FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS ENGINEERING DEPARTMENT
UNDERGRADUATE COURSE IN CONTROL AND AUTOMATION ENGINEERING

Jean Guilherme Neiverth

Farcaster sybil account detection using graph-based and machine learning models

Florianópolis
2025

Jean Guilherme Neiverth

Farcaster sybil account detection using graph-based and machine learning models

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergraduate Course in Control and Automation Engineering of the Federal University of Santa Catarina.
Supervisor: Prof. Leandro Buss Becker, Dr.
Co-supervisor: José Fernando Rosa Ribeiro

Florianópolis
2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Neiverth, Jean Guilherme
Farcaster sybil account detection using graph-based and machine learning models / Jean Guilherme Neiverth ; orientador, Leandro Buss Becker, coorientador, José Fernando Rosa Ribeiro, 2025.
66 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2025.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Sybil Attack detection. 3. Graph-based algorithm. 4. Machine learning model. I. Becker, Leandro Buss. II. Ribeiro, José Fernando Rosa. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Jean Guilherme Neiverth

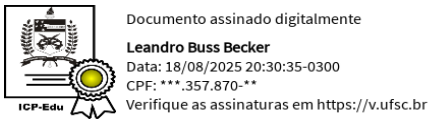
Farcaster sybil account detection using graph-based and machine learning models

This dissertation was evaluated in the context of the subject DAS5511 (Course Final Project) and approved in its final form by the Undergraduate Course in Control and Automation Engineering

Florianópolis, August 04-th, 2025.

Prof. Marcelo de Lellis Costa de Oliveira, Dr.
Course Coordinator

Examining Board:



Prof. Leandro Buss Becker, Dr.
Advisor
UFSC/CTC/DAS

José Fernando Rosa Ribeiro
Supervisor
Bleu Studio

Prof. Públio Macedo Monteiro Lima, Dr.
Evaluator
UFSC/CTC/DAS

Prof. Hector Bressa Silveira, Dr.
Board President
UFSC/CTC/DAS

ACKNOWLEDGEMENTS

I am deeply grateful to my parents, whose commitment to my education and unconditional support throughout my undergraduate journey made this achievement possible. To my brother, my companion during these years, thank you for standing by me through every challenge. I would also like to thank my girlfriend Emmanuele, who brought support into my life during this project's completion.

My professional growth owes much to José Ribeiro, whose mentorship opened doors to work alongside the exceptional Bleu team. I'm equally grateful to my colleagues there, who invested in both my personal development and the success of this project with remarkable generosity.

I Would also like to thank the Professors at DAS, for the educational excellence, elevating standards not only at UFSC but throughout Brazil. Special recognition goes to my advisor, Leandro Becker, whose guidance and constructive feedback proved crucial throughout this process.

To NEO Empresarial, that provided valuable connections and transformative experiences that shaped my professional identity in ways I continue to appreciate.

Finally, to my fellow students, especially Juliano Nunes, Juan Amorim, Luis Parise, Leonardo Clivati, and Ian Marchetti, thank you for your companionship through demanding semesters and for generously sharing your knowledge along the way.

DISCLAIMER

Florianópolis, August 04-th, 2025.

As representative of the Bleu Studio company in which the present work was carried out, I declare this document to be exempt from any confidential or sensitive content regarding intellectual property, that may keep it from being published by the Federal University of Santa Catarina (UFSC) to the general public, including its online availability in the Institutional Repository of the University Library (BU). Furthermore, I attest knowledge of the obligation by the author, as a student of UFSC, to deposit this document in the said Institutional Repository, for being it a Final Program Dissertation (*"Trabalho de Conclusão de Curso"*), in accordance with the *Resolução Normativa n° 126/2019/CUn*.

José Fernando Rosa Ribeiro
Bleu Studio

ABSTRACT

Farcaster, the leading blockchain-based social network with over 800,000 registered accounts, faces significant challenges from Sybil attacks where fraudulent users create multiple fake accounts to exploit financial incentives and distribute malicious content. These attacks degrade user experience through meaningless content and create administrative burdens. This thesis presents a mixed algorithm approach combining a graph-based algorithm with machine learning models to detect Sybil accounts on Farcaster. The methodology employs SybilSCAR as the graph-based component, selected for its balance of performance and execution time. The machine learning component utilizes an ensemble of LightGBM, Random Forest, and XGBoost models, incorporating features such as text analysis metrics, behavioral patterns, and network structure indicators. The approach was evaluated on a dataset of approximately 4,000 Sybil samples, alongside 400 manually verified human accounts. The hybrid approach achieved superior performance with a ROC AUC score of 0.99, compared to 0.95 for the standalone graph-based algorithm and 0.98 for the machine learning model alone. This work reinforces the academic finding that mixed approaches outperform single-method solutions for Sybil detection. The practical implications include enabling Farcaster clients like Warpcast to more efficiently identify and filter fraudulent accounts, significantly enhancing the user experience across the platform.

Keywords: Sybil detection. Graph-based algorithm. Machine learning.

RESUMO

Farcaster, a principal rede social baseada em blockchain com mais de 800 mil contas registradas, enfrenta desafios significativos de ataques *Sybil*, onde usuários fraudulentos criam múltiplas contas falsas para explorar incentivos financeiros e distribuir conteúdo malicioso. Esses ataques degradam a experiência do usuário através de conteúdo sem sentido e criam encargos administrativos. Este projeto apresenta uma abordagem que consiste na combinação de um algoritmo baseado em grafos e modelos de aprendizado de máquina para detectar contas *Sybil* no Farcaster. A metodologia emprega o *SybilSCAR* como componente baseado em grafos, selecionado por seu equilíbrio entre assertividade e tempo de execução. O componente de aprendizado de máquina utiliza um conjunto de modelos *LightGBM*, *Random Forest* e *XGBoost*, incorporando características como métricas de análise de texto, padrões comportamentais e indicadores de estrutura de rede. A abordagem foi avaliada em um conjunto de dados de aproximadamente 4 mil amostras *Sybil*, juntamente com 400 contas humanas verificadas manualmente. A abordagem híbrida alcançou desempenho superior com uma pontuação ROC AUC de 0,99, comparado a 0,95 para o algoritmo baseado em grafos isolado e 0,98 para o modelo de aprendizado de máquina sozinho. Este trabalho reforça a descoberta acadêmica de que abordagens mistas superam soluções de método único para detecção de *Sybils*. As implicações práticas incluem permitir que clientes do Farcaster como o Warpcast identifiquem e filtrem contas fraudulentas de forma mais eficiente, melhorando significativamente a experiência do usuário em toda a plataforma.

Palavras-chave: Detecção de *Sybils*. Algoritmo baseado em grafo. Aprendizado de máquina.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under the Curve
CRISP-DM	Cross-Industry Standard Process for Data Mining
DeFi	Decentralized Finance
EAS	Ethereum Attestation Service
ENS	Ethereum Name Service
FID	Farcaster ID
KNN	K-Nearest Neighbors
LBP	Loopy Belief Propagation
LightGBM	Light Gradient Boosting Machine
LLM	Large Language Model
ML	Machine Learning
NFT	Non-Fungible Token
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

CONTENTS

1 INTRODUCTION

Social media adoption has grown substantially over the past decade, transforming digital communication and information sharing patterns. Concurrently, the deployment of automated agents and bots across digital platforms has increased significantly. While many autonomous systems provide beneficial services such as customer support, news aggregation, and market analysis, their widespread adoption has intensified the problem of Sybil attacks, where malicious actors create networks of fake accounts to manipulate platform operations and exploit system vulnerabilities.

Farcaster, as the leading blockchain-based social network, faces distinctive security challenges due to its decentralized architecture. The platform's decentralized design provides censorship resistance and user data ownership while creating an environment that accommodates autonomous agent operations. This same architectural openness that enables beneficial automation also increases susceptibility to sophisticated Sybil attacks, as decentralized systems limit traditional centralized moderation approaches while blockchain infrastructure facilitates the creation and coordination of multiple fake identities with reduced barriers to entry.

Sybil attacks cause measurable harm across the Farcaster community through three primary channels. Users experience reduced platform quality as spam content, promotional schemes, and inauthentic interactions populate their social feeds, diminishing genuine social engagement. Platform administrators and community moderators allocate substantial time and resources to identifying, reporting, and managing Sybil accounts rather than focusing on community development and platform enhancement. External companies and protocols that provide financial incentives for organic content sharing inadvertently reward coordinated networks of fake accounts, resulting in economic losses and distorted engagement metrics that compromise the effectiveness of their community-building initiatives.

The primary objective of this project is to develop an algorithm for accurate Sybil account detection within the Farcaster ecosystem through probabilistic user authenticity assessments. The solution generates Sybil probability scores from 0 (high confidence in legitimacy) to 1 (high likelihood of Sybil behavior) for individual users, with results accessible through a public Application Programming Interface (API) that enables integration across the Farcaster ecosystem by developers, researchers, and community tools.

The implemented solution combines SybilSCAR, a graph-based algorithm that analyzes social network structure and trust propagation, with an ensemble of machine learning classifiers including Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), and Random Forest models that evaluate behavioral patterns and content characteristics. This hybrid methodology achieved AUC score of

0.99 on the evaluation dataset. The complete implementation includes a public API for programmatic access, an interactive Farcaster Frame interface for direct user queries within the social feed, and a monitoring dashboard that tracks algorithm performance and system metrics.

This work contributes empirical confirmation that hybrid approaches combining graph-based and machine learning techniques outperform individual algorithmic solutions for Sybil detection, particularly against sophisticated attackers. While the employed techniques are established in existing literature, this study demonstrates their effectiveness against advanced Sybil strategies, including accounts using Large Language Model (LLM)s for human-like content generation and complex behavioral patterns. The successful application to Farcaster's ecosystem, where many Sybil accounts employ Artificial Intelligence (AI)-generated content and sophisticated tactics, validates the robustness of combined detection approaches against evolving attack methodologies.

The completion of this project enables multiple stakeholders within the Farcaster ecosystem to implement enhanced automated content filtering mechanisms. Social network managers and client developers can integrate Sybil probability scores into their platforms to reduce spam visibility and improve user experience through intelligent content curation. Individual users gain access to tools for evaluating account authenticity, supporting informed decisions about social interactions and content consumption. Companies and protocols providing financial incentives for community engagement can implement additional verification layers to identify suspect accounts before reward distribution, protecting their investment in genuine community building while maintaining incentive program integrity.

The following sections establish the foundational context for this research by examining four key components: Farcaster as the blockchain-based social network in the project scope, the Optimism ecosystem and its financial incentive mechanisms, Bleu Studio as the development company, and the central problem of Sybil attacks exploiting reward systems through fake account networks. The chapter examines existing detection solutions and their limitations, establishing the need for an improved Sybil detection algorithm that can operate effectively in the Farcaster environment.

1.1 FARCASTER

Farcaster represents the leading blockchain-based social network, with over 800,000 registered accounts. It operates through a casting system that bears functional similarities to platforms like X (formerly Twitter) and Threads, where users publish short-form content called "casts" that can be liked, recasted (shared), and replied to by other users (see Figure 1). Built on Ethereum's Optimism Layer 2 blockchain, it positions itself as a sufficiently decentralized alternative to traditional social media platforms

(FARCASTER, 2025).

However, Farcaster distinguishes itself from conventional social networks through several key architectural and operational differences. The platform operates as a decentralized protocol rather than a centralized service, meaning user data and social graphs are stored on the blockchain and controlled by users rather than a single corporation. Figure 2 illustrates this decentralized architecture. This design makes the network inherently censorship-resistant, as no central authority can unilaterally remove content or ban users from the protocol itself.

Figure 1 – Cast example in Farcaster.

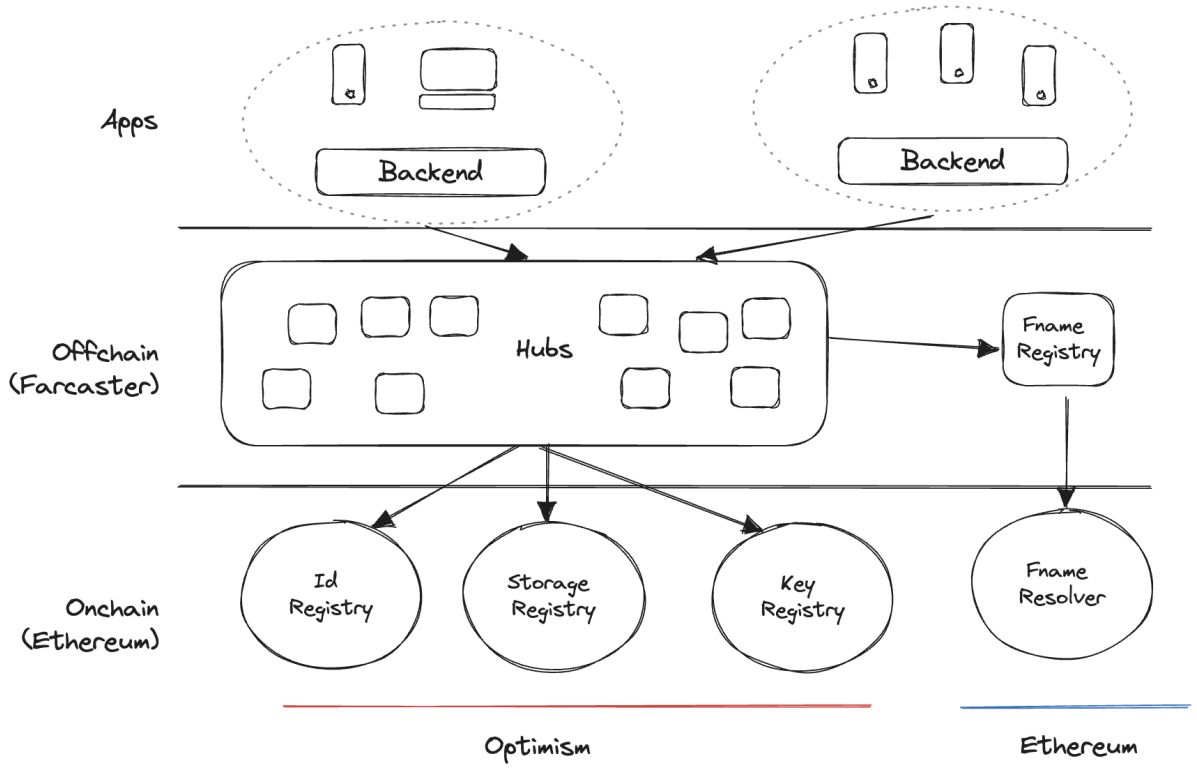
Source: Obtained from Warpcast client.

One of Farcaster's most innovative features is Frames—interactive apps embedded within casts that allow users to perform actions directly within the social feed without leaving the platform. These Frames can facilitate activities such as voting in polls, minting Non-Fungible Token (NFT)s, making purchases and interacting with other decentralized applications, creating a more integrated Web3 experience than traditional social platforms.

