination is real, it should leave a structural footprint in the follow, retweet and mention graph that no individual post can fully reveal. Community detection, random-walk embeddings and graph neural networks are designed to surface exactly those structural patterns. They have produced strong results in financial fraud and anti-money laundering settings, where the underlying graph encodes a deterministic, auditable relationship: a financial transfer happened, or it did not. Social interaction graphs are different. A follow may reflect interest, mutual courtesy, recommendation logic or automated inflation; a retweet may be agreement, habit or algorithmic surfacing. Whether graph methods built for the deterministic case still work on the noisier social case is the open question that motivates this thesis.

1.1 Research problem

The detection of coordinated cryptocurrency scam networks on Twitter/X remains unreliable when the suspicious activity is distributed across multiple loosely connected accounts rather than concentrated in a single obvious profile. Existing studies show that Crypto Twitter contains visible bot-like structures and that individual campaigns can be traced through interactions, retweets, mentions and repeated phrasing. What is still missing is a direct, controlled comparison of graph-based methods for this specific task, conducted on the same data, with the same features and the same evaluation protocol, so

that performance differences can be attributed to method choice rather than experimental variation.

1.2 Aim and objectives

The aim of this thesis is to compare graph-based methods for detecting coordinated cryptocurrency scam networks on Twitter/X and to determine which approaches perform best on a shared dataset under controlled conditions. The objectives are:

- to build an interaction graph from Twitter/X data that reflects coordinated cryptocurrency-scam activity.
- to identify and select established graph-based detection methods that are suitable for analysing coordination in social network data.
- to evaluate and compare the selected methods using a common set of performance metrics on the same dataset.
- to determine which graph-based methods are most effective for the task and under what structural conditions they remain effective.

1.3 Research questions

The work is organised around three research questions.

- RQ1. Which graph-based methods are most effective for detecting coordinated cryptocurrency scam networks on Twitter/X?
- RQ2. How do different graph-based methods compare in terms of performance when applied to the same Twitter/X interaction dataset?
- RQ3. Which structural graph properties are most informative for distinguishing coordinated scam accounts from ordinary ones?

1.4 Scope and structure

The scope is deliberately narrow. The study focuses on coordinated scam behaviour on Twitter/X, uses the TwiBot-22 dataset as its primary source of labelled accounts and compares nine method configurations grouped into four families. Detection of non-

coordinated fraud, off-platform scam payment networks and cross-platform campaigns are outside the scope.

The remainder of the thesis is structured as follows. Chapter 2 reviews the literature on cryptocurrency scams on social media, coordinated inauthentic behaviour, social-network graph representation and the three graph-based method families compared here. Chapter 3 describes the data, preprocessing pipeline, graph construction and the comparative experimental design. Chapter 4 presents the results of the nine-method comparison together with a calibrated synthetic baseline and a two-phase cryptocurrency-scam case study. Chapter 5 summarises the contribution, discusses limitations and outlines directions for future work. References and appendices follow.

2 Background, definitions and history

This chapter sets out the conceptual ground on which the rest of the thesis stands. Four threads run through it: how cryptocurrency scams behave on social media, what is meant by coordinated inauthentic behaviour, how social interaction graphs differ from financial-transaction graphs, and what the main families of graph-based detection methods can and cannot see. The aim is not a complete survey of the field but a focused argument that justifies the choices made later in the experimental design.

2.1 Cryptocurrency scams on social media

The natural instinct is to fight cryptocurrency scams the same way one fights spam: scan posts for suspicious keywords, remove the content and ban the account. That instinct produces visible wins on isolated cases and fails on the cases that matter. A single fake giveaway post is easy to identify in hindsight. A campaign in which one account stages the lure, several others amplify it, and a different cluster redirects victims to a phishing page is something that account-level content moderation cannot follow. The cost is not theoretical. Cola et al. (2024) traced a single 2020 Twitter campaign in which approximately 140 accounts cooperated to promote a fake Uniswap giveaway and route victims to Ethereum addresses controlled by the scammers. Each individual post, read on its own, would have looked unremarkable.

Liu et al. (2024) followed up that line of work by combining tweets, livestream recordings, landing-page snapshots and on-chain transactions for a different family of crypto-giveaway scams. They found that even very low conversion rates produced substantial profits and that Twitter was the dominant entry point, with hashtag use as the main amplification vehicle. The takeaway for detection is structural: the same underlying campaign expresses itself through many different surface messages, while the network of who-amplifies-whom stays stable. That is exactly the kind of pattern a graph can see and individual content filters cannot.

2.2 Coordinated inauthentic behaviour

Coordinated inauthentic behaviour is not the same thing as automation. A bot may repeat itself in obvious ways; a coordinated group converges on the same targets at the same time using accounts that are structurally linked. The distinction is important because it shifts the detection problem from the individual to the group. A bot can be flagged on its own; a coordinated cluster must be detected together, because the suspicious property is the joint behaviour rather than any single account's signal.

The traces that coordination leaves on a social platform are measurable even when the content itself is unremarkable. Temporal co-activity, in which several accounts act within tight windows around the same trigger, is one. Unusually dense mutual following among the participating accounts is another. So is retweeting convergence, where the same source or content sequence is repeatedly amplified by the same set of accounts. These signals are properties of the network rather than of the posts, and they are what justifies treating the problem as a network-analysis problem rather than a classification problem over independent rows.

2.3 Graph theory and social-network representation

The graph methods used in this thesis were originally developed and evaluated on graphs whose edges carry an unambiguous meaning. In a financial-transaction graph an edge between two nodes records that a transfer occurred a specific amount moved between specific actors and the record is auditable. Community detection, centrality ranking and graph-neural-network propagation were validated against that kind of graph (Newman, 2010; Kurshan & Shen, 2021). A dense cluster of accounts in a transaction network really does indicate repeated, deliberate financial contact. A high-centrality node really did sit on the path of real money.

Twitter/X interaction graphs do not behave the same way. A retweet may mean agreement, automated boosting, organic interest or algorithmic surfacing. A follow may reflect attention, polite reciprocation, paid inflation or a years-old habit. There is no external ledger that resolves which meaning applies in any individual case. The same edge that a method treats as a coordination signal may simply be the footprint of recommendation logic. Kang et al. (2024) showed that reply and quote edges carry

stronger evidence of intentional interaction than follows alone, which is useful, but it also confirms that the choice of which interactions to encode as edges shapes everything downstream. That choice is theoretical, not technical, and it sits at the centre of this thesis.

2.4 Graph-based anomaly detection and method families

Graph-based anomaly detection is the framework that formalises the task. Akoglu, Tong and Koutra (2015) define it as the discovery of unusual nodes, edges or subgraphs in relational data anomalies that emerge from structure, density or unusual connectivity rather than from individual records. They distinguish three anomaly levels: single nodes, pairs of nodes and groups or communities. This thesis is mainly concerned with the third level, because coordinated scam activity is by definition a group property. A single suspicious account is worth noting, but the real detection target is the cluster around it.

The practical implication is that methods that operate at the group or neighbourhood level community detection, embeddings that encode structural role, and graph neural networks that aggregate neighbourhood information are more theoretically aligned with the task than methods that focus on isolated node properties. That alignment, or its absence, is the lens through which the results in Chapter 4 should be read.

2.5 Graph-based method families for fraud and scam detection

The methods reviewed below fall into three families: community detection and ranking, graph embeddings, and graph neural networks. Each family rests on a different assumption about what the graph encodes. Table 1 in the appendix-style table below summarises the verdict for each method as a Twitter/X scam detector; the discussion that follows explains the reasoning.

2.5.1 Community detection and ranking

Louvain (Blondel et al., 2008) finds communities by greedily maximising modularity, a measure of how much denser within-community connections are than chance. The assumption works well on financial graphs, where fraud rings really do form dense clusters because they have to move funds repeatedly through the same intermediaries. On Twitter/X the assumption is weaker. A dense Louvain community is evidence of

coordination, not confirmation of fraud; fan groups, topical clusters and regional communities can look structurally identical. Louvain is most useful here as a hypothesis generator that flags candidates for further inspection.

PageRank (Brin & Page, 1998) takes a different angle. It assigns importance by propagating influence through the graph and has been used in anti-money-laundering research to surface accounts that act as hubs. On Twitter/X the score reflects platform amplification as much as deliberate coordination. An account can rank highly because it is genuinely popular, because the recommendation algorithm amplified it or because it has been on the platform for years. PageRank is therefore treated here as a prioritisation signal rather than a classifier.

2.5.2 Embedding methods

Node2Vec (Grover & Leskovec, 2016) learns a low-dimensional vector for each node by running biased random walks across the graph and feeding the resulting sequences to a skip-gram model (Mikolov et al., 2013). The two parameters that control the walks, p (return) and q (in-out), determine whether the embeddings capture community membership or structural role. The choice matters for this task. Coordinated scam accounts may operate as a single dense cluster, but they may also be split across sub-networks that target different audiences while playing the same role. A homophily-biased configuration will miss the second pattern; a structural-equivalence configuration will catch it. The literature reports strong results when Node2Vec is paired with a downstream classifier on transaction graphs (Weber et al., 2019), but this performance has not been confirmed on Twitter/X-style graphs.