Unlike most social networks that offer free account creation, Farcaster requires users to pay a small fee to register an account and obtain a Farcaster ID (FID). This economic barrier ensures users have genuine commitment to the platform and provides sustainable funding for network operations. Each user receives a unique FID that serves as their immutable identity across the network, regardless of username changes.

The platform's primary community centers around cryptocurrency and blockchain technology, with users frequently discussing Decentralized Finance (DeFi) protocols, NFT projects, governance proposals, and market analysis. However, the network has expanded to encompass diverse communities organized through its channel system, including channels for art, technology, gaming, and general discourse. These channels function similarly to Discord servers or Reddit communities, allowing users to follow specific topics and engage with like-minded individuals.

Figure 2 – Farcaster architecture illustration.



Source: (FARCASTER, 2025)

Farcaster's decentralized architecture enables multiple client applications to interact with the same underlying protocol and user base, serving as the actual frontend interface users interact with. While Warpcast is the primary and most popular frontend client, developed by the Farcaster team itself, alternative clients such as Supercast, Herocast, and others offer different user interfaces and feature sets while accessing the same social graph and content. This client diversity prevents platform lock-in and allows for innovation in user experience design.

The platform's integration with Web3 infrastructure allows seamless connection with cryptocurrency wallets, enabling users to display their token holdings, NFT collections, and transaction history as part of their social identity. This deep Web3 integration has made Farcaster a natural gathering place for crypto enthusiasts and a testing ground for new blockchain-based social features, while also making it an attractive target for various forms of exploitation and manipulation.

1.2 OPTIMISM

Optimism is a blockchain infrastructure that processes transactions faster and cheaper than Ethereum main network, the underlying environment. While Ethereum

transactions can cost dozens of dollars and take minutes to complete, Optimism reduces these costs by 10-100 times and processes transactions in seconds, while maintaining the same security standards. This efficiency enables companies and blockchain projects to provide frequent, small-value digital rewards to Farcaster users for social engagement activities such as content creation, platform participation, and community interaction. The low transaction costs make these micro-reward systems economically viable at scale.

While many Layer 2 solutions focus primarily on financial applications, Optimism has strategically invested in diverse use cases that extend beyond financial services. The ecosystem encompasses financial protocols, but also implements social applications like Farcaster, identity solutions through the Ethereum Attestation Service (EAS), and various consumer-facing applications. This diversification reflects Optimism's broader vision of creating a sustainable ecosystem where "impact equals profit"—rewarding builders who create genuine value for users and the broader crypto community.

Farcaster's success demonstrates Optimism's viability for consumer applications beyond DeFi, while its integration with the broader crypto ecosystem creates natural synergies with other Optimism-based projects. This relationship has fostered a vibrant economy of financial incentives where blockchain projects, DeFi protocols, and crypto companies actively reward Farcaster users who engage with their platforms or mention their activities.

These financial incentive mechanisms typically operate through several channels: direct rewards to active Farcaster users, digital influencers who cast about specific projects, bounties for creating educational content, and participation rewards for engaging with protocol governance. For example, many decentralized financial protocols have allocated portions of their token supplies to reward Farcaster users who demonstrate genuine engagement with their platforms, while venture capital firms and crypto companies often sponsor campaigns that reward users for sharing insights or participating in discussions about their portfolio companies.

1.3 BLEU STUDIO

Founded in 2022 by former students of the Control and Automation Engineering program at the Federal University of Santa Catarina (UFSC), Bleu Studio is specialized in Web3-related software development and solutions. The company's mission centers on open-source development and simplifying complex Web3 processes, making blockchain technology more accessible to developers and end users.

This approach has enabled Bleu Studio to establish strategic partnerships that include Balancer, CoW Protocol and Optimism. Through these collaborations, Bleu Studio has delivered comprehensive solutions spanning multiple domains: full-stack

software development, advanced data analysis, and AI-powered automation tools.

Bleu Studio's relationship with Optimism encompasses various project types that demonstrate the company's versatility and deep ecosystem understanding. Previous collaborations have included front-end application development, AI assistant implementation, and ongoing partnership agreements in which Bleu Studio provides dedicated technical support in multiple improvement areas over one year.

The current project, formally known as the Farcaster Social Graph, represents a significant departure from Bleu Studio's traditional Web3 development work, requiring expertise in machine learning algorithms and algorithm research. This project became feasible due to the diverse backgrounds of Bleu Studio's team members. The interdisciplinary nature of this challenge aligns perfectly with Bleu Studio's founding philosophy of tackling complex problems through innovative technical approaches and collaborative open-source development.

1.4 PROBLEM DESCRIPTION

Recent advances in web scraping, artificial intelligence, and digital automation have led to a proliferation of autonomous agents across social networks and other online platforms. These AI-powered entities can publish daily stock summaries, promotional content, weather alerts, and provide customer service through chatbots, reflecting a natural evolution in web application design that increasingly accommodates nonhuman participants.

The Ethereum blockchain exemplifies this trend, offering an environment where automated agents operate with the same freedoms as human users. Some of these agents have gained influential status in the Web3 ecosystem, offering investment advice and executing cryptocurrency transactions in near real time. This technological capability presents both opportunities and risks for blockchain-based platforms.

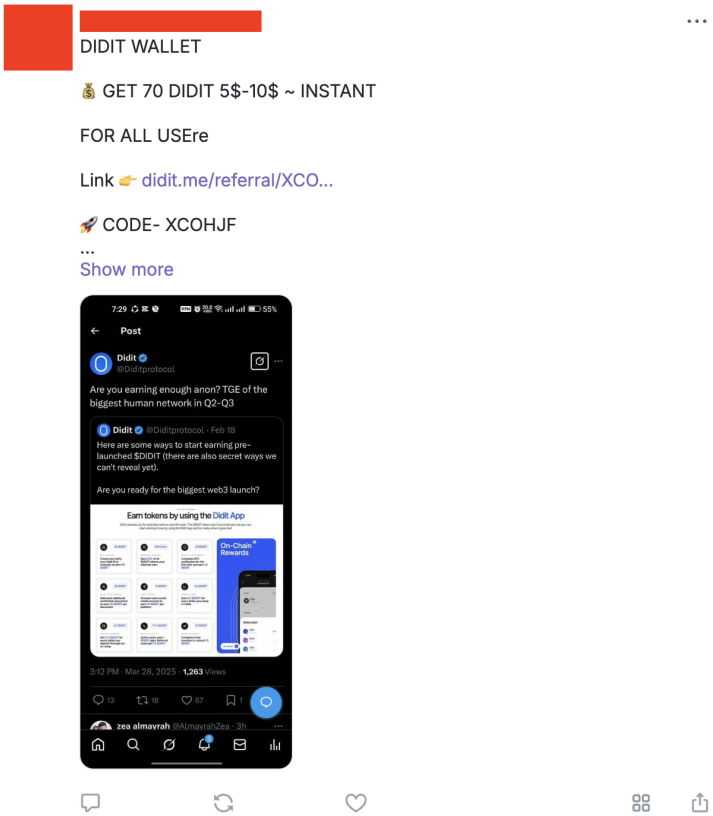
Farcaster's decentralized architecture, as described in section 1.1, inherently supports beneficial automated agents and bots that provide valuable services to the community. These legitimate agents contribute to the platform's ecosystem by sharing market insights, providing technical analysis, or just casting relevant information with a certain frequency. However, this same openness that enables beneficial automation also creates vulnerabilities for malicious exploitation.

The platform's susceptibility to Sybil attacks, combined with the financial incentive mechanisms on Optimism 's ecosystem detailed in section 1.2, created an environment in which malicious actors exploit these reward systems through Sybil attacks, the creation of multiple fake accounts designed to manipulate algorithms and gain unfair financial advantages. These attackers employ two primary strategies: first, they create networks of interconnected bot accounts that follow and interact with each other to collect multiple rewards from the various airdrops, bounties, and participation incentives

offered by DeFi protocols and crypto companies (Figure 3). Second, they generate coordinated campaigns using numerous fake accounts to promote worthless cryptocurrencies through false positive news and fabricated results (exemplified in Figure 4).

These Sybil attacks create cascading negative impacts across the entire ecosystem. Farcaster users experience degraded platform quality as their timelines become populated with meaningless content, while they must also navigate unwanted friend invitations from fake accounts and need to be more careful when consuming platform content. The attacks place significant operational strain on front-end clients like Warpcast, whose teams face Sybil reports and must continuously develop countermeasures to reduce the visibility of these malicious accounts—a particularly challenging task given Farcaster's censorship-resistant design outlined in section 1.1. Additionally, the protocols and companies offering financial rewards suffer economic harm as they pay substantial amounts for fraudulent engagement between Sybil accounts rather than genuine user participation.

Figure 3 – Sybil-like spam cast example.

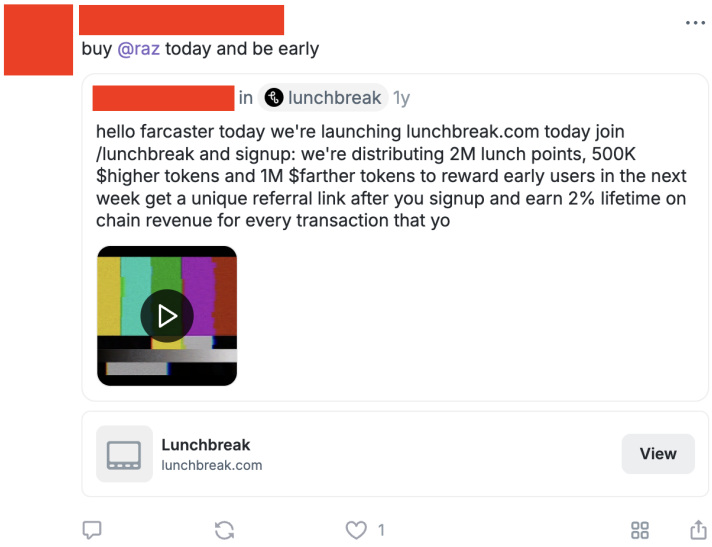


Source: Obtained from Warpcast client.

1.5 EXISTING COUNTERMEASURES

While Sybil attacks pose significant challenges to the Farcaster ecosystem, several countermeasures have emerged to address these threats. These existing solutions

Figure 4 – Sybil-like financial-related cast example.



Source: Obtained from Warpcast client.

employ different methodologies, ranging from content analysis to behavioral scoring and identity verification, each with distinct strengths and limitations in combating malicious account networks.

1.5.1 Bot or not

Bot or Not represents one prominent approach, a Farcaster application that operates as an interactive Frame, allowing users to analyze any cast to determine whether the author exhibits bot-like or human-like patterns (see example in Figure 5). The system relies heavily on LLM analysis and responds to user queries by casting an analysis summary through the official Bot or Not account. However, this approach has limitations: research has demonstrated that accurately distinguishing human-written and AI-generated text is still a challenge (WU et al., 2025), making Bot or Not effective only against basic spammers while failing to detect sophisticated LLM-based bots. Additionally, the system identifies bot behavior rather than malicious intent, meaning users cannot reliably use its assessments to justify blocking or banning accounts.

1.5.2 Neynar User Scores

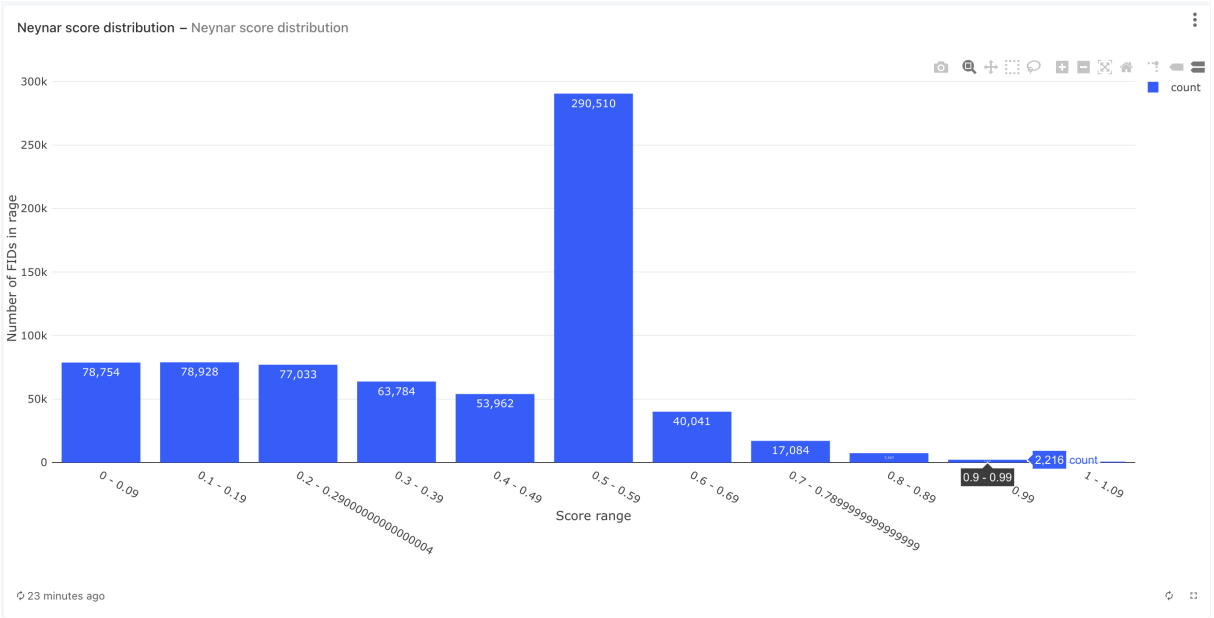
The Neynar User Score system employs a comprehensive behavioral analysis approach, assigning users scores between 0 and 1 based on account quality metrics rather than humanity verification. This scoring mechanism evaluates multiple behavioral factors including content originality, engagement authenticity, posting frequency, and interaction patterns with other high-quality users. The system updates scores weekly, allowing for dynamic assessment as user behavior evolves over time.

Figure 5 – Bot or Not analysis example.

Source: Obtained from Warpcast client.

According to Neynar's documentation, the scoring algorithm specifically measures account value added to the network and demonstrates capability in discriminating between high and low quality activity (NEYNAR, 2025). For instance, legitimate automated agents like Bracky and Clanker achieve high scores due to their valuable contributions, while accounts generating spam content receive lower ratings. The score distribution reveals that approximately 2,500 accounts maintain scores above 0.9, while 27,500 accounts score above 0.7 (see Figure 6), indicating the system's selectivity in identifying high-quality participants.

Figure 6 – Neynar Users Scores distribution.



Source: (NEYNAR, 2025)

However, the Neynar User Score system presents several limitations for Sybil detection purposes. The scoring methodology heavily emphasizes popularity metrics and

social proof, requiring substantial follower counts, engagement rates, and visibility to achieve top scores. This popularity bias creates scenarios where legitimate but lesser-known users may receive low scores despite authentic behavior, while sophisticated Sybil networks that successfully cultivate artificial popularity could potentially achieve higher ratings. Furthermore, the system explicitly focuses on content quality and network value rather than malicious intent detection, meaning low scores may reflect poor content quality or limited engagement rather than Sybil behavior.

1.5.3 Human verification mechanisms

Identity verification systems represent another category of Sybil countermeasures, with Ethereum Name Service (ENS) domains and Gitcoin Passport serving as prominent examples of human verification mechanisms within the Web3 ecosystem. These systems attempt to establish proof of humanity and unique identity through various verification processes and credential accumulation.

ENS domains function as blockchain-based identity markers that map human-readable names with '.eth' domain to Ethereum addresses and associated metadata (ENS, 2025). The ENS system operates through a hierarchical structure where domain ownership is secured by the Ethereum blockchain, creating verifiable digital identities that users can associate with their social media accounts. ENS domains require payment for registration and renewal, creating an economic barrier that may deter some Sybil attackers from acquiring multiple domains for fake accounts.

Gitcoin Passport implements a more comprehensive verification approach through its Unique Humanity Scorer tool, which allows users to accumulate verifiable credentials called "Stamps" from various Web2 and Web3 platforms (LAND, 2025). These stamps include social media account verifications, GitHub contributions, transaction history, and other digital footprints that collectively contribute to a Unique Humanity Score. Organizations can then require minimum scores as gatekeeping mechanisms, with Gitcoin recommending scores of 20 or higher for effective Sybil defense.

Despite their utility, both verification mechanisms face significant limitations in comprehensive Sybil detection. ENS domain ownership primarily indicates economic investment rather than humanity verification, as motivated attackers can purchase multiple domains to legitimize Sybil accounts. The system also suffers from low adoption rates within the broader Farcaster community, limiting its effectiveness as a universal verification standard.

Gitcoin Passport, while more comprehensive in its verification approach, relies heavily on existing Web2 and Web3 platforms that may themselves be vulnerable to manipulation. Sophisticated attackers can potentially manipulate the stamp collection process by creating profiles across multiple platforms over extended time periods. Additionally, both systems require proactive user participation and technical knowledge,

creating barriers for legitimate users while not preventing determined malicious actors from meeting verification requirements. Most importantly, these verification mechanisms address identity authenticity but do not directly detect coordinated behavioral patterns or malicious intent that characterize sophisticated Sybil networks operating with verified identities.

1.6 DOCUMENT OVERVIEW

Given this context, the entire Farcaster ecosystem would benefit from a publicly available algorithm capable of accurately detecting Sybil accounts. A solution like that would enable Warpcast and other clients to implement more effective preventive measures against Sybil content visibility, allow reward-providing companies to implement targeted blocking and enhanced security protocols for flagged accounts, and empower individual users to independently identify and block suspicious accounts. In response to this community need, Bleu Studio has undertaken this project to develop a comprehensive Sybil detection algorithm that addresses these challenges.

A theoretical background is presented in Chapter 2 with the foundation for Sybil detection research, examining the evolution of Sybil attacks from their initial identification to their current manifestation in social networks. The chapter reviews two primary detection approaches: graph-based algorithms that leverage social network structure and trust propagation, and machine learning classifiers that analyze behavioral patterns and content characteristics. The review also explores semi-supervised learning techniques, particularly self-training approaches, which address the challenge of limited labeled data in social network analysis.

Chapter 3 outlines the proposed hybrid solution that combines graph-based and machine learning approaches to generate probabilistic Sybil scores for Farcaster users. The methodology encompasses four key components: evaluation dataset construction through systematic sampling and manual verification, comparative assessment of graph-based algorithms, machine learning model development using the CRISP-DM framework, and system deployment with public API access, front-end interface and monitoring dashboard. The solution targets specific performance metrics including minimum 0.7 AUC score, sub-500ms API response times, and daily batch processing capabilities for the entire user base.

The development and evaluation process is shown in Chapter 4, which begins with the creation of a labeled dataset containing 4,416 samples through systematic evaluation of existing detection systems and manual verification processes. The implementation demonstrates that SybilSCAR outperformed other graph-based algorithms with 0.95 AUC, while the machine learning ensemble achieved 0.98 AUC. The final hybrid system reached 0.99 AUC performance and includes production deployment with weekly batch processing, public API access, interactive Farcaster Frame interface,

and comprehensive monitoring dashboard.

Finally, Chapter 5 summarizes the project achievements, confirming that all primary objectives were met with the hybrid system exceeding performance requirements and providing practical utility to the Farcaster ecosystem. The discussion identifies key challenges including static labeling limitations, scalability concerns for future growth, and vulnerability to adversarial adaptation. The work validates the effectiveness of hybrid detection approaches in blockchain-based social networks and establishes a foundation for ongoing research in decentralized platform security.

2 THEORETICAL BACKGROUND

This section presents a comprehensive review of academic studies to establish a theoretical foundation for the proposed solution. The review is structured in three parts: first, state-of-the-art research was examined to understand the concept of Sybil attacks, their occurrence patterns, and primary countermeasures. Second, specific articles focused on the two dominant detection approaches (graph-based algorithms and machine learning models) were analyzed to identify suitable algorithms for implementation. Third, machine learning classifiers are explored to address the challenge of limited labeled data in social network analysis, examining self-training approaches that can leverage abundant unlabeled data to enhance detection performance.

2.1 SYBIL ATTACKS

Sybil attacks represent one of the most fundamental security threats in distributed systems and online social networks (AL-QURISHI et al., 2017). Named after the book "Sybil," which describes a person diagnosed with dissociative identity disorder, this attack was first formally identified by Douceur in 2002 in the context of peer-to-peer systems. In a Sybil attack, a malicious entity creates and controls multiple fake identities (Sybil nodes) within a network, leveraging these fabricated accounts to gain disproportionate influence over system operations and compromise the integrity of distributed protocols (ZHANG, K. et al., 2014).

A comprehensive survey on Sybil defense techniques in online social networks was carried out by (AL-QURISHI et al., 2017). The paper systematically reviews and categorizes various approaches to defend against Sybil attacks on platforms such as Facebook, Twitter, and YouTube. The authors analyze how these attacks exploit vulnerabilities in social networks and present a taxonomy of defense mechanisms. They classified defense mechanisms in 4 main groups: graph-based methods, machine learning approaches, manual verification, and prevention approaches.

The term "Sybil" has since become synonymous with any form of identity manipulation in distributed systems where a single adversary can masquerade as multiple distinct entities (ZHANG, K. et al., 2014). In the context of online social networks, Sybil attacks manifest as the creation of fake user accounts that appear legitimate but are actually controlled by a single malicious actor or coordinated group of attackers.

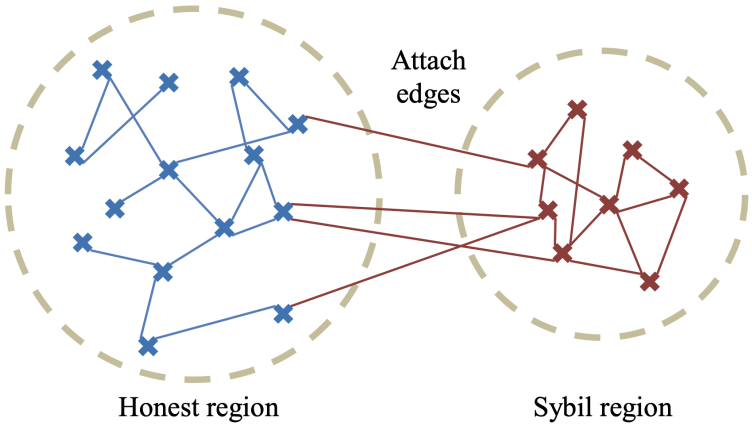
Sybil attacks are characterized by distinctive patterns in identity creation, content behavior, and social network infiltration that differentiate them from legitimate users. Attackers exploit the lightweight registration processes of social platforms, requiring only basic information like email addresses that can be easily fabricated at scale, allowing them to create multiple fake identities with minimal resource investment. Unlike authentic users who display diverse and organic behaviors, Sybil accounts exhibit predictable

patterns including spam generation, misinformation spreading, automated interactions, and repetitive high-frequency activities that create detectable anomalies in their profiles (JAVED et al., 2025).

Sybil attacks in online social networks are primarily motivated by financial gain, information warfare, privacy violations, and reputation system manipulation (AL-QURISHI et al., 2017). Attackers create fake accounts to manipulate marketplaces and review systems for economic benefit, spread misinformation and propaganda to influence public opinion or elections, harvest personal data by establishing false trust relationships, and game collective rating systems to skew democratic processes. Sybil accounts tend to connect more between themselves than healthy users, which usually creates separate honest and Sybil zones, with attack edges on benign accounts that eventually interact with the Sybil region (Figure 7).

The cascading effects of Sybil attacks significantly compromise system security and user experience across multiple dimensions (AL-QURISHI et al., 2017). These attacks distort recommendation algorithms and content ranking systems, erode fundamental trust relationships between users, create economic inefficiencies by wasting advertising resources on fake engagement, and prompt increased regulatory scrutiny that raises operational costs for platform operators. The cumulative impact undermines both the technical integrity and social value of online networks, forcing platforms to invest heavily in identity verification and content moderation systems.

Figure 7 – Sybil attack common regions on networks.



Source: (AL-QURISHI et al., 2017).

Sybil attacks have evolved significantly since their initial identification, becoming increasingly sophisticated and difficult to detect. Modern attackers employ advanced techniques including machine learning algorithms to create more convincing fake profiles, natural language processing to generate authentic-seeming content, and social engineering tactics to build credible social connections (CÁRDENAS-HARO et al., 2024).

Furthermore, the integration of artificial intelligence and machine learning into attack methodologies has enabled the creation of Sybil accounts that can adapt their behavior in real-time to evade detection systems. This arms race between attackers and defenders continues to drive innovation in both attack techniques and defense mechanisms (CÁRDENAS-HARO et al., 2024).

2.2 GRAPH-BASED ALGORITHMS

Graph-based algorithms represent the most prominent approach for Sybil detection in online social networks, leveraging the structural properties of social graphs to distinguish between legitimate users and Sybil accounts. These methods are founded on the homophily assumption—that benign users and Sybil accounts form distinct regions with sparse connections between them, while maintaining dense connections within their respective communities.

Graph-based Sybil detection methods operate on the principle that social networks can be modeled as undirected graphs G = (V, E), where nodes represent users and edges represent social relationships. The fundamental assumption underlying these approaches is that the benign region and Sybil region are sparsely connected compared to the internal connections within each region. This creates a natural bottleneck between honest users and Sybil accounts, limiting the attack edges that adversaries can establish.

2.2.1 SybilSCAR

One existing technique to address Sybil detection in online social networks is SybilSCAR (WANG; ZHANG, L.; GONG, 2017), an algorithm that introduces a structure-based approach that unifies Random Walk and Loopy Belief Propagation methods through a local rule framework that iteratively propagates label information across the social graph. By leveraging the homophily principle—the assumption that connected users (for example, when two users mutually follow each other) tend to share the same label as either benign or Sybil—SybilSCAR can effectively distinguish between legitimate and fake accounts in social networks (Algorithm 1 and Table 1).

The algorithm demonstrated great performance with AUC scores of 1.00 on Facebook, 0.99 on small Twitter datasets, and 0.82 on large Twitter networks. SybilSCAR's key advantages include its scalability, guaranteed convergence, and robustness to label noise, while being orders of magnitude more efficient than traditional Loopy Belief Propagation (LBP)-based approaches. However, the algorithm faces limitations in its dependency on the homophily assumption, which requires sparse connections between benign and Sybil regions, and its need to receive a set of known benign and Sybil labels as input. Similarly to SybilSCAR, this algorithm also requires a list of known Sybil and

benign users to be set on the input (Figure 8).

Algorithm 1 SybilSCAR algorithm.

procedure SYBILSCAR($G, L_s, L_b, \theta, w, \varepsilon, T$) ▷ Sybil detection in social networks
  Initialize prior probabilities:
  for $u \in V$ **do**
    if $u \in L_s$ **then**
      $q_u \leftarrow \theta$ ▷ Labeled Sybil
    else if $u \in L_b$ **then**
      $q_u \leftarrow 1 - \theta$ ▷ Labeled benign
    else
      $q_u \leftarrow 0.5$ ▷ Unlabeled node
  $\hat{p}^{(0)} \leftarrow \hat{q}$
  $t \leftarrow 1$
  while $\frac{\|\hat{p}^{(t)} - \hat{p}^{(t-1)}\|_1}{\|\hat{p}^{(t)}\|_1} \geq \varepsilon$ **and** $t \leq T$ **do**
    $\hat{p}^{(t)} \leftarrow \hat{q} + 2\hat{w}A\hat{p}^{(t-1)}$ ▷ Update using local rule
    $t \leftarrow t + 1$ ▷ Increment iteration
  return $\hat{p}^{(t)} + 0.5$ ▷ Convert residual to actual probabilities

Source: Based on Wang, L. Zhang, and Gong (2017).

Table 1 – SybilSCAR algorithm notation.

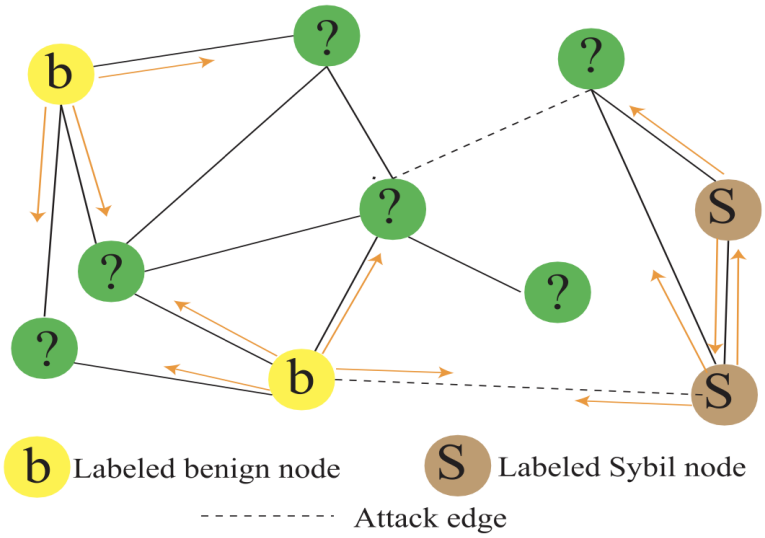
Notation	Description
$G = (V, E)$	Social network graph with vertices V and edges E
L_s	Set of labeled Sybil nodes
L_b	Set of labeled benign nodes
θ	Prior probability parameter for labeled nodes
w	Edge weight parameter
ε	Convergence threshold
T	Maximum number of iterations
q_u	Prior probability that node u is a Sybil
$\hat{q}$	Residual prior probability vector
$\hat{p}^{(t)}$	Residual posterior probability vector at iteration t
A	Adjacency matrix of the social graph
$\hat{w}$	Residual homophily strength

Source: Based on Wang, L. Zhang, and Gong (2017).

2.2.2 SybilBelief

Another approach is SybilBelief (GONG; FRANK; MITTAL, 2014), a semi-supervised learning framework designed to detect Sybil attacks in social networks and distributed systems. The algorithm uses Markov Random Fields combined with Loopy Belief Propagation to propagate label information from a small set of known benign and Sybil nodes

Figure 8 – SybilSCAR and SybilBelief start nodes.