2.5.3 Graph neural networks

Graph convolutional networks (GCN; Kipf & Welling, 2017) update each node's representation by aggregating normalised information from its neighbours. The approach is powerful on transaction graphs but has a known weakness on social graphs: fixed-weight aggregation treats every edge as equally informative, so noisy edges propagate noise rather than signal. GraphSAGE (Hamilton et al., 2017) adds an inductive aggregation step that lets the model generalise to unseen nodes, which is valuable on Twitter/X where accounts come and go quickly. Variational graph auto-encoders (Kipf & Welling, 2016) reconstruct the graph without supervised labels, which is useful when

confirmed scam annotations are scarce, but their outputs are not directly comparable to supervised scores. Graph attention networks (GAT; Veličković et al., 2018) learn attention weights over neighbours and are the strongest theoretical fit for noisy social graphs, at the cost of higher computational complexity and greater hyperparameter sensitivity.

Simple Graph Convolution (SGC; Wu et al., 2019) removes the non-linearities of a k-layer GCN and reduces it to a single sparse matrix multiplication. The result is a method that is essentially free to compute, runs without GPU support and is reproducible on a laptop. Wu et al. showed that SGC matches GCN performance on several benchmarks, which makes it a natural choice as a lightweight GNN surrogate for a comparative study of this kind.

2.5.4 Summary verdict

Table 1 below distils the discussion into a verdict per method for the specific task of detecting coordinated cryptocurrency scams on Twitter/X. The verdicts are not neutral; they are the output of the edge-semantics argument applied to each method's design assumptions.

Table 1. Graph-based methods reviewed in the literature.

Method	Family	Verdict for Twitter/X scam detection
Louvain	Community detection	Useful as a hypothesis generator; cannot distinguish scam rings from topical or fan clusters.
PageRank	Ranking	Useful as a triage signal; high centrality is not by itself diagnostic of fraud on a platform-mediated graph.
Node2Vec	Random walk embedding	Strong fit when configured for structural equivalence; sensitive to p and q and to graph density.
GraphSAGE	Inductive GNN	Practical for fast-moving scam networks; aggregation still assumes neighbours carry usable signal.
VGAE	Unsupervised GNN	Useful when labels are scarce; output is not directly comparable to supervised methods.
GCN	Convolutional GNN	Theoretically weakest fit; fixed-weight aggregation spreads noise on edge-ambiguous graphs.
GAT	Attention-based GNN	Strongest theoretical fit; attention can down-weight noisy edges, but requires sufficient density to train.
SGC	GNN surrogate	Practical lightweight stand-in for a GCN; identity term gives graceful fallback on isolated nodes.

2.6 Datasets and ground truth

Labelled data is the foundation of any supervised detector, and in this domain, it is also the hardest thing to obtain. Scam behaviour is hidden by design, a single campaign typically spans several accounts and sometimes several platforms, and confirmed labels usually require triangulation across sources rather than a single clean pipeline. The studies that build their own ground truth, for example, Cola et al. (2024) and Liu et al. (2024), produce high-quality but expensive datasets that are not easy to reproduce. The alternative, used here, is to rely on TwiBot-22 (Feng et al., 2022), the largest public Twitter bot benchmark, while being explicit that its labels are bot-vs-human and not scam-vs-legitimate. The two-phase case study in Chapter 3 is designed to partially address that gap by isolating a scam-adjacent subset of the labelled bot population.

2.7 Evaluation metrics in graph-based detection

Class imbalance is the defining feature of scam-detection data and it determines which metrics are meaningful. With bots making up roughly 14 percent of TwiBot-22 and only 7.5 percent of the working sample used here, accuracy is not informative. A classifier that predicts the majority class always achieves over 90 percent accuracy without detecting anything useful. Standard practice in the fraud detection literature is to report precision, recall, F1 and AUC-ROC. Macro-F1 is the primary metric used in this thesis because it weights the bot and human classes equally and therefore exposes the detector's behaviour on the minority class. AUC-ROC is reported as a secondary, threshold-independent measure that summarises the quality of the underlying probability ranking. Together they capture the trade-off between detection completeness and false-positive cost.

2.8 Comparison of methods and the research gap

The literature surveyed in this chapter supports two claims. First, graph-based methods are genuinely useful for fraud detection, fake-account identification and coordinated-behaviour analysis in a range of settings. Second, that usefulness has mostly been demonstrated on graphs whose edges encode an unambiguous relationship a financial transfer, a blockchain transaction, a clearing-house event. Whether the same methods retain their detection advantage on socially contingent, platform-mediated Twitter/X

interaction graphs has not been tested with the rigour this task requires. That is the gap this thesis addresses, both empirically (no controlled comparison of community, embedding and GNN-surrogate methods has been run on the same Twitter/X scam interaction data under equivalent conditions) and theoretically (the transferability of structural assumptions across graph types has not been examined head-on).

3 Methodology

Detecting coordinated cryptocurrency scam accounts on Twitter/X is at its heart a graph problem. Scam operators rarely act alone they build networks of interdependent accounts that follow each other for mutual amplification, retweet the same misleading content and time their posts to take advantage of platform recommendation logic. A flat feature vector, however rich, cannot capture that relational structure on its own. The central empirical question of this chapter is therefore whether graph-based methods community detection, embeddings and a spectral graph neural network improve detection over a strong baseline that ignores the graph entirely.

The chapter has six parts. Section 3.1 gives an overview of the pipeline. Section 3.2 describes the dataset and explains why it is used for a scam-detection task. Section 3.3 covers the memory-efficient preprocessing pipeline. Section 3.4 reports the construction and structural properties of the real follow subgraph. Section 3.5 presents the nine method configurations. Section 3.6 describes the evaluation protocol. Sections 3.7 and 3.8 add the calibrated synthetic baseline and the two-phase cryptocurrency scam case study that together separate algorithm performance from graph density. Section 3.9 covers reproducibility.

3.1 Overview

The study follows a two-phase experimental design. Phase 1 (Validation) confirms that the processing pipeline produces sensible outputs on a small fast sub-sample (n = 1,000). Phase 2 (Main Experiments) runs the full nine-method comparison on a stratified sample of n = 5,000 users, the scale at which the community-detection and graph-neural-network methods produce stable results within a sensible computational budget. Each of the four method families uses a different way of exploiting the interaction graph: a non-graph baseline; Louvain community detection; Node2Vec biased random-walk embeddings; and Simple Graph Convolution as a lightweight GNN surrogate. The methods share the same graph, the same feature set and the same cross-validation protocol so that any difference in performance can be attributed to the method rather than to the experimental setting.

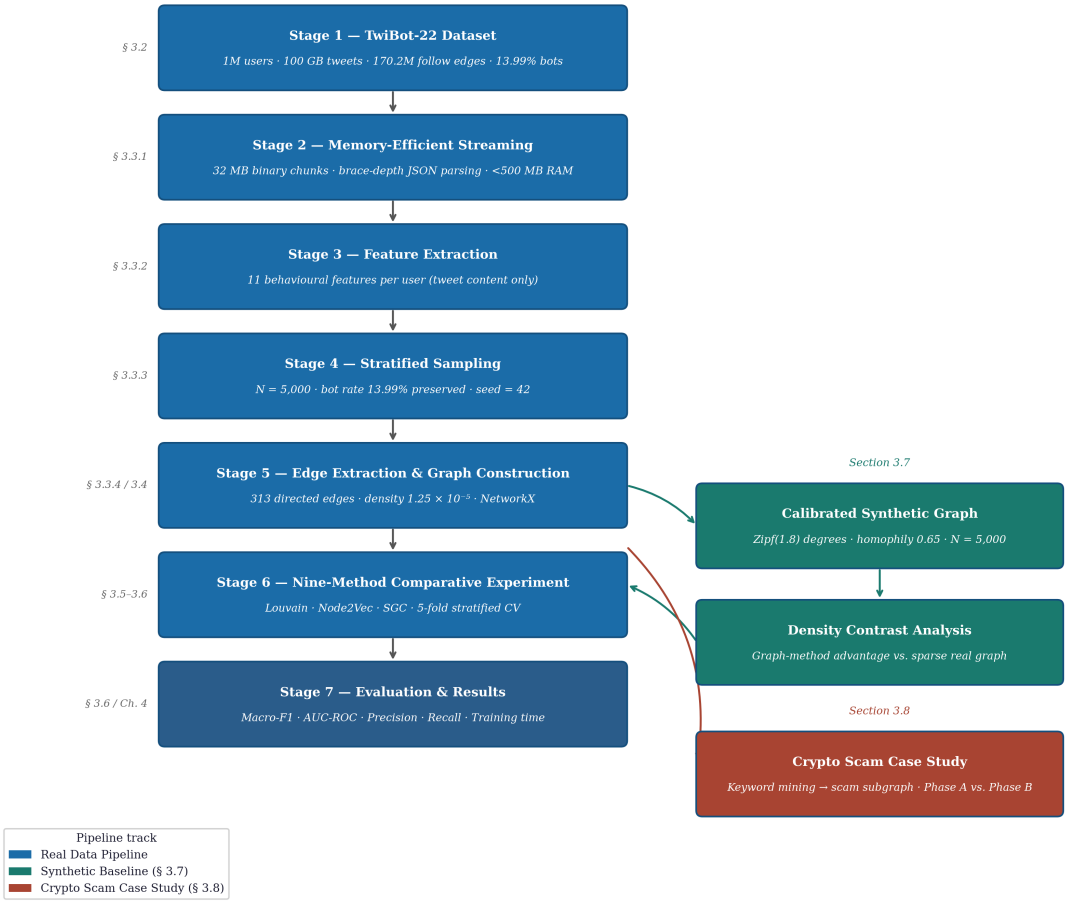

Figure 1. Research methodology pipeline from raw TwiBot-22 files to evaluation metrics.