Source: (GONG; FRANK; MITTAL, 2014).

to all other nodes in the network. The algorithm is presented in Algorithm 2 and the notation can be seen in Table 2.

Based on that study, SybilBelief's can tolerate up to 49% incorrect labels), has a scalability with O(n*d) complexity, and has robustness to network community structures while requiring only one label per community. However, the method also faces challenges in requiring social networks that satisfy the homophily assumption, where connected nodes tend to have the same labels, and depends on obtaining trust-based social connections rather than arbitrary friendships to maintain detection accuracy.

2.2.3 SybilRank

The authors of the last mapped technique propose an enhanced detection algorithm that improves upon existing method SybilRank by addressing two critical shortcomings: the concentration of trust seeds in specific communities and the vulnerability to sophisticated attackers who artificially increase common friends between Sybil and honest nodes (HARUTA; TOYODA; SASASE, 2016). The focus is Sybil attacks in social networking services like Facebook and Twitter, for scenarios in which attackers create lots of fake accounts to spam and harass honest users. Their approach combines community detection using a fast greedy algorithm with robust seed selection from each detected community, followed by graph pruning based on a "Trusted Area" concept that evaluates relationship density with trusted nodes rather than simply counting common friends (see Algorithm 3 and Table 3).

The algorithm demonstrates significant improvements in detection accuracy, particularly against sophisticated attacks, achieving AUC values around 0.9 in standard

Algorithm 2 SybilBelief algorithm

procedure SYBILBELIEF($G, L_b, L_s, w, \varepsilon, d_{max}$) ▷ Semi-supervised Sybil detection
  Initialize node potentials:
  for $v \in V$ **do**
    if $v \in L_b$ **then**
      $\theta_v \leftarrow 0.9$ ▷ Known benign node
      Fix $x_v = +1$
    else if $v \in L_s$ **then**
      $\theta_v \leftarrow 0.1$ ▷ Known Sybil node
      Fix $x_v = -1$
    else
      $\theta_v \leftarrow 0.5$ ▷ Unlabeled node (neutral prior)
  Initialize edge potentials:
  for $(u, v) \in E$ **do**
    $w_{uv} \leftarrow w$ ▷ Homophily strength parameter
  Initialize messages:
  for $(u, v) \in E$ **do**
    $m_{uv}^{(0)}(x_v) \leftarrow 1$ for $x_v \in \{-1, +1\}$
  $t \leftarrow 1$
  while $t \leq d_{max}$ **and** convergence not reached **do**
    for $(u, v) \in E$ **do** ▷ Message passing iteration
      for $x_v \in \{-1, +1\}$ **do**
        $m_{uv}^{(t)}(x_v) \leftarrow \sum_{x_u \in \{-1,+1\}} \Phi_u^L(x_u)\Phi_{uv}(x_u, x_v) \prod_{k \in \mathcal{N}(u)\setminus\{v\}} m_{ku}^{(t-1)}(x_u)$
      Normalize $m_{uv}^{(t)}(x_v)$
    Check convergence: $\sum_{(u,v)\in E} |m_{uv}^{(t)}(x_v) - m_{uv}^{(t-1)}(x_v)| < \varepsilon$
    $t \leftarrow t + 1$
  Compute posterior probabilities:
  for $v \in V \setminus (L_b \cup L_s)$ **do** ▷ Only for unlabeled nodes
    $P(x_v = +1|\bar{x}_L) \propto \phi_v(+1) \prod_{u \in \mathcal{N}(v)} m_{uv}^{(t)}(+1)$
    $P(x_v = -1|\bar{x}_L) \propto \phi_v(-1) \prod_{u \in \mathcal{N}(v)} m_{uv}^{(t)}(-1)$
    Normalize to ensure $P(x_v = +1|\bar{x}_L) + P(x_v = -1|\bar{x}_L) = 1$
  return Posterior probabilities $P(x_v = +1|\bar{x}_L)$ for all $v \in V$

Source: Based on Gong, Frank, and Mittal (2014).

Table 2 – SybilBelief algorithm notation.

Notation	Description
$G = (V, E)$	Social network graph with vertices V and edges E
L_b	Set of labeled benign nodes
L_s	Set of labeled Sybil nodes
w	Homophily strength parameter ($w > 0.5$)
ε	Convergence threshold for message passing
d_{max}	Maximum number of LBP iterations
θ_v	Prior probability parameter for node v
x_v	Binary random variable for node v (+1 benign, –1 Sybil)
$\phi_v(x_v)$	Node potential for node v
$\phi_{uv}(x_u, x_v)$	Edge potential between nodes u and v
$\phi_v^L(x_v)$	Evidence potential incorporating known labels
w_{uv}	Coupling strength between connected nodes u and v
$m_{uv}^{(t)}(x_v)$	Message from node u to node v at iteration t
$\mathcal{N}(v)$	Set of neighbors of node v in the social graph
$P(x_v = +1 \mid \bar{x}_L)$	Posterior probability that node v is benign
$\bar{x}_L$	Observed labels for nodes in set L
K	Number of boosting trials

Source: Based on Gong, Frank, and Mittal (2014).

scenarios and substantially outperforming conventional schemes when attackers attempt to game the system by creating artificial common friends. This study claims that this technique ensures uniform seed distribution across communities, robustness against advanced attack strategies, and maintained computational efficiency with O(n log²n) complexity. However, the approach faces challenges including parameter sensitivity requiring heuristic tuning, dependency on accurate community detection, higher false positive rates in simple attack scenarios, and limited evaluation scope using only Facebook data with simulated attacks, which may affect its generalizability to other social networks.

2.3 MACHINE LEARNING ALGORITHMS

Machine learning classifiers for tabular data are computational algorithms designed to automatically learn patterns from structured datasets and make predictions on new, unseen instances. The fundamental task of machine learning classifiers is to map input features to output labels by discovering underlying relationships in the training data (GÉRON, 2019). For binary classification problems, such as Sybil detection, the classifier learns to distinguish between two classes (e.g., legitimate users vs. Sybil accounts) by analyzing feature vectors that represent user characteristics and behaviors.

Algorithm 3 SybilRank algorithm.

procedure SYBILRANK(G, M, T_G, C_{TH}) ▷ Sybil detection using trust propagation
  Select M honest seeds from high-degree nodes
  Initialize trust values:
  for $v \in V$ **do**
    if $v \in$ honest seeds **then**
      $T^{(0)}(v) \leftarrow \frac{T_G}{M}$ ▷ Initial trust for seeds
    else
      $T^{(0)}(v) \leftarrow 0$ ▷ No initial trust
  $w \leftarrow \lceil \log |V| \rceil$ ▷ Number of iterations
  for $i \leftarrow 1$ **to** w **do**
    for $u \in V$ **do**
      $T^{(i)}(u) \leftarrow 0$ ▷ Initialize current iteration
      for $v \in$ neighbors of u **do**
        $T^{(i)}(u) \leftarrow T^{(i)}(u) + \frac{T^{(i-1)}(v)}{\deg(v)}$ ▷ Aggregate trust from neighbors
  for $v \in V$ **do**
    if $T^{(w)}(v) < C_{TH}$ **then** ▷ Classify nodes
      Label v as Sybil
    else
      Label v as honest
  return Classification labels

Source: Based on HARUTA, TOYODA, and SASASE (2016).

Table 3 – SybilRank algorithm notation.

Notation	Description
$G = (V, E)$	Social network graph with vertices V and edges E
M	Number of honest seed nodes to select
T_G	Total trust value distributed among seeds
C_{TH}	Classification threshold for Sybil detection
$T^{(i)}(v)$	Trust value of node v after i iterations
w	Number of power iterations $\lceil \log \lvert V \rvert \rceil$
$\deg(v)$	Degree of node v (number of neighbors)
$\lvert V \rvert$	Total number of nodes in the graph

Source: Based on HARUTA, TOYODA, and SASASE (2016).

The core mathematical framework involves learning a function $f : X \rightarrow Y$, where X represents the feature space and Y the label space. For a dataset with m training instances, the goal is to minimize a cost function that measures the difference between predicted and actual labels, as shown in Equation 1:

$$\min_{\theta} J(\theta) = \min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(f_{\theta}(x^{(i)}), y^{(i)}) \tag{1}$$

where $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$ represents the training dataset, θ denotes the model parameters, L is the loss function, and f_θ is the parameterized prediction function (GÉRON, 2019).

Evaluating machine learning classifiers requires multiple metrics beyond simple accuracy, particularly for imbalanced datasets common in Sybil detection. Key evaluation metrics include precision (the ratio of true positives to all positive predictions), recall (the ratio of true positives to all actual positives), and the F1-score (the harmonic mean of precision and recall). The confusion matrix provides a comprehensive view by showing true positives, true negatives, false positives, and false negatives. For probabilistic classifiers, the ROC curve and AUC score measure the trade-off between true positive rate and false positive rate across different decision thresholds (GÉRON, 2019).

2.3.1 Supervised Learning approaches

Table 4 presents an overview of the main supervised learning algorithms commonly used for classification tasks, highlighting their key advantages and limitations (GÉRON, 2019).

Recent studies have applied various machine learning approaches to Sybil detection with different levels of success. (CÁRDENAS-HARO et al., 2024) developed the SybilSocNet algorithm using K-Nearest Neighbors (KNN), SVM, and Random Forest classifiers. Their evaluation focused on accuracy as the primary metric, achieving 97% accuracy with KNN, 99.9% accuracy with SVM, and 99.6% accuracy with Random Forest on their Twitter dataset after partitioning the data into 71 smaller matrices to manage computational demands.

(BANSAL; MISRA, 2016) employed a combination approach integrating content-based features with graph-based properties. They used SGD Classifier as the primary algorithm and evaluated performance using accuracy, precision, and recall metrics. Their method achieved approximately 14% reduction in false positive rates compared to existing approaches like SybilGuard and SybilShield, while maintaining comparable accuracy levels around 90-95% on Facebook datasets.

(JAVED et al., 2025) proposed an interpretable AI-based framework using multiple algorithms including LightGBM, XGBoost, Random Forest, SVM, and Naive Bayes. Their evaluation employed accuracy, precision, recall, F1-score, and AUC metrics across two datasets (Cresci-15 and Cresci-17). The results showed LightGBM achieving the best performance with 99.1% accuracy and 99.3% F1-score on Cresci-15, while XGBoost achieved 99.0% accuracy and 99.3% F1-score on Cresci-17, demonstrating the effectiveness of ensemble methods for bot detection tasks.

Table 4 – Comparison of supervised learning classifiers

Classifier	**Main advantages**	**Main cons**
Stochastic Gradient Descent (SGD) Classifier	Handles large datasets efficiently; suitable for online learning; fast training	Relies on randomness; requires parameter tuning; sensitive to feature scaling
Logistic Regression	Outputs class probabilities; interpretable; good baseline; works well with linear boundaries	Limited to linear relationships; struggles with complex non-linear patterns
Random Forest	Handles multiple classes; provides feature importance; robust to overfitting; no feature scaling needed	Slower on large datasets; less interpretable than single trees
Support Vector Machine (SVM)	Effective for high-dimensional data; memory efficient; versatile with kernels	No direct probability estimates; sensitive to scaling; slow on large datasets
K-Nearest Neighbors	Simple and intuitive; no data distribution assumptions; good for small datasets	Computationally expensive; sensitive to irrelevant features; degrades in high dimensions
Decision Trees	Highly interpretable; minimal data preparation; handles non-linear relationships	Prone to overfitting; unstable to data changes; can create overly complex models
Naive Bayes	Fast and simple; works with small datasets; good for text classification	Strong independence assumption; can be outperformed by sophisticated methods
Ensemble Methods	Higher accuracy through combination; reduces overfitting; robust performance	Computationally expensive; less interpretable; sequential training for boosting

Source: Based on (GÉRON, 2019).

2.3.2 Semi-supervised Learning approaches

Semi-supervised learning becomes essential when labeled data is scarce or expensive to obtain, while unlabeled data is abundant (TANHA; SOMEREN; AFSARMANESH, 2017). This scenario is particularly relevant in Sybil detection, where manually labeling large numbers of accounts as legitimate or malicious requires significant human expertise and time. Self-training, a wrapper algorithm around base classifiers, addresses this challenge by iteratively expanding the labeled dataset using high-confidence predictions on unlabeled instances.

(TANHA; SOMEREN; AFSARMANESH, 2017) applied self-training specifically to decision tree classifiers, addressing the fundamental problem that standard decision trees produce poor probability estimates for instance selection. Their research context involved scenarios where the amount of labeled data is typically quite small (10%

labeled, 90% unlabeled), which mirrors real-world conditions in social network analysis where obtaining ground truth labels is resource-intensive.

The study demonstrated that standard decision tree learning fails as a base learner in self-training algorithms due to unreliable probability estimation. However, by implementing several modifications including Laplacian correction, no-pruning (C4.4), grafting techniques, Naive Bayes Trees (NBTree), and distance-based measures, the authors achieved significant improvements.

Before applying self-training, the standard decision tree (J48) showed essentially no improvement when using unlabeled data (average improvement of 0.15%). After implementing the enhanced probability estimation methods, self-training achieved substantial gains: C4.4 showed 1.9% average improvement, grafted trees achieved 1.6% improvement, the combination C4.4graft reached 3.5% improvement, and NBTree demonstrated 2.7% average improvement across multiple UCI datasets (TANHA; SOMEREN; AFSARMANESH, 2017). These results highlight the importance of reliable confidence estimation in semi-supervised learning approaches for classification tasks.

3 PROPOSED SOLUTION

In summary, the proposed solution is a Sybil detection system that combines a graph-based algorithm with machine learning models to generate probability scores for Farcaster user accounts. Results will be publicly accessible through an open API, and intuitive interface for users and a monitoring dashboard. This will enable platform administrators, companies, and users to identify and handle Sybil accounts exploiting financial incentives in the Farcaster ecosystem, ultimately enhancing Farcaster users experience by allowing features to filter Sybil content on the client side. This chapter breaks down the main and specific goals for project development, as well as the methodology to be used to address Sybil detection in Farcaster.

3.1 PRIMARY GOALS

The primary goals of this project were established through formal agreement with Optimism and define the core requirements that the final solution must satisfy. These objectives ensure that the developed algorithm meets both technical performance standards and practical deployment requirements for integration with the Farcaster ecosystem.

The algorithm must provide probabilistic Sybil detection by returning a Sybil probability score between 0 and 1 for each user, where 0 indicates a benign account and 1 indicates a highly likely Sybil account. This probabilistic approach enables clients to implement flexible thresholds based on their specific use cases and risk tolerance levels.

Public accessibility represents a critical requirement, with algorithm results made available through a public API that enables integration with existing Farcaster clients such as Warpcast, Supercast, and other applications within the ecosystem. This open access approach aligns with Farcaster's decentralized philosophy and ensures broad community benefit.

The solution must achieve a minimum AUC score of 0.7 on the test dataset, establishing a baseline performance threshold that ensures practical utility for Sybil detection tasks. This metric provides a standardized evaluation criterion that accounts for the trade-off between true positive and false positive rates across different decision thresholds.