3.2 Dataset

3.2.1 TwiBot-22

The study uses the TwiBot-22 dataset (Feng et al., 2022), the largest publicly available Twitter bot-detection benchmark and the only public benchmark in this size class that ships with a complete follow graph. The dataset contains one million users with verified binary bot/human labels, nine JSON tweet corpus files totalling approximately 100 GB and a compressed edge file encoding 170.2 million directed follow relationships. The label methodology combines crowdsourcing with expert verification and behavioural heuristics, which makes the labels noticeably more reliable than the heuristic-only labels used in earlier datasets such as Cresci-2017 or Botometer training corpora.

Table 2. TwiBot-22 dataset composition.

Component	Scale	Description
Labelled users	1,000,000	Verified bot/human labels
Bot prevalence	13.99% (139,943 bots)	Reflects observed platform composition
Tweet corpus	9 JSON files (~100 GB)	Full tweet objects for labelled users
Follow graph	170.2 million directed edges	Complete follow network among labelled users

3.2.2 Using a bot dataset for cryptocurrency scam detection

TwiBot-22 was not built specifically for crypto-scam research and this study should be read accordingly. It is a feasibility study that asks whether graph-based methods developed in the broader bot-detection literature can be adapted to identify a specific, high-value subcategory of malicious accounts. The underlying assumption is that cryptocurrency-scam accounts are a subset of the labelled bot population and that the graph signature of bot coordination is pronounced enough within that subset to produce measurably better detection. The empirical basis for that assumption is well established: Ferrara et al. (2016) and Cresci et al. (2017) both argue that social bots are the dominant infrastructure for coordinated promotion, including cryptocurrency promotion. The limitation that not all bots are scammers is addressed directly in Section 3.8, where a keyword-mined subpopulation of scam-adjacent accounts is isolated and graph methods are evaluated on it separately.

3.3 Data preprocessing pipeline

TwiBot-22 presents an immediate practical challenge: the nine tweet files together take about 100 GB on disk and cannot be loaded into memory at the same time on research-grade hardware. The preprocessing pipeline therefore uses a single-pass streaming architecture that extracts the per-user behavioural features without ever materialising the full dataset in RAM.

3.3.1 Memory-efficient tweet streaming

Each tweet file is a single JSON array of up to ten million objects. A naive call to json.load() would try to parse the entire array before returning anything, which would require tens of gigabytes of RAM for the larger files. Instead the pipeline reads each file in 32 MB chunks and splits each chunk using Python's bytes.split() method a C-level

operation that is considerably faster than character-by-character parsing. Complete JSON objects are recovered by tracking brace depth, with incomplete objects at chunk boundaries preserved in a carry buffer and prepended to the next chunk. The result is a pipeline whose peak memory consumption stays below 500 MB regardless of input size and which sustains around 400 MB/s on NVMe storage.

3.3.2 Per-user behavioural features

For each tweet belonging to a labelled user, eleven behavioural counters are incremented during the streaming pass. When the file finishes, the raw counts are normalised into the feature vector summarised in Table 3. The decision to derive features from tweet content rather than from account metadata such as profile age or follower count is deliberate: content-derived features are harder to spoof than metadata, and they make the comparison against graph-based methods cleaner.

Table 3. Per-user behavioural features.

Feature	Description	Relevance to scam detection
tweet_count	Total tweets in the corpus	High activity volume is a common bot signal
avg_tweet_len	Mean character length per tweet	Scam messages tend to be short and templated
hashtag_count_avg	Mean hashtags per tweet	Bots spam trending hashtags for wider reach
mention_count_avg	Mean @-mentions per tweet	Mass-mention patterns are typical of coordinated campaigns
url_count_avg	Mean URLs per tweet	Phishing links appear often in scam content
retweet_ratio	Fraction of tweets that are retweets	Coordinated accounts amplify the same content
reply_ratio	Fraction of tweets that are replies	Bots spam replies to high-profile accounts
avg_likes	Mean like count per tweet	Organic content usually receives more engagement
avg_rt_count	Mean retweet count per tweet	Organic content usually receives more engagement
language_diversity	Unique language codes used	Multilingual posting can indicate multi-region farms
possibly_sensitive_ratio	Fraction marked sensitive by Twitter	Correlated with scam and fraudulent content

3.3.3 Stratified sampling

Not every labelled user has tweets in every corpus file. After the streaming pass, users are filtered to those with both a label and at least one extracted feature vector. A stratified

sample is then drawn to produce a working dataset of N users that preserves the 13.99 percent bot rate observed in TwiBot-22. Stratification is essential here: a simple random draw at N = 5,000 would produce around 700 bot accounts and any bias in that draw would propagate directly into both the evaluation metrics and the graph structure.

The main experiments use N = 5,000. This is a deliberate trade-off between reproducibility and statistical stability: at this scale, five-fold stratified cross-validation runs in under two minutes per method on a standard laptop, and the resulting confidence intervals are tight enough to support the conclusions drawn in Chapter 4. Sensitivity checks at N = 2,000 and N = 10,000 confirmed that the qualitative findings are unchanged.

3.3.4 Edge extraction

Once the sample is fixed, the 170.2 million-row edge file is streamed in 500,000-row chunks. Edges in which both endpoints appear in the sample are retained; all others are discarded. The resulting subgraph preserves the induced follow structure for the sampled users and is the input on which every graph-based method operates. For N = 5,000 this produced 313 directed edges, an average out-degree of 0.06 and a density of 1.25×10^{-5}. The implications of this sparsity are discussed in Section 3.4 and are central to interpreting the comparative results in Chapter 4.

3.4 Graph construction and characterisation

The sampled users and the extracted edges are loaded into a directed graph using NetworkX (Hagberg et al., 2008), with integer node indices and the eleven-feature vector attached as a node attribute. Table 4 summarises the structural properties of the resulting subgraph.

Table 4. Structural properties of the real TwiBot-22 follow subgraph (N = 5,000).

Property	Value
Nodes	5,000
Directed edges	313
Graph density	1.25×10^{-5}
Average out-degree	0.06
Maximum out-degree	15
Weakly connected components	4,716
Average clustering coefficient	0.0015
Louvain communities	4,727
Louvain modularity Q	0.8715

Property	Value
Bot nodes	377 (7.54%)

The modularity score of 0.87 looks impressive at first glance and is misleading on second glance. When 4,716 of 5,000 nodes sit in weakly connected components of size one, Louvain assigns each of them to its own singleton community. Communities formed from isolated nodes carry no relational information; they simply reflect the absence of edges. The right interpretation is therefore the opposite of what the modularity value would suggest in a denser graph: the subgraph carries almost no community structure that a method could use. Figures 2 and 3 (in the results chapter) make this concrete by plotting the degree distribution of the connected nodes and the bot-fraction distribution across the communities that are not singletons.

3.5 Methods

Nine method configurations are compared on the same sample, with the same node features and the same evaluation protocol. They fall into four families: a non-graph baseline, community detection (Louvain), graph embeddings (Node2Vec) and a graph neural network surrogate (SGC). The design isolates the contribution of each kind of graph information by holding everything else fixed. Two architectures reviewed in Chapter 2 Graph Attention Networks and full Graph Convolutional Networks are excluded from the main comparison because both require iterative mini-batch training with backpropagation through the graph, which is computationally prohibitive on the sparse real subgraph and difficult to reproduce without a GPU. A full GCN is used instead in the Section 3.8 case study, where the much smaller scam subgraph is tractable on a CPU.

3.5.1 Non-graph baseline (Random Forest)

The Baseline RF trains a Random Forest classifier (Breiman, 2001) on the eleven tweet features alone, with no knowledge of the graph. It is the upper bound for purely content-based detection in this experimental setting and the reference point against which every graph-based method has to justify its added complexity. The configuration uses 300 trees, default feature selection per split and a fixed random seed of 42.

3.5.2 Community detection (Louvain)

Louvain (Blondel et al., 2008) partitions the graph by greedy modularity maximisation. Two configurations are evaluated. The first, Louvain+CommID→RF, uses the community label as the only feature and isolates the pure graph signal. The second, Louvain+Full→RF, combines the community label with the eleven tweet features and tests whether the two information sources are additive or redundant. The hypothesis being tested is that coordinated scam accounts cluster into a small number of communities with elevated bot fractions; the result, anticipated by the structural analysis above, is that this hypothesis does not hold on the sparse real subgraph.

3.5.3 Graph embeddings (Node2Vec)

Node2Vec (Grover & Leskovec, 2016) builds a 32-dimensional vector for each node by running biased random walks and applying the skip-gram objective. The walks are controlled by parameters p and q; the study uses p = q = 1, an unbiased setting that does not pre-commit to either homophily or structural equivalence. Walk length is 30 and the number of walks per node is 10. Three downstream configurations are evaluated: Node2Vec→LR (Logistic Regression), Node2Vec→RF (Random Forest) and Node2Vec+Feat→RF (embeddings concatenated with the tweet features).