User accessibility requires implementation of an intuitive interface for checking the Sybil probability of any Farcaster account, with Frames recommended as the primary interaction mechanism. This approach leverages Farcaster's native interactive capabilities, allowing users to query Sybil probabilities directly within their social feed without leaving the platform environment. There is also the requirement for having a feedback mechanism that allows users to report false algorithm detections, despite

there is no need for a new calculation of sybil probabilities on those cases.

Also, operational transparency will be maintained through a comprehensive monitoring dashboard that tracks algorithm performance statistics, usage patterns, and system health metrics. This dashboard provides stakeholders with real-time insights into algorithm behavior and enables continuous performance monitoring.

3.2 SPECIFIC GOALS

Beyond the primary objectives, several specific technical goals were established to ensure optimal system performance and scalability. These goals address implementation details that are crucial for practical deployment in a high-throughput environment.

API performance optimization targets a maximum response time of 500 milliseconds per query, ensuring that Sybil probability requests can be served with minimal latency. This performance requirement enables seamless integration with real-time applications and maintains acceptable user experience standards for interactive queries.

Scalability design accommodates daily updates of Sybil probabilities for all Farcaster users, requiring an architecture capable of processing hundreds of thousands of accounts efficiently. The system must handle both batch processing for periodic updates and real-time queries for individual users without performance degradation.

3.3 METHODOLOGY

This section presents the methodology developed to address Sybil detection in Farcaster, structured around four key components: evaluation dataset construction, graph-based algorithm selection, machine learning model development, and final system deployment. The idea is to show an overview planning for what will be done in Section 4.

3.3.1 Evaluation Dataset

The construction of a labeled dataset represents the first step to allow a high-quality algorithm evaluation. a methodology was previously established in order to avoid possible biases on the testing labels, while balancing accuracy requirements with time efficiency.

Regardless of the evaluation dataset source, the proposal is that a minimum of 100 samples (50 benign and 50 Sybil) will undergo manual inspection to ensure dataset quality. It will be considered acceptable only if it achieves at least 90% accuracy during manual verification. Within each subset, manual inspection categorizes accounts as clear positives, unclear labels, or clear negatives. For acceptance, the 50 Sybil samples must contain at least 45 clear Sybil accounts, with the same standard applied to benign samples. The amount of 50 samples per class was chosen because it balances

reasonable sampling requirements with time efficiency, as manual evaluation of 100 accounts requires several hours of analysis.

The human inspection follows a protocol to ensure consistency and reduce bias. The 50 samples from each class are shuffled together before evaluation, preventing the reviewer from knowing the original labels during assessment. Before results are compared against original labels to calculate accuracy metrics, human evaluators classify accounts based on Sybil identification criteria including:

- Spamming promotional content with repetitive messaging patterns
- Frequent posts about earning yields and easy profit from tokens or financial programs;
- Posts attempting to imitate human routines using clearly LLM-generated text and images;
- Posting images with very low resolution, typically AI-generated or web-scraped images not validated by humans;
- Very low engagement rates on casts, with minimal genuine replies and likes;
- Interactions predominantly from other suspect accounts exhibiting similar characteristics;
- Behavioral pattern of imitating normal human activity followed by sudden drops of spam content;
- Frequent mentions of memecoins, betting opportunities, and free NFT distributions.

The proposal is to use a three-tier priority for obtaining labels. The primary approach evaluates existing detection systems such as Bot or Not for label accuracy. If these systems demonstrate sufficient reliability (accuracy greater than 90% on manual validation), their classifications provide the foundation for the evaluation dataset. If existing systems prove inadequate, the secondary approach implements automated filtering mechanisms targeting users with verification indicators such as ENS names, Gitcoin Passport scores above specified thresholds, and X verification status. These filters aim to identify subsets with higher concentrations of legitimate users. If automated approaches fail to produce reliable labels, the tertiary approach implements direct manual sampling and classification of randomly selected users from the Farcaster user base, applying the Sybil identification criteria consistently across all evaluations.

3.3.2 Graph-based Algorithm

The graph-based component evaluation compares three algorithms, SybilSCAR, SybilRank, and SybilBelief, selected based on their demonstrated effectiveness in academic literature and theoretical foundations. All three algorithms will undergo testing and comparison using two evaluation metrics: AUC (the primary metric) and execution time (which also represents a critical evaluation dimension given Farcaster's scale of over 800,000 users and potential future expansion).

The selected algorithm must demonstrate computational efficiency suitable for periodic batch processing of the entire user base. AUC scores provide performance comparison across different algorithmic approaches while accounting for class imbalance considerations. Only one graph-based algorithm will be selected for final implementation due to the complexity of optimizing algorithm time-efficiency for large-scale deployment.

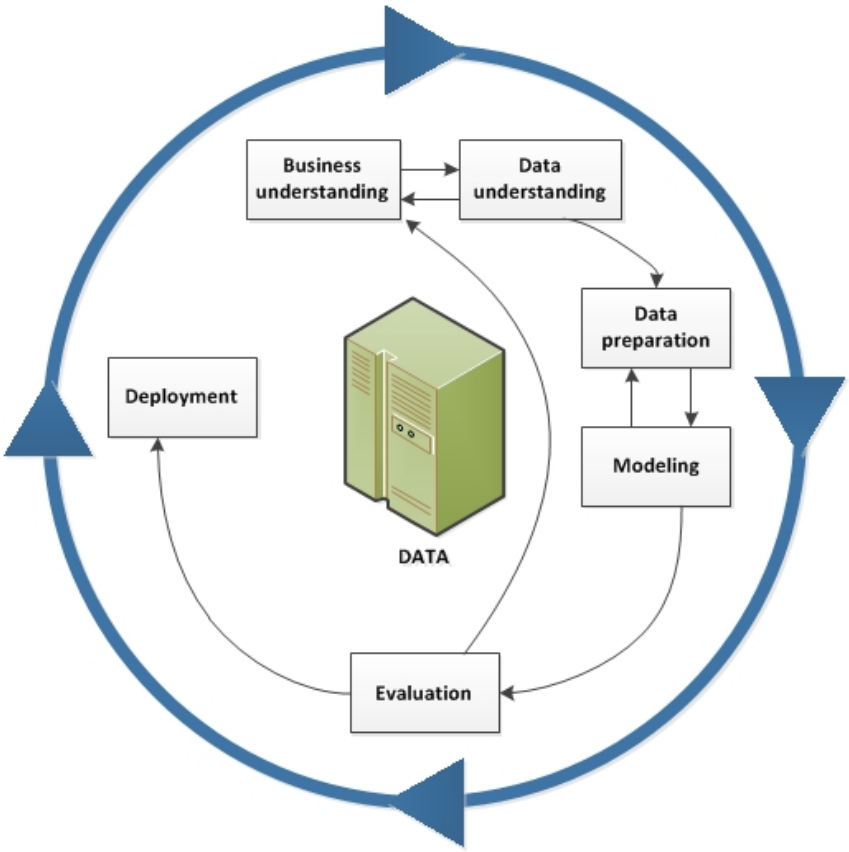
3.3.3 Machine Learning Model

The development of machine learning models follows the methodology CRISP-DM (Cross-Industry Standard Process for Data Mining), a six-phase framework from initial business understanding through deployment and monitoring (IBM, 2021). CRISP-DM encompasses business understanding to define project objectives and success criteria, data understanding to explore dataset characteristics and quality issues, data preparation involving cleaning and feature engineering, modeling through algorithm selection and parameter tuning, evaluation using appropriate metrics and validation techniques, and deployment with ongoing model maintenance. This iterative methodology accommodates the cyclical nature of data science projects, where insights from later phases often inform refinements in earlier stages (see Figure 9).

A list of the most promising machine learning algorithms was created sorted by a combination of potential effectiveness and practical deployment considerations. Advanced decision tree based models including XGBoost, LightGBM, and Random Forest represent the highest priority due to advantages aligned with project requirements. These algorithms offer low training costs, enabling frequent model updates as new data becomes available. They require minimal data preparation, handling null values and outliers naturally without extensive preprocessing. Low prediction costs support fast API response requirements, while built-in feature importance calculations enable future interpretability enhancements, such as explaining why specific users received Sybil classifications.

Classical machine learning approaches including Naive Bayes and k-Nearest Neighbors provide secondary options that may require additional data preparation and techniques for feature importance calculation. However, they maintain computational

Figure 9 – CRISP-DM methodology overview.



Source: (IBM, 2021).

efficiency for both training and prediction phases, making them viable alternatives if ensemble methods prove inadequate. Deep learning approaches serve as final options due to slower training requirements and black-box characteristics that complicate feature importance analysis. Neural networks offer potential advantages in capturing complex non-linear relationships within the data that simpler algorithms might miss.

Unlike graph-based algorithms specifically designed for Sybil detection, machine learning classifiers represent general-purpose tools applied across diverse problem domains. This eliminates the need to restrict algorithm selection to those used in existing Sybil detection literature, allowing exploration of approaches that may prove more effective for the characteristics of Farcaster data.

Beyond that, self-training represents another possible useful technique for addressing label scarcity in the dataset. This semi-supervised learning approach (TANHA; SOMEREN; AFSARMANESH, 2017) proves valuable when obtaining labeled samples is expensive or time-consuming. Implementation depends on the final size of the evaluation dataset - if either the Sybil or benign label categories contain fewer than several thousand samples, self-training may improve model performance by leveraging unlabeled data to expand the training set.

3.3.4 Final Model and Deployment

The deployment methodology integrates graph-based and machine learning approaches while establishing infrastructure for operational requirements. Both graph-based and machine learning models contribute complementary strengths to Sybil detection, motivating an ensemble approach. The initial integration strategy employs simple averaging of predictions from both components. More sophisticated ensembling techniques, such as weighted averaging, will be considered based on relative performance differences. If one algorithm demonstrates substantially superior performance, weighted combination may optimize overall accuracy.

PostgreSQL database deployment will likely be implemented as solution for storing Sybil probabilities and users feedback on the algorithm. This choice reflects the technology simplicity, widespread adoption, and the development team's existing expertise, ensuring reliable implementation and maintenance. The frequency of Sybil probability updates depends on computational requirements for processing the complete user base. If algorithm execution can be completed efficiently on modest hardware within reasonable time constraints, serverless solutions such as AWS Lambda functions will enable daily updates. For computationally intensive scenarios requiring substantial processing time, more complex scheduling and resource management will be implemented to balance update frequency with infrastructure costs.

The proposed system architecture will consist of multiple interconnected components designed for reliability and maintainability. An initial data flow representation is illustrated in Figure **??**, despite the final design will depend on many other factors, like hardware needed to update users Sybil probabilities, the update frequency, and costs considerations. The plan is that data collection component will integrate multiple sources including Neynar as the primary data provider, with potential supplementary information from GitScore and CoinGecko when needed to enhance feature completeness.

4 DEVELOPMENT AND RESULTS

This chapter presents the development and evaluation results for the Sybil detection system. It covers the creation of a labeled evaluation dataset through systematic sampling and manual verification processes. The chapter examines the implementation and performance assessment of graph-based algorithms including SybilSCAR, SybilBelief, and SybilRank. It details the development of machine learning models using ensemble methods and comprehensive feature engineering. Finally, it describes the hybrid solution deployment, including system optimization and infrastructure architecture.

4.1 EVALUATION DATASET

The creation of a reliable labeled dataset represents one of the most critical components in developing and evaluating Sybil detection algorithms, as these labels serve as the ground truth foundation for both graph-based and machine learning approaches. The quality and representativeness of this dataset directly impacts the validity of algorithm performance assessments and the reliability of comparative evaluations between different detection methods.

4.1.1 Evaluation of Bot or Not Detection Results

To establish the foundation for our labeled dataset, we first evaluated the accuracy of existing Bot or Not detection results on the Farcaster platform. A manual verification process was conducted on 100 randomly selected samples: 50 accounts marked as "Bot" and 50 accounts marked as "Human" by the Bot or Not system.

The manual inspection results revealed significant insights into the Bot or Not classification performance. Among accounts classified as "Human" by Bot or Not, manual verification showed that 75% were actually Sybil accounts, while only 25% were genuine benign users. Conversely, among accounts marked as "Bot", 90% were confirmed as Sybil accounts, with 10% being legitimate users or unknown results.

This classification pattern suggests that Bot or Not employs a conservative detection strategy, prioritizing the avoidance of false positives (incorrectly marking humans as bots) over achieving high recall. The system appears to classify an account as "Bot" only when exhibiting strong Sybil indicators, resulting in high precision for bot detection but potentially missing more subtle Sybil accounts.

4.1.2 Sybil Label Dataset Construction

Given the 90% accuracy rate of Bot or Not's "Bot" classifications in identifying actual Sybil accounts, and our requirement for a labeled dataset with approximately 90% reliability, the set of users marked as "Bot" by Bot or Not provides a suitable

foundation for Sybil labels. This approach yielded approximately 4,000 Sybil accounts with high confidence, eliminating the need for extensive manual verification of Sybil samples.

4.1.3 Benign Label Dataset Construction

With the Sybil dataset established through Bot or Not classifications, the primary challenge shifted to identifying reliable sources of high-confidence benign users. A systematic investigation of the Farcaster user base revealed significant challenges in identifying high-confidence benign users among active accounts. Through comprehensive manual inspection of randomly selected users with more than 10 casts using the Warpcast interface, only approximately 5% exhibited clear characteristics of legitimate human users. The remaining accounts were distributed as follows: approximately 50% displayed obvious Sybil characteristics, while 45% fell into an uncertain category with ambiguous behavioral patterns that made definitive classification challenging.

Three distinct filtering approaches were evaluated to identify subsets of users with higher proportions of legitimate accounts, each targeting different validation mechanisms commonly associated with authentic account ownership. However, as shown in Table 5, these targeted approaches showed limited improvement in identifying high-confidence benign accounts.

Table 5 – Results of automated filtering approaches for benign user identification.

Filter Name	**Description**	**High-trust benign %**
Ethereum Name Service (ENS) name validation	Selected accounts with validated Ethereum Name Service (ENS) names linked uniquely to that account	5%
Gitcoin score	Selected accounts with Gitcoin Passport scores above 40	15%
X verification	Identified accounts with verified connections to X (formerly Twitter)	5%

The Gitcoin Passport filter showed modest improvement, achieving 15% high-confidence benign users, while ENS validation and X verification filters showed minimal improvement over baseline identification rates.

Given the limitations of automated filtering approaches, a multi-stage manual identification strategy was developed to construct a reliable benign user dataset. This process balanced the need for representative sampling with the practical constraints of manual verification requirements.

The initial step involved creating a seed set of 40 high-confidence benign users, with equal representation from two distinct categories: 20 well-known personalities within the Farcaster ecosystem (including developers, prominent community members, and verified public figures), and 20 users obtained through the filtering approaches

described above. This seed approach ensured inclusion of both clearly legitimate high-profile accounts and representative community members.