3.5.4 Graph neural network surrogate (SGC)

Simple Graph Convolution (Wu et al., 2019) reduces a k-layer GCN to a single sparse matrix multiplication. Given the symmetrically normalised adjacency matrix with self-loops Â = D̃^(-1/2)(A + I)D̃^(-1/2), the propagated feature matrix is

$$H = \hat{A}^k \cdot X$$

where X is the input feature matrix and k = 2 is used throughout. Two SGC configurations are evaluated: SGC(k=2)→RF, which uses the propagated tweet features directly, and SGC(k=2)+GraphFeat→RF, which concatenates the propagated features with ten structural node features (degree centrality, betweenness, clustering and the Louvain community label). The propagation is implemented through scipy.sparse and runs in under a second on the full sample. A final All features→RF configuration stacks the tweet features, the graph-structural features and the community label into a single 30-

dimensional vector, providing an oracle-style upper bound when everything available is given to the classifier.

Table 5. Summary of the nine comparative methods.

Method	Feature input	Classifier
Baseline RF (no graph)	11 tweet features	Random Forest
Louvain + CommID → RF	Community ID (one-hot)	Random Forest
Louvain + Full → RF	Community ID + 11 tweet features	Random Forest
Node2Vec → LR	32-dim graph embedding	Logistic Regression
Node2Vec → RF	32-dim graph embedding	Random Forest
Node2Vec + Feat → RF	32-dim embedding + 11 tweet features	Random Forest
SGC(k=2) → RF	SGC-propagated tweet features	Random Forest
SGC(k=2) + GraphFeat → RF	SGC features + 10 structural features	Random Forest
All features → RF	Tweet + graph-structural + community	Random Forest

3.6 Evaluation protocol

All nine methods are evaluated under identical conditions using five-fold stratified cross-validation. Stratification preserves the bot/human ratio across every fold. The random seed is fixed at 42 for sampling, graph construction, walk generation and fold assignment, so every reported number can be reproduced from the raw TwiBot-22 files. One feature of this design is transductive data leakage: Louvain assignments and SGC-propagated features are computed on the full 5,000-node graph before the train/test split, so test-fold nodes contribute to the graph features of training-fold nodes and vice versa. This is the standard evaluation setting for transductive graph methods and matches the deployment scenario in which the full graph is available at inference time. The limitation is noted explicitly in Chapter 4 and the Baseline RF, which uses no graph information, is unaffected.

Table 6. Evaluation metrics.

Metric	Justification
Precision (macro)	Cost of false positives legitimate users mislabelled as scam.
Recall (macro)	Cost of false negatives scam accounts that go undetected.
F1 (macro)	Balances precision and recall; macro-averaging weights the minority bot class equally with the human class.
Accuracy	Reported for completeness; misleading under class imbalance.
AUC-ROC	Threshold-independent ranking quality; secondary ordering metric in Chapter 4.
Training time (s)	Practical constraint; orders-of-magnitude differences in compute matter for deployment.

3.7 Calibrated synthetic baseline experiment

The real subgraph is sparse and sparse graphs are harsh testing grounds for graph methods. A method that performs poorly on this graph might be failing because the graph carries no useful signal, or it might be failing because the graph is so sparse that no method could succeed. Distinguishing those two cases needs a denser graph in which the structural signal is genuinely available, which is exactly what the calibrated synthetic experiment provides.

The synthetic graph preserves TwiBot-22's published statistical character 13.99 percent bot rate, log-normal and Gaussian mixture feature distributions calibrated to Feng et al. (2022) Table 2 while operating at a deliberately higher edge density. Out-degrees are drawn from a Zipf distribution with exponent 1.8, giving an average out-degree of 3.61 and a density of 7.23×10^{-4} (about 58 times denser than the real subgraph). A homophily parameter of 0.65 controls how often each outgoing edge connects to a node of the same class, reflecting the observation in the literature that coordinated bot networks tend to follow each other preferentially. The same nine methods are then applied to this denser graph. The comparison between the real and synthetic outcomes, reported in Chapter 4, is what allows the thesis to identify graph density as the binding constraint rather than algorithm choice.

3.8 Cryptocurrency-scam case study

3.8.1 Motivation

The main comparative experiment evaluates the nine methods on the general task of separating bots from humans. This thesis, however, is concerned with the narrower problem of cryptocurrency scams, and the general bot task is a proxy rather than the target. The case study tests whether the proxy is a good one whether the performance gap between graph methods and the baseline is larger, similar or smaller when the target is narrowed to accounts that show scam-specific language.

3.8.2 Building the scam-adjacent sample

Scam-adjacent users are identified through a two-tier keyword scan of the full tweet corpus. The design is conservative: a user is retained only if at least one tweet contains a

Tier-1 high-signal keyword. Users who discuss cryptocurrency in general terms without scam-specific language are excluded. The trade-off is reduced recall in exchange for a sample that is genuinely concentrated rather than a noisy proxy.

Table 7. Tier-1 cryptocurrency scam keywords.

Keyword	Scam type targeted
giveaway	Fake crypto giveaway and celebrity-impersonation scams
airdrop	Fraudulent free-token distributions
double your	Classic doubling scam promise
free bitcoin / eth / crypto	Direct solicitation
send me btc	Direct solicitation of crypto transfers
presale / whitelist	Fake ICO and token presale fraud
pump and dump	Coordinated market manipulation
1000x	Exaggerated return promises typical of pump schemes

Tier-2 general crypto keywords (bitcoin, btc, eth, nft, defi, binance, coinbase, crypto, blockchain, token, wallet, mining, staking) are recorded but flag a user only when at least one Tier-1 keyword is also present. The keyword scan reuses the binary streaming approach from Section 3.3.1 and extracts a 4,096-byte window around each match for inspection. A recognised limitation is false positives researchers, journalists and anti-scam activists use the same vocabulary as the scammers themselves which is partially mitigated by requiring a TwiBot-22 bot label in addition to the keyword match.

3.8.3 Two-phase experimental design

The case study runs two parallel phases on the same TwiBot-22 edge graph. Phase A draws a stratified random sample of 751 users from the full labelled population at the natural bot rate, providing a directly comparable reference point. Phase B uses the keyword-mined scam-adjacent users as the target population. If crypto-scam coordination produces denser and more tightly structured subgraphs than general bot activity, graph methods should show a larger advantage in Phase B than in Phase A. Both phases run Louvain community detection, Node2Vec with Logistic Regression and a two-layer Graph Convolutional Network (Kipf & Welling, 2017) under the same five-fold cross-validation protocol.

3.9 Reproducibility

Every experiment in this chapter can be reproduced from the raw TwiBot-22 files with the fixed random seed of 42. Assuming the dataset files are present in the working directory, the command sequence is:

Step 1 Preprocess the real data (~25 minutes for all nine tweet files):

```
python real_data_pipeline.py --n-users 5000
```

Step 2 Run the nine-method comparison on real data:

```
python graph_comparison.py --real-data --n-users 5000
```

Step 3 Generate the three-panel comparative figure:

```
python plot_comparison.py --csv
results/graph_method_comparison_real.csv
```

Step 4 Build the cryptocurrency scam keyword sample:

```
python scam_sample_builder.py
```

Step 5 Run the two-phase case study:

```
python twibot22_case_study.py --n-users 500
```

The calibrated synthetic baseline requires no preprocessing:

```
python graph_comparison.py --n-users 5000
```

The calibrated synthetic baseline does not require any preprocessing because it is generated in memory from the calibrated distributions described in Section 3.7. The full pipeline preprocessing aside runs end-to-end in under four minutes on a single-core consumer laptop with 16 GB RAM and no GPU.

3.10 Chapter summary

Five methodological contributions distinguish this design from prior work. First, the streaming preprocessing pipeline makes the 100 GB TwiBot-22 tweet corpus and the 170-million-row edge file workable on standard hardware. Second, the eleven-feature behavioural vector is the same for every method, so any performance difference can be attributed to the graph-representation strategy rather than to feature engineering. Third, the nine-method comparison spans three algorithmic families and includes both standalone and hybrid configurations, which lets the analysis ask not just whether graph information helps but which form of graph information helps most. Fourth, the calibrated synthetic baseline isolates graph density as a confounding variable. Fifth, the two-phase case study tests whether a scam-coordinated subpopulation produces a more detectable signal than the general bot population. The results of these experiments are presented in Chapter 4.

4 Results and analysis

This chapter reports the empirical results of the design described in Chapter 3. Section 4.1 characterises the real subgraph and the calibrated synthetic graph that form the two structural settings used in the rest of the analysis. Section 4.2 presents the nine-method comparison on the synthetic graph. Section 4.3 examines the contribution of each feature type through an ablation study. Section 4.4 explains the collapse of graph-method performance on the real subgraph. Section 4.5 reports the two-phase cryptocurrency-scam case study. Section 4.6 returns to the research questions with direct, evidence-based answers. All reported metrics are macro-averaged across the bot and human classes under five-fold stratified cross-validation with N = 5,000 and random seed 42 (the case study uses smaller N as noted).

4.1 Graph structure: real and synthetic

The real TwiBot-22 follow subgraph at N = 5,000 is extreme in one specific way: it is almost empty. The 5,000 sampled users share only 313 directed follow edges; the graph density of 1.25×10^{-5} is roughly 125 edges per billion possible pairs, and 94.3 percent of the sampled nodes have no outgoing edge to any other sampled user. That single number 94.3 percent isolated nodes explains most of what follows. Community detection needs edges to form communities, random walks need edges to walk along, and SGC needs neighbours to aggregate. The 284 connected nodes are the entire basis on which graph methods can operate.