The second phase leveraged social network principles by extracting users who were followed by members of the seed set, operating under the assumption that legitimate users would be more likely to follow other legitimate accounts. This approach generated a candidate pool where the proportion of high-confidence benign accounts increased to approximately 10-15%, representing a substantial improvement over baseline identification methods.

The final verification phase involved intensive manual inspection of candidate accounts over a one-week period, applying the established Sybil identification criteria consistently across all evaluations. Accounts that did not clearly exhibit Sybil characteristics and showed genuine human behavioral patterns were classified as benign.

Table 6 shows the composition of the final benign dataset, demonstrating how the multi-stage approach successfully identified a diverse set of legitimate users.

Table 6 – Composition of the benign user dataset.

Description	**Number of Labels**
Famous users (developers, community leaders, verified public figures)	20
Users identified through ENS name validation filter	10
Users identified through Gitcoin Passport filter	10
Users followed by the initial 40, and users followed by newly verified accounts	376
Total	**416**

4.1.4 Final Dataset Composition

The complete labeled dataset comprises 4,416 samples structured to balance representativeness with practical algorithmic evaluation requirements. The Sybil subset contains 4,000 accounts identified through Bot or Not's "Bot" classifications, providing high-confidence Sybil labels with approximately 90% reliability. The benign user subset contains 416 manually verified legitimate accounts, representing various levels of user engagement and community participation patterns. This creates approximately a 10:1 ratio that maintains sufficient representation of both classes while avoiding the challenges associated with highly imbalanced datasets.

This labeling methodology leverages existing detection systems while addressing the fundamental challenge of label scarcity through systematic manual verification for benign accounts. The approach provides sufficient high-quality labels for reliable algorithm assessment while reflecting the actual distribution challenges present in the Farcaster ecosystem.

4.2 GRAPH-BASED ALGORITHM

The evaluation of graph-based algorithms for Sybil detection in Farcaster involved implementing and assessing three algorithms: SybilSCAR, SybilBelief, and SybilRank. Given the computational complexity of optimizing graph-based algorithms, a two-phase evaluation strategy was employed to efficiently identify the most promising approach before investing in full-scale optimization: first, an initial algorithm selection based on labeled data, followed by optimization with final evaluation.

4.2.1 Initial Algorithm Selection

The first phase focused on comparative performance assessment using unoptimized implementations on a subset of the available data. This approach balanced evaluation thoroughness with computational efficiency constraints, as optimizing all three algorithms for full-dataset processing would require prohibitive development time.

The evaluation methodology used 300 accounts randomly selected from each class (4,000 Sybil accounts and 416 benign accounts) to serve as prior labels for algorithm initialization. The remaining accounts—3,700 Sybil and 116 benign—constituted the test set for performance evaluation. However, due to optimization constraints, the algorithms were evaluated only on the labeled portion of the dataset rather than the complete Farcaster network.

Performance was assessed using ROC AUC as the primary metric, consistent with established practices in Sybil detection research, along with execution time as a practical deployment consideration. Table 7 presents the evaluation results for all three unoptimized algorithms on the labeled dataset.

Table 7 – Graph-based algorithms evaluation results.

Algorithm	**ROC AUC**	**Time to run (seconds)**
SybilBelief	0.701	301.6
SybilSCAR	0.703	4.2
SybilRank	0.678	338.1

Source: Author's elaboration.

The results demonstrate that SybilSCAR achieved the highest detection accuracy with a ROC AUC of 0.703, marginally outperforming SybilBelief (0.701). More significantly, SybilSCAR exhibited a substantial computational efficiency advantage, completing evaluation in 4.2 seconds compared to over 300 seconds for both SybilBelief and SybilRank. This represents approximately 70x faster execution, making SybilSCAR the optimal candidate for optimization and full-scale deployment.

4.2.2 Algorithm optimization and evaluation on full dataset

Based on the initial evaluation results, SybilSCAR was selected for comprehensive optimization to enable processing of the complete Farcaster network. The optimization process focused on multithread processing and enhanced data structure implementations to address the computational bottlenecks identified during initial testing.

The optimization strategy involved parallelizing independent computational components across multiple threads while implementing more efficient data processing routines. These improvements reduced processing time from several hours to approximately 30 minutes for the full network analysis, making practical deployment feasible.

Following optimization, SybilSCAR was evaluated on the complete Farcaster dataset, leveraging all available users and their interconnections. This full-network analysis achieved a substantially improved ROC AUC of 0.95, demonstrating that graph-based algorithms benefit significantly from access to the complete social network structure rather than isolated subsets.

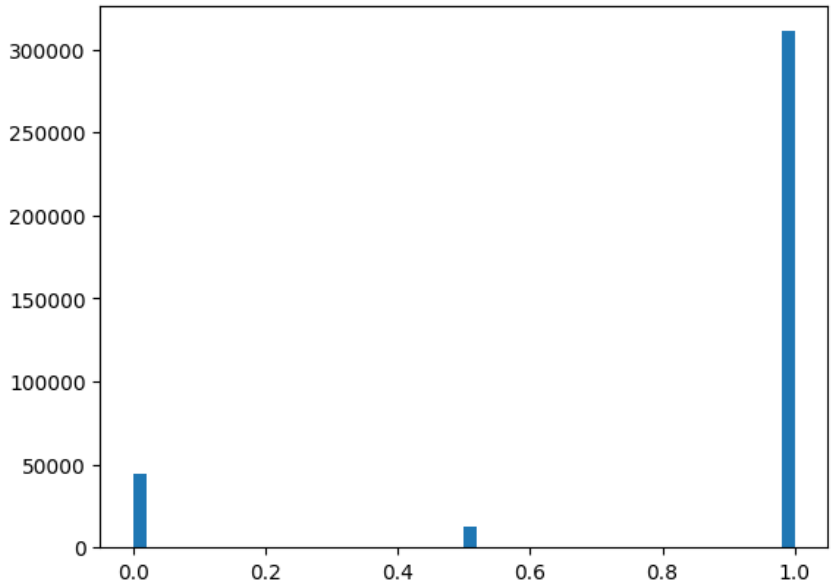
During implementation, an important constraint was identified: graph-based algorithms cannot compute individual node outputs in isolation but require processing the entire social graph to generate probability scores for all nodes simultaneously. This characteristic stems from the iterative trust propagation mechanisms employed by these algorithms. Consequently, deployment requires a batch processing model where algorithms run periodically to compute Sybil probabilities for all users, with results stored and served through the API between batch runs.

4.2.3 Analysis of SybilSCAR results on unlabeled data

Figure 10 shows the probability distribution generated by SybilSCAR on the complete unlabeled Farcaster user base. The analysis reveals three critical findings. First, SybilSCAR generated predictions for only around 300,000 users out of approximately 800,000 registered accounts. The remaining users lack mutual connections, placing them outside the social graph structure required for analysis. These isolated accounts receive null predictions that will need to be handled in the final hybrid algorithm.

Second, the algorithm identifies significantly more Sybil accounts than benign users, consistent with the earlier finding that over 95% of active Farcaster users exhibit Sybil characteristics. This validates the representative nature of the labeled dataset. Finally, SybilSCAR exhibits high prediction confidence, with scores concentrated at extremes (near 0 and 1) and few uncertain predictions. While this demonstrates algorithmic certainty, human inspection reveals many accounts with ambiguous characteristics that challenge definitive classification. This observation reinforced the development of a hybrid approach to provide more nuanced uncertainty quantification.

Figure 10 – SybilSCAR unlabeled data Sybil probability distribution.



Source: Author's elaboration.

4.3 MACHINE LEARNING MODEL

The machine learning component of the hybrid Sybil detection system followed the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, which provides a structured framework encompassing data understanding, data preparation, modeling, and evaluation phases. This systematic approach ensured comprehensive exploration of available data sources and methodical development of predictive features suitable for Sybil detection in the Farcaster environment.

4.3.1 Data Understanding

The data understanding phase involved comprehensive mapping of all data sources available through the Neynar API, which serves as the primary data provider for Farcaster user information. The complete dataset was downloaded and systematically inspected to assess data quality, identify potential issues, and determine viable feature extraction opportunities. Table 8 presents the comprehensive mapping of available data sources and their characteristics.

Table 8 – Neynar Farcaster tables descriptions.

Table	**Content description**
farcaster_casts	Contains all posts (casts) with temporal information (timestamp), user identification (fid), content analysis data (text, embeds, mentions), and social interaction patterns (parent_hash for replies, root_parent_hash for thread structure).
farcaster_fids	Provides user identification mapping (fid) and account creation timing (custody_address).
farcaster_fnames	Maps usernames to user IDs with temporal data (timestamp, expires_at) and ownership verification (custody_address).
farcaster_links	Records follow/unfollow relationships (fid, target_fid) with temporal information (timestamp) and relationship types.
farcaster_profile_addresses	Contains user profile information (display_name, bio, avatar_url) and verified wallet connections (verified_addresses).
farcaster_reactions	Tracks user interactions (likes, recasts) with temporal data (timestamp), reaction types (reaction_type), and target identification (target_hash, target_fid).
farcaster_signers	Records cryptographic signers (signer, hash) associated with user accounts (fid) and their creation timing (timestamp).
farcaster_storage	Provides storage allocation data (units) and expiration timing (expiry) per user (fid).
farcaster_user_data	Contains structured user information with data types (type: avatar, display name, bio, URL, preferred username) and temporal tracking (timestamp).
farcaster_verifications	Records Ethereum wallet address connections (claim) to Farcaster profiles (fid) with verification timing (timestamp).
warpcast_power_users	Identifies users with power badges, representing established community members.

Source: (DUNE, 2025)

The analysis revealed that available data can be categorized into three primary domains that provide complementary insights into user behavior patterns. Text analy-

sis metrics include entropy measurements and lexical diversity measures that capture content sophistication and originality patterns. Behavioral activity patterns encompass posting frequency analysis, temporal activity distributions, and engagement consistency indicators that reveal automated versus human-like interaction patterns. Network structure indicators comprise follower ratios, mutual connection statistics, and engagement distribution metrics that illuminate social relationship authenticity.

Data quality assessment revealed excellent overall data integrity, with no missing or null values detected across critical fields and no anomalous values that would require extensive preprocessing. This high data quality simplified subsequent processing steps and ensured reliable feature extraction. Additionally, since the initial modeling approach focused on ensemble tree-based algorithms including XGBoost, LightGBM, and Random Forest, there was no immediate requirement for data normalization or one-hot encoding procedures, as these algorithms naturally handle mixed data types and varying feature scales.

4.3.2 Data Preparation

The data preparation phase addressed the primary challenge of converting raw API responses into a structured feature matrix suitable for machine learning algorithms. This process involved systematic feature engineering to transform user data into quantitative indicators that capture the behavioral differences between legitimate users and Sybil accounts.

The feature engineering process generated comprehensive user profiles across three complementary categories. Tables 9, 10, and 11 provide detailed descriptions of all engineered features, demonstrating how raw social media data was systematically converted into meaningful predictive indicators.

Table 9 – User Identity Features for Sybil Detection.

Feature Identifier	**Feature Description**
has_ens	Binary indicator of whether the user has an ENS domain name.
has_bio	Binary indicator of whether the user has filled out their bio section.
has_avatar	Binary indicator of whether the user has set an avatar image.
has_display_name	Binary indicator of whether the user has set a display name.

Continued on next page

Table 9 – continued from previous page

Feature Identifier	**Feature Description**
profile_completeness	Normalized score (0-1) representing the completeness of the user's profile based on bio, avatar, ENS, and display name.
display_name_length	Character length of the user's display name.
bio_length	Character length of the user's bio description.
fname_random_numbers	Binary indicator of suspicious patterns in username, specifically sequences of 4 or more random numbers.
fname_wallet_pattern	Binary indicator of wallet address patterns (0x followed by 40 hexadecimal characters) in username.
fname_excessive_symbols	Binary indicator of excessive use of symbols (underscores, dots, dashes) in username.
fname_airdrop_terms	Binary indicator of airdrop-related terms (airdrop, farm, degen, wojak) in username.
fname_has_year	Binary indicator of year patterns (19xx or 20xx) in username.
bio_random_numbers	Binary indicator of suspicious patterns in bio, specifically sequences of 4 or more random numbers.
bio_wallet_pattern	Binary indicator of wallet address patterns in bio description.
bio_excessive_symbols	Binary indicator of excessive use of symbols in bio description.
bio_airdrop_terms	Binary indicator of airdrop-related terms in bio description.
bio_has_year	Binary indicator of year patterns in bio description.
fname_entropy	Character diversity measure in username based on unique character ratio.
bio_entropy	Character diversity measure in bio based on unique character ratio.
total_verifications	Total number of verification claims made by the user.

Continued on next page

Table 9 – continued from previous page

Feature Identifier	Feature Description
eth_verifications	Number of Ethereum signature-based verifications.
verification_consistency	Standardized measure of timing consistency between verifications.
platforms_verified	Number of unique platforms where the user has verified accounts.
verification_span_days	Time span in days between first and last verification.
avg_hours_between_verifications	Average time interval in hours between consecutive verifications.
platform_diversity	Number of unique platforms verified (same as platforms_verified).
verification_velocity	Total number of platform verifications performed.
sequential_verifications	Number of verifications performed within 1 hour of each other.
verification_gaps	Number of verification gaps longer than 24 hours.
storage_units	Average number of storage units allocated to the user.
storage_utilization	Maximum storage units ever allocated to the user.
storage_update_frequency	Number of times storage allocation was updated.
storage_growth_rate	Rate of storage growth per update normalized by update frequency.
storage_stability	Standard deviation of storage unit allocations over time.
storage_efficiency	Ratio of average storage usage to maximum storage capacity.
identity_strength	Composite score combining profile completeness, verification consistency, and platform diversity.

Continued on next page

Table 9 – continued from previous page

Feature Identifier	**Feature Description**
verification_quality	Composite score based on platform diversity, verification consistency, and Ethereum verification ratio.
profile_authenticity	Authenticity score derived from absence of suspicious patterns in username and bio.
resource_utilization	Efficiency metric combining storage efficiency, stability, and controlled growth rate.

Table 10 – Network Analysis Features for Sybil Detection.

Feature Identifier	**Feature Description**
follow_ratio	Ratio of total follows to unique follows, indicating connection redundancy patterns.
network_growth_rate	Combined growth rate based on standard deviations of follower and following count changes over time.
follow_velocity	Rate of new follow connections per hour, calculated as total follows divided by average time between follows.
network_age_hours	Total time span in hours between the first and last follow activity of the user.
growth_stability	Standard deviation of time intervals between consecutive follow actions, measuring consistency of growth.
follower_growth_rate	Relative growth rate of follower count from first to last recorded measurement.
following_growth_rate	Relative growth rate of following count from first to last recorded measurement.
follow_reciprocity	Proportion of follows that are reciprocated (mutual following relationships).
network_density	Ratio of unique connections to total connections, indicating diversity of network composition.
network_reach	Number of unique users that the account follows, representing network breadth.
network_churn_rate	Proportion of connections that have been deleted, indicating relationship instability.
network_volatility	Standard deviation of time intervals between follow actions, measuring temporal consistency.

Continued on next page

Table 10 – continued from previous page

Feature Identifier	**Feature Description**
stable_connections	Number of follow relationships that have lasted more than 30 days without being deleted.
network_centrality	Combined metric of total connections and unique connections, indicating network importance.
bridge_score	Number of unique users followed, representing potential for connecting different network clusters.
community_embedding	Ratio of unique follows to total follows, indicating integration within network communities.
network_diversity	Number of unique users in the follow network, measuring connection variety.

Table 11 – Temporal Behavior Features for Sybil Detection.

Feature Identifier	**Feature Description**
hour_diversity	Number of unique hours during which the user has been active, indicating temporal distribution.
weekday_diversity	Number of unique weekdays during which the user has been active, measuring weekly activity spread.
inactive_periods	Number of gaps longer than 24 hours between consecutive activities, indicating periods of inactivity.
daily_active_hours	Number of unique hours per day during which the user shows activity patterns.
posting_frequency	Total number of posts/casts made by the user, representing overall activity volume.
response_latency	Average time interval in hours between consecutive activities, measuring response timing patterns.
interaction_timing	Standard deviation of time intervals between activities, indicating timing consistency.
engagement_windows	Number of activities that occur within 1-hour windows, showing concentrated engagement periods.
burst_frequency	Number of activity bursts, defined as actions occurring within 5-minute intervals.
burst_intensity	Ratio of burst activities to total burst periods, measuring intensity of concentrated activity.
burst_duration	Average duration in minutes of activity bursts when they occur.

Continued on next page

Table 11 – continued from previous page

Feature Identifier	**Feature Description**
inter_burst_interval	Average time in minutes between separate burst periods, measuring spacing of intense activity.
burst_engagement_ratio	Proportion of total activities that occur during burst periods versus regular activity.
burst_impact	Total number of burst activities, representing the cumulative effect of concentrated behavior.
temporal_consistency	Standard deviation of time intervals between activities, measuring regularity of activity timing.
rhythm_score	Number of unique hours during which the user is active, indicating temporal rhythm diversity.
variability_index	Difference between 90th and 10th percentiles of inter-activity intervals, measuring timing variability.
prime_time_ratio	Proportion of activities occurring during prime hours (9 AM to 10 PM), indicating normal usage patterns.
off_hours_activity	Proportion of activities occurring outside prime hours (before 9 AM or after 10 PM).
weekend_activity_ratio	Proportion of activities occurring on weekends (Saturday and Sunday).
global_reach	Ratio of unique active hours to total possible hours (24), measuring temporal coverage.
activity_entropy	Shannon entropy of hourly activity distribution, measuring randomness of temporal patterns.
temporal_clustering	Proportion of activities occurring within 1-hour clusters, indicating tendency for grouped activity.
trend_stability	Rolling standard deviation of inter-activity intervals, measuring stability of activity trends over time.
adaptation_rate	Standard deviation of changes in rolling mean inter-activity intervals, measuring behavioral adaptation.
avg_follow_latency_seconds	Average time in seconds between consecutive follow actions, measuring follow behavior timing.
cross_channel_activity	Ratio of unique channels to total activities, indicating diversity of platform engagement.
multi_channel_ratio	Proportion of activities that involve multiple channels or parent URLs, measuring cross-platform behavior.

User identity features capture content characteristics including text complexity measures, posting patterns, and account verification indicators. Network analysis

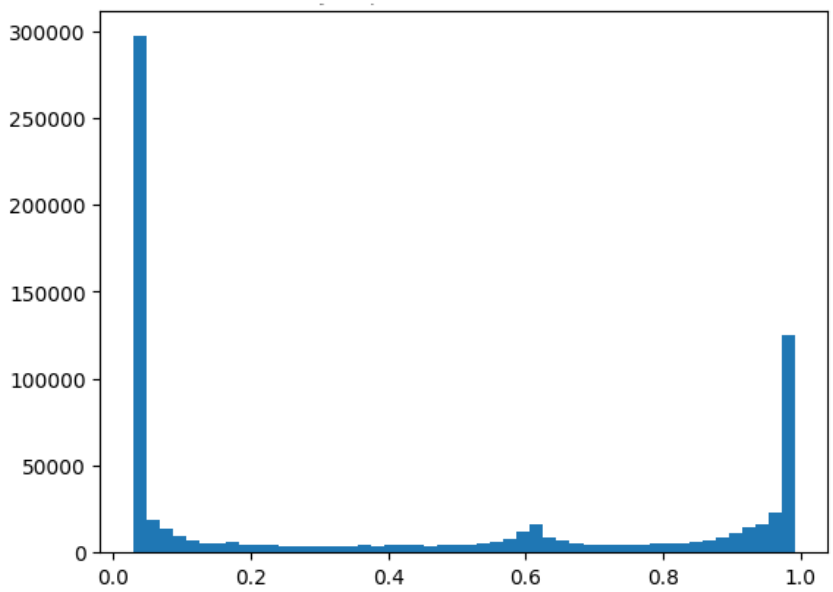
features quantify social relationship patterns such as follower-to-following ratios, mutual connection densities, and engagement distribution statistics. Temporal behavior features analyze posting frequency patterns, activity consistency, and time-based interaction behaviors that distinguish human users from automated accounts.

4.3.3 Modeling and Evaluation

The modeling phase began with systematic evaluation of three ensemble tree-based algorithms selected for their proven effectiveness in handling complex feature interactions and their robustness to the mixed data types present in social media analysis. XGBoost, LightGBM, and Random Forest were chosen as complementary approaches that leverage different ensemble strategies while maintaining interpretability through feature importance analysis.

Hyperparameter optimization was conducted through grid search procedures, with parameter ranges specifically tailored to each algorithm's characteristics. Tables 12, 13, and 14 detail the hyperparameter ranges explored for XGBoost, LightGBM, and Random Forest respectively, ensuring thorough exploration of the parameter space while maintaining computational feasibility.

Figure 11 – Machine learning model unlabeled data Sybil probability distribution.



Source: Author's elaboration.

The evaluation methodology employed a consistent train-validation-test split strategy, with the test set identical to that used in graph-based algorithm evaluation to enable direct performance comparison. This approach ensures that machine learning results are directly comparable with graph-based outcomes and provides reliable performance estimates for the final hybrid system.

Table 12 – XGBoost hyperparameter ranges.

Parameter	**Description**	**Min Value**	**Max Value**
max_depth	Maximum depth of a tree	3	7
learning_rate	Boosting learning rate (shrinkage)	0.01	0.1
n_estimators	Number of boosting rounds	100	500
min_child_weight	Minimum sum of instance weight needed in a child	1	7
subsample	Subsample ratio of the training instances	0.6	1.0
colsample_bytree	Subsample ratio of columns when constructing each tree	0.6	1.0

Source: Author's elaboration.

Table 13 – LightGBM hyperparameter ranges.

Parameter	**Description**	**Min Value**	**Max Value**
n_estimators	Number of boosting iterations	100	500
max_depth	Maximum tree depth for base learners	3	7
learning_rate	Boosting learning rate	0.01	0.1
num_leaves	Maximum number of leaves in one tree	20	100
feature_fraction	Fraction of features to be used for each iteration	0.6	1.0
bagging_fraction	Fraction of data to be used for each iteration	0.6	1.0

Source: Author's elaboration.

Table 14 – Random Forest hyperparameter ranges.

Parameter	**Description**	**Min Value**	**Max Value**
n_estimators	Number of trees in the forest	100	500
max_depth	Maximum depth of the tree	3	7
min_samples_split	Minimum number of samples required to split an internal node	2	10
min_samples_leaf	Minimum number of samples required to be at a leaf node	1	5

Source: Author's elaboration.

Individual model performance demonstrated strong Sybil detection capabilities across all three algorithms. Table 15 presents the final ROC AUC scores achieved

by the best configuration of each individual model following grid search optimization. More significantly, an ensemble approach combining predictions from all three models achieved superior performance with a ROC AUC of 0.98, substantially outperforming any individual algorithm and demonstrating the value of ensemble techniques for this classification task.

Table 15 – Final ROC AUC scores for individual ML models

Model	**AUC**
XGBoost	0.9575
LightGBM	0.9273
Random Forest	0.9424
Ensemble	0.9899

Source: Author's elaboration.

The ensemble model's superiority reflects the complementary strengths of the three constituent algorithms. XGBoost's gradient boosting approach effectively captures complex feature interactions, LightGBM's efficient implementation enables rapid processing of large feature sets, and Random Forest's bagging strategy provides robustness against overfitting while offering natural feature importance ranking.

4.3.4 Model Validation on Unlabeled Data

Beyond traditional metrics calculated on labeled test data, model validation included analysis of prediction distributions on the complete unlabeled Farcaster user base. This approach provides insights into model behavior across the full spectrum of user accounts and validates whether the learned patterns generalize effectively to the broader population.

The ideal Sybil probability distribution for a perfect classifier would exhibit a bimodal structure with distinct peaks at probability values near 0 (representing confidently identified benign users) and near 1 (representing confidently identified Sybil accounts). A well-performing classifier should approximate this ideal while showing some distribution of uncertain cases in the intermediate probability range, representing accounts with ambiguous characteristics that legitimately challenge classification.

Figure 11 displays the actual probability distribution generated by the ensemble model when applied to the unlabeled Farcaster user base. The observed distribution closely approximates the theoretical ideal, with clear concentration of predictions at the extremes and a reasonable distribution of uncertain cases in the intermediate range, providing strong evidence of model validity and effective generalization to unseen data.

4.3.5 Semi-Supervised Learning Investigation

Given the inherent challenge of obtaining large quantities of accurately labeled social media accounts, semi-supervised learning techniques were explored to potentially leverage the abundant unlabeled data available in the Farcaster ecosystem. Self-training approaches were systematically evaluated across all three selected base models, comparing performance metrics before and after semi-supervised implementation.

The self-training methodology involved iteratively expanding the labeled dataset by adding high-confidence predictions from unlabeled instances, then retraining models on the expanded dataset. This process was repeated for multiple iterations to assess potential cumulative improvements in model performance. Results from semi-supervised experiments indicated that while these methods provided marginal performance improvements, the computational overhead associated with training dozens of iterations did not justify the modest gains achieved. The iterative training process significantly increased computation time and complexity while yielding only minimal improvements in ROC AUC scores, leading to the conclusion that the additional labeled data obtained through manual verification provided sufficient training signal for effective model development.

Table 16 – Final ROC AUC scores for individual ML models after applying self-training

Model	**AUC**
XGBoost	0.9404
LightGBM	0.9564
Random Forest	0.9457

Source: Author's elaboration.

This finding suggests that the carefully curated labeled dataset, despite its relatively modest size, contains sufficient information to train effective Sybil detection models when combined with appropriate feature engineering and ensemble techniques. The marginal benefits of semi-supervised approaches likely reflect the high quality of the existing labeled data and the effectiveness of the engineered features in capturing relevant behavioral patterns.

4.4 SYSTEM IMPLEMENTATION AND DEPLOYMENT

This section describes the implementation and deployment of the hybrid Sybil detection system, including algorithm integration, infrastructure setup, and user interfaces.

4.4.1 Hybrid Algorithm Implementation

The hybrid system combines SybilSCAR graph-based predictions with machine learning ensemble predictions using simple averaging. Both algorithms can fail to produce predictions in specific cases: SybilSCAR when users lack mutual connections, and the ML model when feature extraction fails due to invalid or missing data.

The integration uses a fallback mechanism to maximize user coverage. Algorithm 4 shows the prediction logic.

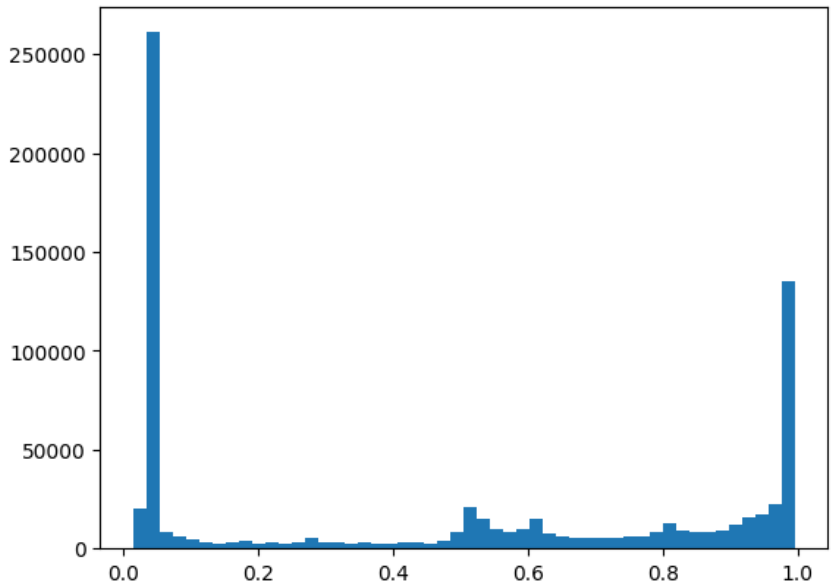
Algorithm 4 Hybrid model prediction logic

procedure HYBRIDPREDICT(*sybilscar_result*, *ml_model_result*)
  if *sybilscar_result* is None **then**
    $mean \leftarrow ml\_model\_result$
  else if *ml_model_result* is None **then**
    $mean \leftarrow sybilscar\_result$
  else
    $mean \leftarrow \frac{sybilscar\_result + ml\_model\_result}{2}$
  return *mean*

Source: Author's elaboration.

Performance results on the test set show: hybrid model ROC AUC = 0.99, SybilSCAR AUC = 0.95, ML ensemble AUC = 0.98. The hybrid approach outperformed individual components.

Figure 12 – Final model unlabeled data Sybil probability distribution.



Source: Author's elaboration.

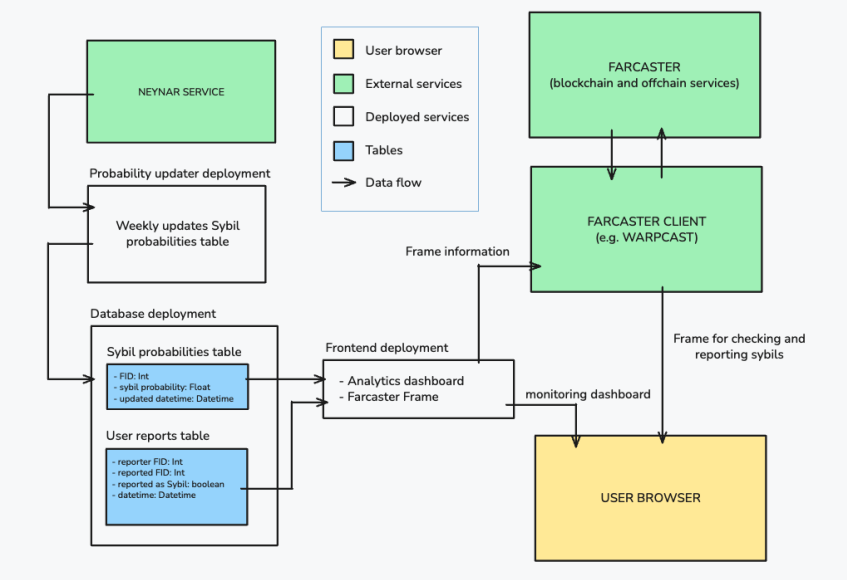
Figure 12 shows the Sybil probability distribution for 750,000+ Farcaster users. Approximately 30% of users received confident Sybil classifications (probability > 0.9).