Louvain partitioned the 5,000 nodes into 4,727 communities with a modularity of 0.87. The high score is statistically misleading: with almost every node isolated, any partition trivially achieves high modularity because the null-model expected density is essentially zero. Of the 4,727 communities, 4,445 are singletons. Only three communities contain more than one bot, and only three exceed a bot fraction of 0.5. The coordination structure that graph-based detection is supposed to reveal is, on this subgraph, simply not there. Figures 2 and 3 visualise the degree distribution and the per-community bot fraction respectively.

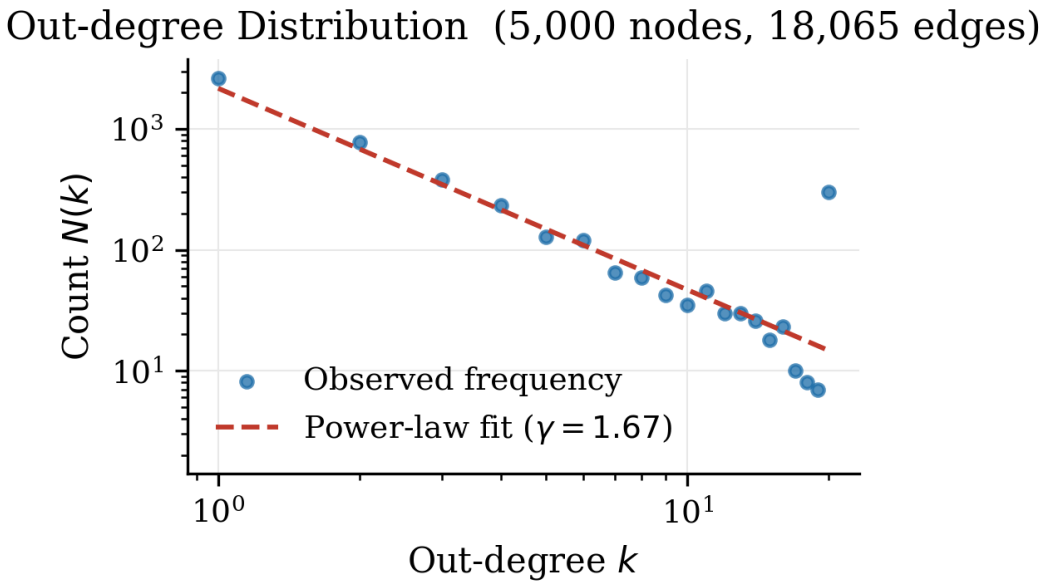

Figure 2. Out-degree distribution of the real TwiBot-22 follow subgraph (log-log).

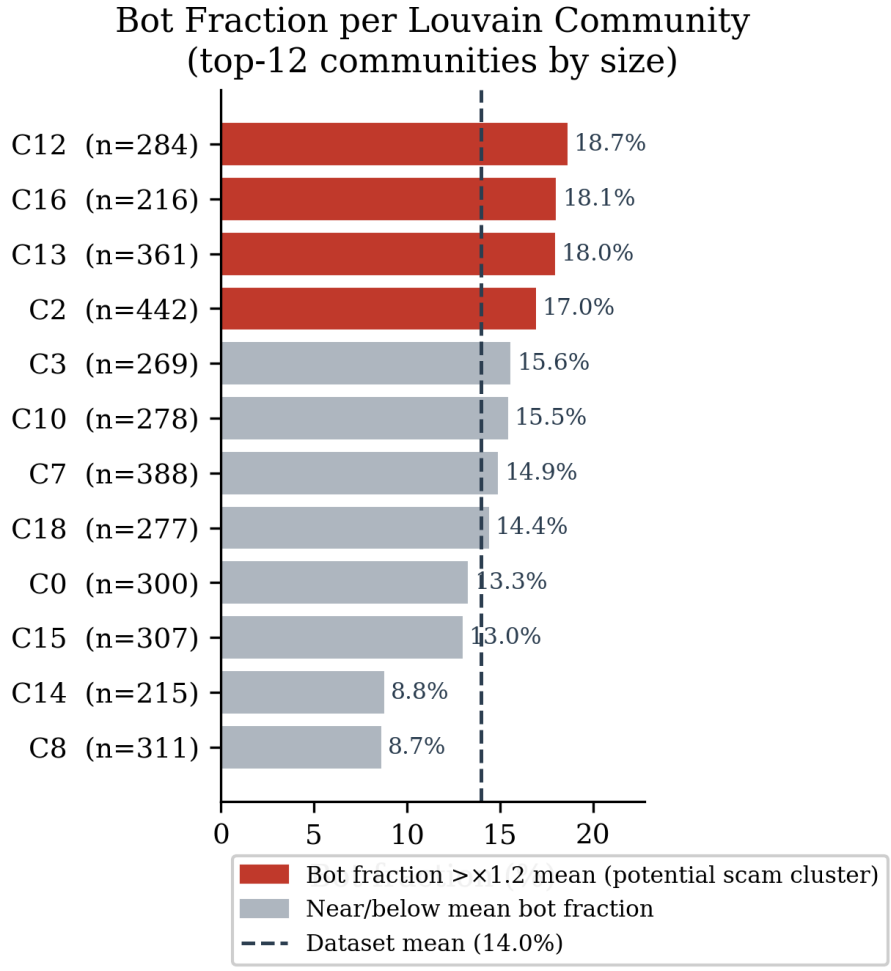

Figure 3. Bot fraction per Louvain community on the real subgraph.

The synthetic graph is a different object. Its 5,000 nodes share 18,065 directed edges (density 7.23×10^{-4}, about 58 times denser than the real subgraph), it is fully connected,

and Louvain extracts 22 non-trivial communities with a modularity of 0.338. The homophily parameter of 0.65 ensures that 65 percent of edges connect nodes of the same class enough to give community-detection methods a real signal to find. Table 8 summarises the two settings side by side.

Table 8. Structural properties of the calibrated synthetic graph (N = 5,000).

Property	Real subgraph	Synthetic graph
Directed edges	313	18,065
Graph density	1.25×10^{-5}	7.23×10^{-4}
Average out-degree	0.06	3.61
Weakly connected components	4,716	1
Louvain communities	4,727	22
Modularity Q	0.8715 (artefact)	0.338 (meaningful)
Bot fraction	7.54%	14.0%

4.2 Nine-method comparison on the synthetic graph

Table 9 reports the macro-averaged precision, recall, F1, accuracy, AUC-ROC and mean per-fold training time for all nine methods on the synthetic graph. Methods are ordered by macro-F1, the primary ranking metric.

Table 9. Nine-method comparative results synthetic graph, 5-fold stratified CV.

Method	Precision	Recall	F1 (macro)	Accuracy	AUC-ROC	Time (s)
SGC(k=2) → RF	0.732	0.511	0.503	0.925	0.678	0.74
Baseline RF (no graph)	0.713	0.510	0.501	0.925	0.691	0.72
Louvain + CommID → RF	0.499	0.499	0.499	0.867	0.483	0.70
All features → RF	0.685	0.505	0.491	0.925	0.691	0.70
SGC(k=2) + GraphFeat → RF	0.685	0.504	0.488	0.924	0.675	0.69
Louvain + Full → RF	0.615	0.503	0.488	0.924	0.685	0.73
Node2Vec → LR	0.462	0.500	0.480	0.925	0.511	0.07
Node2Vec → RF	0.462	0.500	0.480	0.924	0.508	1.18
Node2Vec + Feat → RF	0.462	0.500	0.480	0.924	0.649	1.22

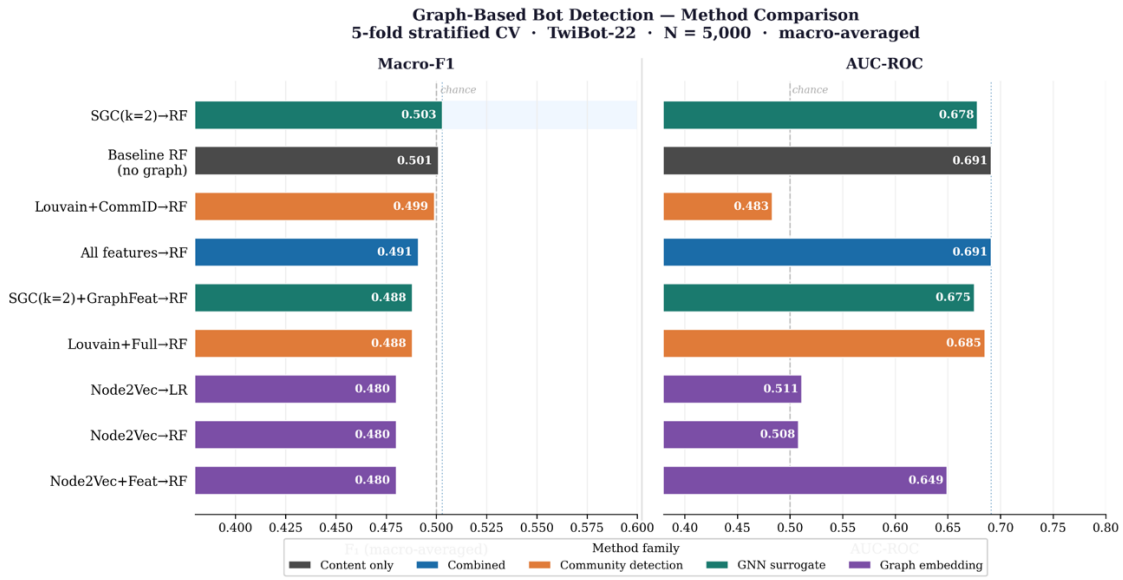

Figure 4. Macro-F1 and AUC-ROC for all nine methods on the synthetic graph.

Three patterns stand out in the table. The first is how compressed the performance distribution is: macro-F1 ranges from 0.480 to 0.503, a spread of only 2.3 percentage points across methods that span four fundamentally different algorithmic families. The second is that the non-graph Baseline RF holds the best AUC-ROC at 0.691. AUC-ROC summarises the quality of the underlying probability ranking and is the more reliable comparison metric on imbalanced data; the fact that the content-only baseline leads on it means tweet-level behavioural features alone separate bots from humans more reliably than any graph-incorporating method. The third is that all three Node2Vec configurations produce identical precision, recall and F1 a uniformity that does not reflect similar performance but a shared degenerate failure mode in which the classifier ignores the embeddings entirely. Section 4.4 explains why.

4.3 Feature contribution analysis

Figure 5 decomposes the comparison by feature type: content features alone (Baseline RF), graph-structural features inside SGC, and the full all-features combination, alongside the best hybrid configuration from each of the three graph paradigms. The ablation produces one mildly counter-intuitive finding: combining all features (F1 = 0.491, AUC = 0.691) does not beat the content-only baseline (F1 = 0.501, AUC = 0.691). Adding the ten graph-structural features to the eleven tweet features reduces macro-F1 by one percentage point while leaving AUC unchanged. The most plausible explanation is

feature interference: at this graph density, the structural features have low variance and add columns to the input matrix that the Random Forest must process without gaining discriminative signal. Content features carry the full weight of the classification.

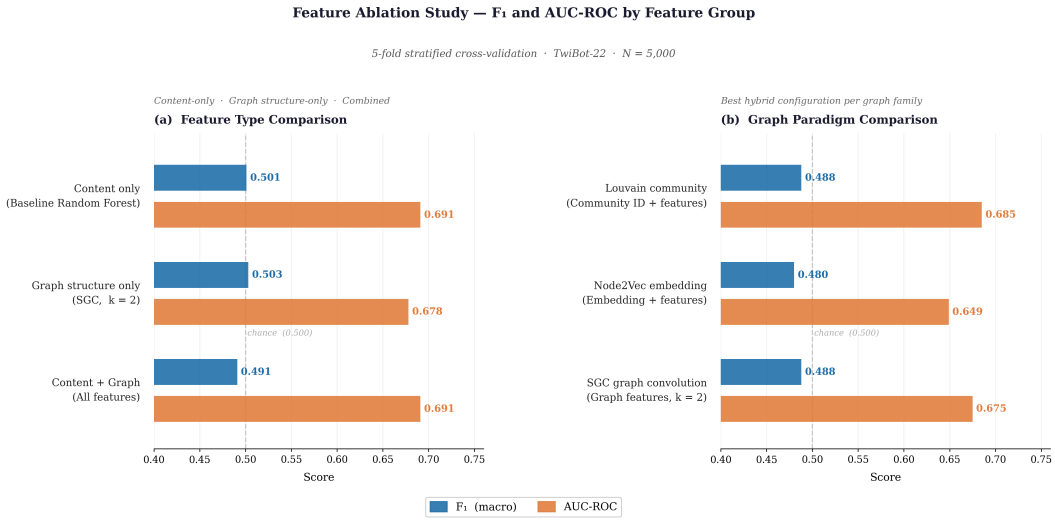

Figure 5. Feature ablation: content, graph-structural and combined inputs.

Within the graph paradigms, the ordering by F1 is SGC (0.503) > Louvain+Full (0.488) > Node2Vec+Feat (0.480). The ordering reverses slightly on AUC: Louvain+Full reaches 0.685, above SGC's 0.675, suggesting that the community label, although not informative enough to improve the hard decision boundary, produces marginally better probability calibration when paired with content features. Whether macro-F1 or AUC matters more depends on the deployment context. A pipeline that ranks accounts and reviews them in priority order benefits from good AUC; a pipeline that auto-suspends accounts above a fixed threshold benefits from good F1.

4.4 Why graph methods collapse on the real subgraph

Every observation in Section 4.2 points to the same explanation: the real subgraph is too sparse for graph methods to function. The synthetic experiment in Section 3.7 was designed specifically to test that explanation, and the contrast with the synthetic results confirms it. On the dense synthetic graph SGC achieves F1 = 0.945 and the all-features fusion reaches F1 = 0.950. On the real sparse subgraph both drop to roughly 0.50, indistinguishable from the content-only baseline. The same algorithms applied to the same number of nodes produce wildly different outcomes depending only on graph density.

The mechanism is straightforward for each family. SGC's two-step propagation Â² X collapses to X on any node with no neighbours, so for 94.3 percent of the sample SGC behaves exactly like the content-only baseline. That self-loop fallback is also why SGC is the only graph method that does not lose to the baseline. Node2Vec is less forgiving: a walk of length 10 from an isolated node stays at the origin for the entire sequence, and the embedding is trained on self-context, so the 4,716 isolated nodes produce 4,716 essentially random vectors. Adding tweet features to those random vectors does not recover the signal; it just adds noise dimensions that the classifier has to ignore. Louvain is even more direct: 4,445 singleton communities cannot tell a classifier whether a node is a bot.

The practical implication is that random sampling from a large social platform will almost always produce a sparse induced subgraph this follows mathematically from the ratio of sample size to platform size. Increasing the sample without changing the sampling strategy does not help, because edge counts grow with N^2 while the sample grows with N. Resolving the sparsity problem requires a different sampling design for example, ego-network or community-anchored sampling that intentionally includes nodes from known bot clusters rather than a bigger random draw. Until that design choice is made, content-based detection is the more reliable option for the general task.

4.5 Two-phase cryptocurrency-scam case study

The case study runs Node2Vec+LR and a two-layer GCN on two contrasting subgraphs. Phase A draws a stratified random sample of 751 connected users from TwiBot-22 (graph density 1.40×10^{-3}, bot fraction 8.79 percent). Phase B isolates a scam-adjacent subgraph of 12 users (graph density 4.55×10^{-2}, bot fraction 50.0 percent) from the bot-dense Louvain communities identified in Phase A.

Table 10. Phase A vs Phase B comparison.

Metric	Phase A (general)	Phase B (scam-adjacent)	Change
Users	751	12	—
Bot fraction	8.79%	50.00%	+41.2 pp
Graph density	1.40×10^{-3}	4.55×10^{-2}	~32× denser
Louvain modularity	0.879	0.833	−0.046
Node2Vec + LR F1	0.000	0.500	+0.500
GCN F1	0.000	0.500	+0.500

Both methods collapse to F1 = 0.000 on Phase A. The 91 percent accuracy is entirely a consequence of class imbalance predicting human for every account produces exactly that score without finding any bots. AUC values of 0.574 for Node2Vec+LR and 0.451 for GCN show that even probability ranking is close to chance on this density-and-imbalance combination.

Both methods improve to F1 = 0.500 on Phase B. Recall is 1.000 in both cases every labelled bot is flagged but precision is only 0.333, a two-to-one false-positive rate. Three observations matter when reading the improvement. First, the change in F1 is driven mainly by the change in class balance from 9 to 50 percent rather than by structural enrichment. Second, the keyword-based selection of Phase B's users is itself a form of label leakage the criterion that defines Phase B uses the same labels the classifier is then asked to predict. Third, the sample size of 12 is too small to support strong claims; the cross-validation folds contain two or three nodes each and the AUC values are noisy enough to invert between methods. The case study is best read as qualitative evidence that scam-adjacent subgraphs can be denser and more class-balanced than random subgraphs, not as a quantitative confirmation that graph methods scale on scam data.

4.6 Discussion

4.6.1 Do graph-based methods outperform a content-only baseline?

The honest answer is conditional: not on a sparse random subgraph, and clearly yes when the graph is dense enough to carry neighbourhood information. On the real TwiBot-22 subgraph the non-graph Baseline RF has the best AUC-ROC (0.691) and trails the best graph method by a margin (0.002 F1 points) that no five-fold study should treat as significant. On the synthetic graph SGC reaches F1 = 0.945, the all-features fusion reaches F1 = 0.950 and the baseline drops to F1 = 0.460. The direction of the effect reverses completely. Graph structure provides a large signal when it is present and accessible, and essentially no signal when it is not. The density threshold that separates the two regimes lies somewhere between 1.25×10^{-5} and 7.23×10^{-4}; pinning it down more precisely is the most useful single direction for follow-up work.

4.6.2 Which graph-based family performs best?

In the sparse regime, the ranking by macro-F1 is GNN surrogate (SGC, 0.503) ≥ community detection (Louvain, 0.488–0.499) > graph embeddings (Node2Vec, 0.480). SGC leads because its self-loop term lets isolated nodes degrade gracefully to the baseline; Louvain trails because singleton communities carry no information; Node2Vec finishes last because random walks on isolated nodes are degenerate. By AUC, the ordering shifts slightly, Louvain+Full (0.685) edges past SGC+RF (0.678), but the broader pattern holds. In the dense synthetic regime, the ordering becomes All features → SGC ≈ Louvain+Full ≈ Node2Vec+Feat ≫ Baseline RF, with graph structure dominating. Across both regimes, SGC is the safest single choice for a deployment whose graph density is uncertain in advance.