The distribution shows bimodal characteristics with clear separation between benign and Sybil populations.

4.4.2 Production System Architecture

The experimental implementations were refactored into production Python scripts with error handling and logging. The complete pipeline (data download, SybilSCAR analysis, ML training, result merging) requires 5 hours on high-performance infrastructure. To reduce costs, the system uses weekly updates and automated resource management. A lightweight control machine activates high-performance instances weekly, runs the pipeline, stores results, and deactivates resources. This reduces computational costs by 85% while maintaining functionality.

Figure 13 – Final deployment overview.



Source: Author's elaboration.

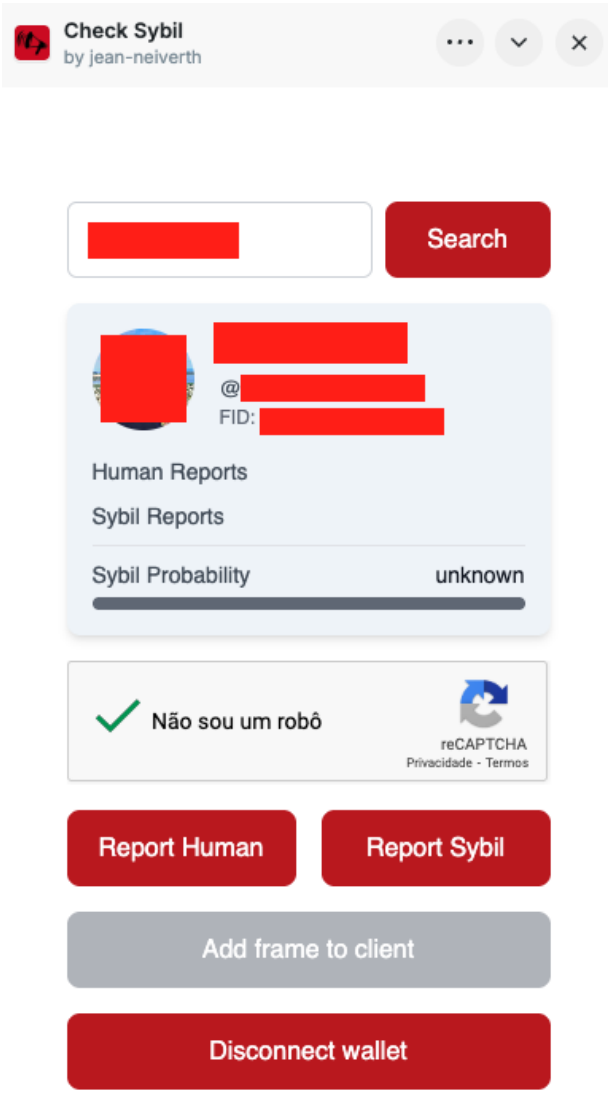
The architecture consists of a PostgreSQL database for storing Sybil probabilities and user feedback, and an orchestration service for resource management and scheduling. Because the final solution stores the weekly probabilities on a table, the API responses tend to be very fast, once they don't pass through any computational expensive algorithm, only data fetching. This way, final API response time doesn't take much than 300ms, achieving one of the project specific goals. Figure 13 shows the system architecture and data flow.

4.4.3 Farcaster Frame Interface

The Farcaster Frame allows users to query Sybil probabilities within their social feed. The implementation follows Farcaster specifications and works with clients including Warpcast and Supercast. The Frame workflow includes a username search

interface, result display with user information and Sybil probability, and optional feedback submission. CAPTCHA verification prevents feedback manipulation by malicious actors.

Figure 14 – Farcaster Frame interface.



Source: Author's elaboration.

Figure 14 shows the Frame interface from search to result display and feedback submission.

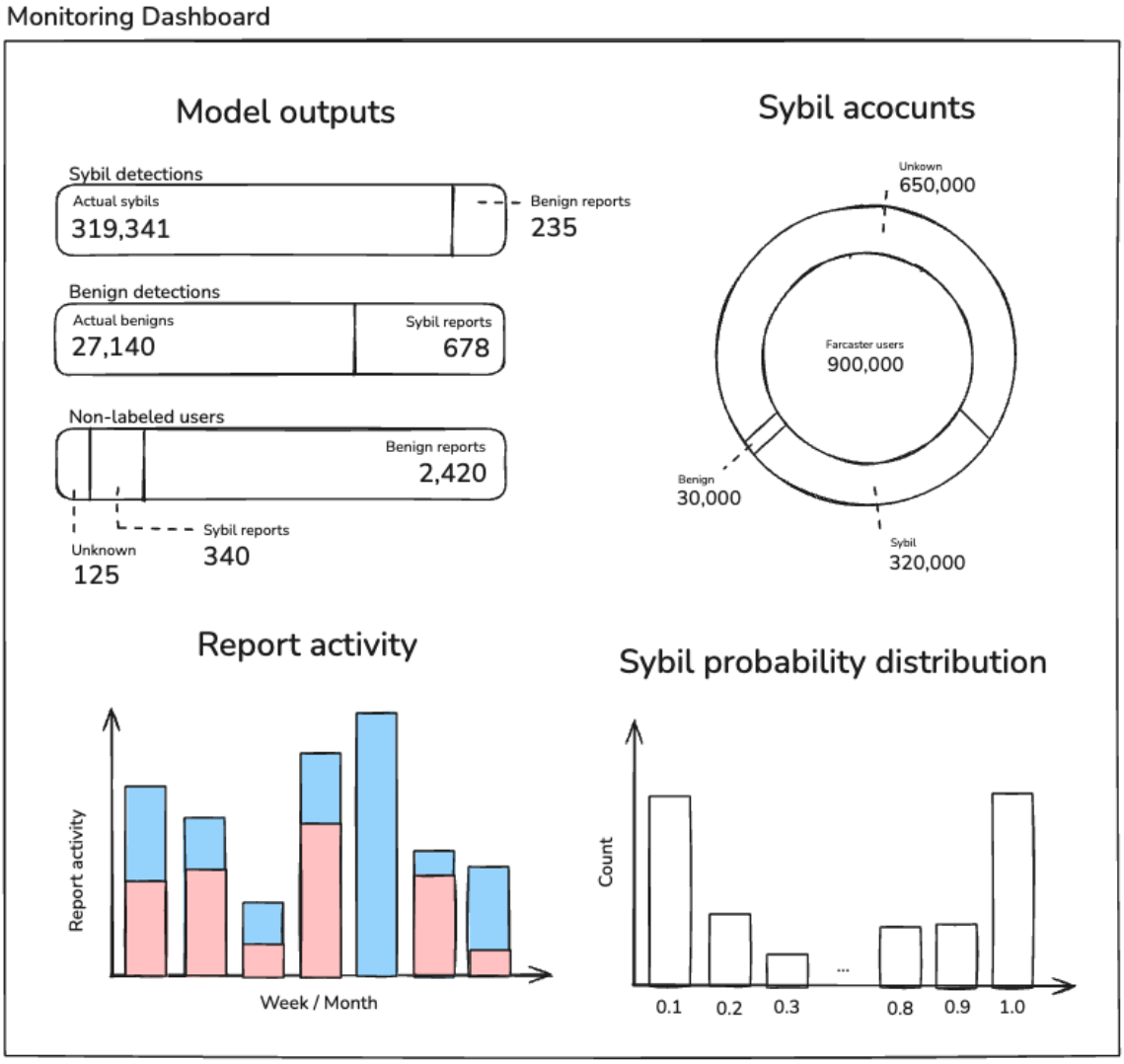
4.4.4 Monitoring Dashboard

The monitoring dashboard tracks algorithm performance and system metrics through statistical analysis and visualization. Key metrics include distribution of Sybil and benign classifications, comparison of predictions with user feedback, allowing estimation of true and false positive rates, considering a base threshold of 0.5. Beyond

that, the amount of unknown user classifications, feedback patterns, frame utilization and weekly activity reports allow quantifying the usage on the users side.

Finally, the dashboard provides sybil probability distributions, which is important to keep track of general ecosystem health and possibly signal changes in the algorithm. Figure 15 shows the dashboard organization. The system integrates with the database to provide real-time updates and historical analysis.

Figure 15 – Monitoring dashboard interface mockup.



Source: Author's elaboration.

5 CONCLUSIONS

This chapter synthesizes the outcomes of the Sybil detection system development for the Farcaster blockchain-based social network. The following sections present the project achievements including both technical accomplishments and practical impact, identify key challenges and future research directions, and provide final remarks on the contribution to decentralized social network security.

5.1 PROJECT ACHIEVEMENTS

The project successfully met all primary objectives established with Optimism, delivering a functional hybrid Sybil detection system for the Farcaster ecosystem. The system achieved a ROC AUC score of 0.99, significantly exceeding the minimum 0.7 threshold requirement. API response times remained consistently below the 500-millisecond target, enabling real-time integration with existing Farcaster clients including Warpcast and Supercast.

Technical deliverables include a comprehensive API deployment providing probabilistic Sybil scores (0-1 range), an intuitive Farcaster Frame interface for user interaction, and a monitoring dashboard for real-time performance tracking. The hybrid algorithm successfully combines SybilSCAR graph-based detection with machine learning ensemble methods (LightGBM, Random Forest, XGBoost), validating the superior performance of mixed approaches over single-method solutions.

Public accessibility through the API extends the system's impact across the entire ecosystem, addressing critical platform challenges by enabling enhanced content filtering, reducing Sybil-generated spam visibility, and improving overall user experience. Organizations within the Optimism ecosystem can now implement targeted screening for reward programs, preventing exploitation through coordinated bot networks. Individual users benefit from improved platform quality and access to Sybil probability scores for informed interaction decisions. Meanwhile, the monitoring dashboard provides community oversight of performance metrics.

One objective was only partially met: the system achieves weekly rather than daily batch processing updates due to computational constraints. However, this frequency still provides practical utility for Sybil detection purposes within the current Farcaster user base.

5.2 CHALLENGES AND FUTURE WORK

The static labeling system presents the most significant limitation, as training labels remain fixed while Sybil attack strategies continue evolving. Future work must develop dynamic label updating mechanisms to maintain detection effectiveness over

time. This includes both automated updating approaches and periodic manual review systems. Beyond that, a possible improvement would be to add more benign samples to the labels dataset, increasing the confidence in the test metrics.

Scalability represents a concern for future deployment, especially if Farcaster experiences massive user growth without substantial infrastructure investment or algorithmic optimization. Research into more efficient data processing optimizations and distributed processing approaches could be essential for maintaining system performance at scale.

The open-source implementation creates vulnerability to adversarial adaptation. Attackers can examine detection features and algorithms to develop evasion strategies. Ongoing algorithm evolution and development of inherently difficult-to-replicate features are necessary to address this challenge.

Community feedback mechanisms require robust verification systems to prevent manipulation. Coordinated false feedback from Sybil attackers could potentially compromise system calibration, necessitating careful design of verification protocols.

5.3 FINAL REMARKS

This work demonstrates the feasibility and effectiveness of hybrid Sybil detection approaches in blockchain-based social networks. The successful Farcaster ecosystem deployment validates the potential for academic research to address practical challenges in decentralized platform security.

The hybrid detection framework, combining graph-based and machine learning methods, provides a robust foundation adaptable to other social network platforms. The open-source implementation enables community-driven improvements while contributing to broader goals of secure and trustworthy decentralized social environments.

The project establishes a foundation for ongoing research in decentralized social network security, with publicly available algorithms and performance metrics supporting continued development in this critical area of emerging technology security.

REFERENCES

BANSAL, H.; MISRA, M. Sybil Detection in Online Social Networks (OSNs). **IEEE 6th International Conference on Advanced Computing (IACC)**, 2016.

CÁRDENAS-HARO, José Antonio; SALEM, Mohamed; ALDACO-GASTÉLUM, Abraham N.; LÓPEZ-AVITIA, Roberto; DAWSON, Maurice. Enhancing Security in Social Networks through Machine Learning: Detecting and Mitigating Sybil Attacks with SybilSocNet. **Algorithms**, 2024.

DUNE. **Farcaster data from Neynar**. 2025. Available from: https://docs.dune.com/data-catalog/community/farcaster/overview.

ENS. **What is the Ethereum Name Service?** 2025. Available from: https://docs.ens.domains/learn/protocol/.

FARCASTER. **Architecture documentation**. 2025. Available from: https://docs.farcaster.xyz/learn/architecture/overview.

GÉRON, Aurélien. **Hands-on Machine Learning with Scikit-Learn, Keras TensorFlow**. [S.l.]: O'Reilly, 2019.

GONG, N. Z.; FRANK, M.; MITTAL, P. SybilBelief: A Semi-Supervised Learning Approach for Structure-Based Sybil Detection. **IEEE Transactions on Information Forensics and Security**, 2014.

HARUTA, S.; TOYODA, K.; SASASE, I. Trust-Based Sybil Nodes Detection with Robust Seed Selection and Graph Pruning on SNS. **IEICE Transactions on Communications**, 2016.

IBM. **CRISP-DM Help Overview**. 2021. Available from: https://www.ibm.com/docs/pt-br/spss-modeler/saas?topic=dm-crisp-help-overview.

JAVED, D.; JHANJHI, N. Z.; KHAN, N. A.; RAY, S. K.; AL-DHAQM, A.; KEBANDE, V. R. Identification of Spambots and Fake Followers on Social Network via Interpretable AI-Based Machine Learning. **IEEE Access**, 2025.

LAND, Collab. **Gitcoin passport**. 2025. Available from: https://docs.collab.land/help-docs/command-center/create-a-tgr/gtc-passport/.

NEYNAR. **Neynar user score**. 2025. Available from: https://docs.neynar.com/docs/neynar-user-quality-score.

AL-QURISHI, M.; AL-RAKHAMI, M.; ALAMRI, A.; ALRUBAIAN, M.; RAHMAN, S. M. M.; HOSSAIN, S., M. Sybil Defense Techniques in Online Social Networks: A Survey. **IEEE Access**, 2017.

TANHA, J.; SOMEREN, M. van; AFSARMANESH, H. Semi-supervised self-training for decision tree classifiers. **Int. J. Mach. Learn. Cyber**, 2017.

WANG, B.; ZHANG, L.; GONG, N. Z. SybilSCAR: Sybil detection in online social networks via local rule based propagation. **IEEE INFOCOM**, 2017.

WU, Junchao; YANG, Shu; ZHAN, Runzhe; YUAN, Yulin; CHAO, Lidia Sam; WONG, Derek Fai. A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions. **Computational Linguistics**, 2025.

ZHANG, K.; LIANG, X.; LU, R.; SHEN, X. Sybil Attacks and Their Defenses in the Internet of Things. **IEEE Internet of Things Journal**, 2014.