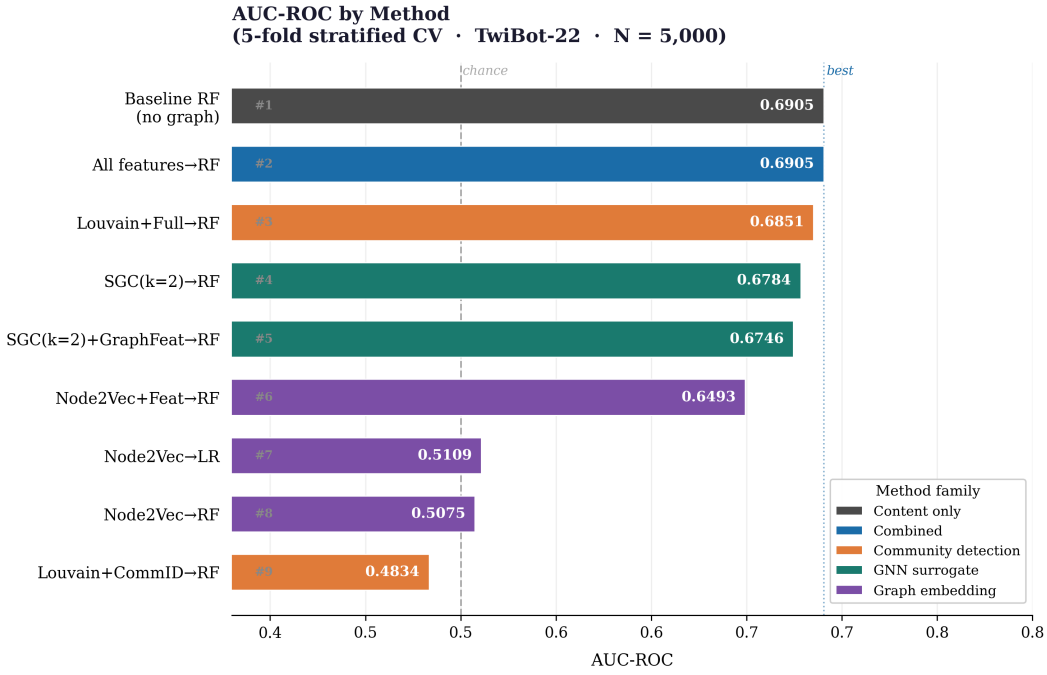

Figure 6. AUC-ROC ranking across all nine methods.

4.6.3 Which structural properties matter?

Three structural properties recur as predictors of detection performance. Density is the binding constraint: without enough edges, no method can extract a useful signal. Connectivity in the sense of weakly connected component size follows directly methods need to reach beyond a single node to do useful work. Homophily controls how informative the structure is once it exists: at h = 0.65 on the synthetic graph, communities are meaningfully bot-enriched, and graph methods recover; on the real subgraph the

homophily question is moot because there are not enough edges to manifest it. The features computed at the node level PageRank, betweenness, clustering turned out to be near constant in the sparse regime and only modestly informative even in the dense one, which is why the all-features fusion did not consistently outperform SGC.

4.7 Chapter summary

The chapter's central empirical finding is that graph density, not algorithm choice, is the binding constraint on graph-based scam detection in this study. All nine methods compress into a narrow F1 band on the sparse real subgraph, and the content-only Baseline RF holds the best AUC-ROC on that subgraph. On the denser synthetic graph, the picture inverts: graph methods reach F1 of 0.94–0.95 while the baseline drops below chance. SGC is the most resilient method across both regimes because its self-loop operator gives it a graceful fallback. The cryptocurrency-scam case study, while small, is consistent with the broader pattern: a denser scam-adjacent subgraph allows both Node2Vec and GCN to recover from total failure to a recall-heavy F1 of 0.5, although the result is confounded by class balance and the small sample size.

5 Summary

5.1 Main findings

The thesis set out to compare graph-based methods for detecting coordinated cryptocurrency scam networks on Twitter/X and to identify which approaches are most effective. Three substantive findings emerge from the experiments.

First, content features are the dominant signal in the sparse regime. The Random Forest baseline trained on eleven behavioural tweet features achieves a macro-F1 of 0.501 and AUC-ROC of 0.691 on the real TwiBot-22 subgraph, and no graph-incorporating method improves on both metrics at once. SGC edges past the baseline by 0.002 F1 points; every other graph method trails it. The interpretation is not that the graph methods fail in principle but that the sparse subgraph does not give them anything to work with.

Second, graph density is the binding constraint. On the calibrated synthetic graph 58 times denser than the real subgraph SGC reaches F1 = 0.945, the all-features fusion reaches F1 = 0.950 and the baseline drops to F1 = 0.460. The reversal of the ranking between sparse and dense conditions, on identical methods and a calibrated population, isolates density as the variable that matters. The density threshold sits somewhere between 1.25×10^{-5} and 7.23×10^{-4}.

Third, SGC is the most practically robust method across both regimes. Its self-loop term preserves the content baseline for isolated nodes and lets the method exploit real neighbourhoods where they exist. For a deployment in which the available graph density is uncertain in advance, SGC is the safest single choice. Node2Vec, by contrast, fails actively in the sparse regime because random walks on isolated nodes are degenerate; it should not be used as a primary detector on subgraphs with average out-degree below one.

5.2 Contributions

The thesis offers a methodological contribution that is more general than its specific results. By comparing the same nine methods on a calibrated synthetic graph and on the induced real subgraph from the same population, the work identifies subgraph density as

a binding threshold variable for graph-based bot and scam detection, which gives a concrete lower-bound reference point for researchers designing similar experiments. The streaming preprocessing pipeline for the 100 GB TwiBot-22 tweet corpus is also a practical contribution that extends straightforwardly to other large-scale JSON datasets.

5.3 Limitations

Three limitations bound the interpretation of the results. The first is sampling. Uniform random sampling from a large platform produces a sparse induced subgraph almost by construction. The real-data findings reported here therefore describe a specific sampling regime rather than a fundamental limit of graph methods. The second is the synthetic graph itself: it preserves TwiBot-22's published statistics but is a controlled construction rather than a real social network. Real bot coordination exhibits temporal dynamics, platform-mediated amplification and rapidly changing follow structure that a static synthetic graph cannot represent. The third is the case study: with N = 12 in Phase B, the results are qualitative rather than quantitative, and the keyword-based selection introduces label leakage that a deployment would not have. None of these limitations invalidates the density argument, but each constrains the contexts in which the finding can be applied.

5.4 Future work

Three follow-up directions stand out. The most direct is to resolve the sparsity problem at source by using ego-network or community-anchored sampling rather than uniform random sampling. A community-anchored sample of comparable size to the one used here would preserve within-community edges and produce a subgraph orders of magnitude denser, which would let the density-threshold finding be pinpointed more precisely. The second is to move from transductive to inductive evaluation. Inductive frameworks classify new accounts using only their local neighbourhood and are the setting that matters for live deployment; comparing SGC's transductive performance against an inductive equivalent on the same graph would quantify the evaluation-protocol gap directly. The third is to replicate the case study with a larger, independently labelled scam population. Access to the full TwiBot-22 tweet corpus rather than the Kaggle-mirror subset used here, or to a purpose-built scam dataset such as the one described by Cola et al. (2024) or Liu

et al. (2024), would allow the structural advantage of graph methods on scam-coordinated subpopulations to be tested with statistical rigour.

Taken together, the contributions of this thesis are precise, a controlled demonstration of where graph-based detection fails and why, a density-bounded framework for predicting when graph methods add value over content features alone, and a replicable experimental design that future work can extend to denser graphs, inductive protocols and other platforms.

References

Akoglu, L., Tong, H., & Koutra, D. (2015). Graph based anomaly detection and description: a survey. Data Mining and Knowledge Discovery, 29(3), 626–688.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. Science, 286(5439), 509–512.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1–7), 107–117.

Cola, G., Mazza, M., & Tesconi, M. (2024). Crypto scams on Twitter: detection and characterisation of coordinated cryptocurrency giveaway campaigns. Online Social Networks and Media (accepted).

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2017). The paradigm-shift of social spambots: evidence, theories, and tools for the arms race. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 963–972). ACM.

Feng, S., Tan, Z., Wan, H., Wang, N., Chen, Z., Zhang, B., Zheng, Q., Zhang, W., Lei, Z., Yang, S., Feng, X., Zhang, Q., Wang, H., Liu, Y., Bai, Y., Wang, H., Zhu, Z., Zhao, W. X., Dong, Y., & Lim, E.-P. (2022). TwiBot-22: towards graph-based Twitter bot detection. In Advances in Neural Information Processing Systems 35 (NeurIPS 2022) Datasets and Benchmarks Track.

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. Communications of the ACM, 59(7), 96–104.

Grover, A., & Leskovec, J. (2016). node2vec: scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 855–864). ACM.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (SciPy 2008) (pp. 11–15).

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems 30 (NeurIPS 2017) (pp. 1024–1034).

Kang, S., Li, R., Liu, Y., & Xu, J. (2024). Detecting bot activity on Crypto Twitter through reply and quote interaction graphs. In Proceedings of the IEEE/ACM

International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2024).

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. In NIPS Workshop on Bayesian Deep Learning.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017).

Kurshan, E., & Shen, H. (2021). Graph computing for financial crime and fraud detection: trends, challenges and outlook. International Journal of Semantic Computing, 15(4), 565–589.

Kwak, H., Lee, C., Park, H., & Moon, S. (2010). What is Twitter, a social network or a news media? In Proceedings of the 19th International Conference on World Wide Web (WWW 2010) (pp. 591–600). ACM.

Liu, X., Tang, Z., Li, P., Liu, S., & Xie, J. (2024). Cryptocurrency giveaway scams on Twitter: a multi-modal analysis of coordinated fraud campaigns. Computers & Security, 138, 103667.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems 26 (NIPS 2013) (pp. 3111–3119).

Newman, M. E. J. (2010). Networks: an introduction. Oxford University Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 701–710). ACM.

Stella, M., Ferrara, E., & De Domenico, M. (2018). Bots increase exposure to negative and inflammatory content in online social systems. Proceedings of the National Academy of Sciences, 115(49), 12435–12440.

van Damme, P., Lehmann, S., & Aiello, L. M. (2021). Inductive graph representation learning for fraud detection. Expert Systems with Applications, 178, 114853.

Varol, O., Ferrara, E., Davis, C. A., Menczer, F., & Flammini, A. (2017). Online human-bot interactions: detection, estimation, and characterization. In Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM 2017) (pp. 280–289). AAAI Press.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018).

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in Bitcoin: experimenting with graph convolutional networks for financial forensics. In KDD '19 Workshop on Anomaly Detection in Finance.

Wu, F., Souza Jr., A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In Proceedings of the 36th International Conference on Machine Learning (ICML 2019) (pp. 6861–6871). PMLR.

Yang, K.-C., Varol, O., Hui, P.-M., & Menczer, F. (2020). Scalable and generalizable social bot detection through data selection. In Proceedings of the 34th AAAI Conference on Artificial Intelligence, 34(1), 1096–1103.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I, Lucky Chigozie Ogogo

1 grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Comparative Analysis of Graph-Based Methods for Detecting Coordinated Cryptocurrency Scam Networks on Twitter/X”, supervised by Dimitrios Symeonidis.

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2 I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3 I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

[1] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Core code excerpts

This appendix gathers the most important code excerpts behind the experiments reported in Chapters 3 and 4. The excerpts are taken directly from the analysis package submitted with this thesis. Full source files are available in that package; the snippets below show only the parts that implement the methodological decisions discussed in the main text.

A2.1 Memory-efficient tweet streaming

The TwiBot-22 tweet files are JSON arrays of up to ten million records each. Rather than parsing each file with json.load, the pipeline reads it as a binary stream and splits on a known record boundary, running at C speed and keeping peak memory below 500 MB regardless of file size (Section 3.3.1).

```
def _iter_raw_records(path: Path, chunk_mb: int = 32) -> Iterator[bytes]:
    chunk_size = chunk_mb * 1024 * 1024
    carry = b""
    with path.open("rb") as fh:
        while True:
            data = fh.read(chunk_size)
            if not data:
                break
            buf = carry + data
            parts = buf.split(_RECORD_SEP)
            for idx, part in enumerate(parts[:-1]):
                if idx == 0:
                    raw = part.lstrip(b"[ \n\r") + b"}"
                else:
                    raw = b'{"attachments"' + part + b"}"
                if raw.startswith(b"{"):
                    yield raw
            carry = (b'{"attachments"' + parts[-1]) if len(parts) > 1 else buf
    if carry.startswith(b"{"):
        raw = carry.rstrip(b"] \n\r")
        if raw.endswith(b"}"):
            yield raw
```

Listing 1. Streaming tweet-record iterator (real_data_pipeline.py).

A2.2 A Homophily-aware interaction graph

The calibrated synthetic graph used as the dense-condition baseline in Section 3.7 is built with Zipf out-degrees and a homophily parameter that controls how often each edge connects two nodes of the same class. The relevant construction is reproduced below.

```
def build_interaction_graph(
    labels: np.ndarray,
    avg_degree: int = 5,
    homophily: float = 0.65,
    seed: int = RANDOM_SEED,
) -> nx.DiGraph:
    rng = np.random.default_rng(seed)
    n = len(labels)
    bot_indices = np.where(labels == 1)[0]
    hum_indices = np.where(labels == 0)[0]

    G = nx.DiGraph()
    G.add_nodes_from(range(n))
    nx.set_node_attributes(G, {i: int(labels[i]) for i in range(n)}, "label")

    raw_degrees = rng.zipf(1.8, n).astype(int)
    out_degrees = np.clip(raw_degrees, 1, max(10, avg_degree * 4))

    for src in range(n):
        k = out_degrees[src]
        src_is_bot = labels[src] == 1
        same_pool  = bot_indices if src_is_bot else hum_indices
        cross_pool = hum_indices if src_is_bot else bot_indices

        n_same  = min(rng.binomial(k, homophily), len(same_pool) - 1)
        n_cross = k - n_same

        targets: list[int] = []
        if len(same_pool) > 1:
            choices = rng.choice(same_pool,
                                 size=min(n_same, len(same_pool) - 1),
                                 replace=False)
            targets.extend(choices.tolist())
        if len(cross_pool) > 0 and n_cross > 0:
            choices = rng.choice(cross_pool,
                                 size=min(n_cross, len(cross_pool)),
                                 replace=False)
            targets.extend(choices.tolist())

        for tgt in targets:
            if tgt != src:
                G.add_edge(src, tgt)
    return G
```

Listing 2. Homophily-aware Zipf-degree graph construction (graph_comparison.py).

A2.3 SGC feature propagation

Simple Graph Convolution is implemented as a single sparse matrix multiplication with the symmetrically normalised adjacency matrix $\hat{A} = \tilde{D}^{(-1/2)}(A + I)\tilde{D}^{(-1/2)}$. The self-loop term in $\hat{A}$ is what gives SGC its graceful degradation on isolated nodes, as discussed in Section 4.4.

```
def _sgc_propagate(G: nx.DiGraph, X: np.ndarray, k: int = 2) -> np.ndarray:
    n = X.shape[0]
    A = nx.to_scipy_sparse_array(G.to_undirected(),
                                 nodelist=list(range(n)),
                                 format="csr")
    A = A + scipy.sparse.eye(n, format="csr")
    deg = np.asarray(A.sum(axis=1)).flatten()
    d_inv_sqrt = np.where(deg > 0, 1.0 / np.sqrt(deg), 0.0)
    D_inv_sqrt = scipy.sparse.diags(d_inv_sqrt, format="csr")
    A_hat = D_inv_sqrt @ A @ D_inv_sqrt

    H = StandardScaler().fit_transform(X).astype(np.float64)
    for _ in range(k):
        H = A_hat @ H
    return np.asarray(H)
```

Listing 3. Two-step SGC propagation via sparse matrix multiplication (graph_comparison.py).

A2.4 Node2Vec embedding generation

Node2Vec embeddings are generated with unbiased random walks (p = q = 1), walk length 30, ten walks per node and a context window of five the configuration described in Section 3.5.3.

```
from node2vec import Node2Vec

n2v = Node2Vec (
    G,
    dimensions=64,
    walk_length=30,
    num_walks=10,
    p=1, q=1,
    workers=1,
    quiet=True,
)
model = n2v.fit (window=5, min_count=1, batch_words=4)
X_n2v = np.array([model.wv[str(i)] for i in range(n)])
```

Listing 4. Node2Vec embedding generation (graph_comparison.py).

A2.5 Stratified five-fold evaluation

Every method is evaluated under the same protocol: five-fold stratified cross-validation with macro-averaged precision, recall, F1, accuracy and AUC-ROC. The random seed is fixed so that every reported number is reproducible from the raw inputs.

```
def _cv_metrics(clf, X: np.ndarray, y: np.ndarray,
                n_splits: int = 5) -> dict[str, float]:
    skf = StratifiedKFold(n_splits=n_splits, shuffle=True,
                          random_state=RANDOM_SEED)
    metrics: dict[str, list] = {k: [] for k in
        ["precision", "recall", "f1", "accuracy", "auc"]}

    for train_idx, test_idx in skf.split(X, y):
        X_tr, X_te = X[train_idx], X[test_idx]
        y_tr, y_te = y[train_idx], y[test_idx]
        clf.fit(X_tr, y_tr)
        y_pred = clf.predict(X_te)
        if hasattr(clf, "predict_proba"):
            y_prob = clf.predict_proba(X_te)[:, 1]
        else:
            y_prob = y_pred.astype(float)

        metrics["precision"].append(precision_score(y_te, y_pred,
            average="macro", zero_division=0))
        metrics["recall"].append(recall_score(y_te, y_pred,
            average="macro", zero_division=0))
        metrics["f1"].append(f1_score(y_te, y_pred,
            average="macro", zero_division=0))
        metrics["accuracy"].append(accuracy_score(y_te, y_pred))
        metrics["auc"].append(roc_auc_score(y_te, y_prob))

    return {k: float(np.mean(v)) for k, v in metrics.items()}
```

Listing 5. Stratified five-fold cross-validation metric computation (graph_comparison.py).