Securing the Gaming Ecosystem: Anomaly Detection in Steam User Data

Deepanshu Jain djain63@asu.edu
Rishit Puri rpuri10@asu.edu
Shreya Sett ssett@asu.edu
Mounusha Metti mmetti@asu.edu

Abstract

This project aims to identify suspicious user behavior in Steam gaming profiles using anomaly detection techniques, focusing on features like account activity, playtime, and ban history. The goal is to develop a robust system for flagging potentially fraudulent accounts, contributing to a fairer gaming environment. By comparing statistical and machine learning-based methods, the project will offer insights into effective anomaly detection techniques and improve cheat detection on gaming platforms.

1 Introduction

The rapid expansion of online gaming has brought millions of players together on platforms like Steam, which serves as a central hub for game distribution, social interaction, and multiplayer engagement. However, this growth has also introduced significant challenges—most notably, the rise of cheaters, bots, and malicious users who exploit system vulnerabilities, disrupt fair gameplay, and erode trust within the gaming community.

This project addresses these challenges by developing an anomaly detection framework to identify suspicious user behavior on the Steam platform. Using a comprehensive dataset that includes account-level features such as playtime, game history, ban records, and profile metadata, we apply a combination of rule-based heuristics, statistical methods, clustering techniques, and machine learning models to detect outliers.

The primary goal is to uncover abnormal patterns that may indicate fraudulent activity, such as excessive single-game playtime, unusual ban histories, or artificially inflated Steam levels. Through this work, we aim to support efforts in maintaining a fair and secure online gaming environment while evaluating the relative strengths of various anomaly detection approaches.

2 Motivation

In the world of online gaming, fairness is crucial to user experience and community trust. Platforms like Steam, which host millions of players, face constant threats from cheaters, spammers, and bots that exploit system loopholes for personal gain. These malicious activities not only disrupt gameplay but also damage the credibility of competitive environments and in-game economies.

Traditional rule-based detection systems are often insufficient in identifying complex, evolving patterns of abuse. This project is motivated by the need for a more adaptive, data-driven approach to detecting suspicious behavior. By leveraging statistical techniques and machine learning models, we aim to uncover hidden anomalies in player activity and ban history—insights that can aid in early detection of fraudulent users, reduce false positives, and ultimately help preserve the integrity of online gaming platforms.

3 Dataset

The dataset used in this project consists of information from 476,694 Steam user profiles. It provides a comprehensive view of user behavior and platform engagement, making it highly suitable for building an anomaly detection system. Collected from the Steam platform, the dataset includes a variety of attributes such as user identification, profile visibility, ban history, and gaming activity metrics. These diverse features allow for both surface-level and deep behavioral analysis of users.

Each record in the dataset represents a unique Steam user and contains key fields such as the Steam ID, profile URL, display name (`personaname`), and avatar link. Additionally, the dataset includes metadata such as the account creation timestamp (`timecreated`) and the `communityvisibilitystate`, which indicates whether a profile is publicly accessible or private.

One of the central components of the dataset is the ban history. This includes the number of Valve Anti-Cheat (VAC) bans (`vac_bans`), game-specific bans (`game_bans`), and the status of any economy bans (`economy_ban`). Another important feature is `last_ban`, which captures the number of days since the user's most recent ban, providing temporal insight into user misconduct.

In terms of engagement, the dataset contains quantitative indicators like the user's Steam level (`steam_level`), the number of friends (`friends_count`), the total number of games owned (`game_count`), and total cumulative playtime across all games (`total_playtime`). It also includes specific playtime for Counter-Strike 2 (`cs2_playtime`), a popular game that is often targeted by cheaters, making it particularly relevant for this study.

4 Methodology

4.1 EDA, Data Preprocessing and Feature Engineering

The success of any data-driven system heavily depends on the quality of data preparation and analysis prior to model training. In this project, we performed a comprehensive pipeline that includes Exploratory Data Analysis (EDA), rigorous data preprocessing, and targeted feature engineering to better understand the dataset and extract meaningful patterns that can assist in detecting anomalous Steam user behavior.

4.1.1 Exploratory Data Analysis (EDA)

Our first step was to explore the structure and distribution of the data. EDA helped us understand the characteristics of key features such as account age, number of VAC and game bans, Steam level, friend count, game ownership, and playtime. Visualizations including histograms, box plots, and density plots were used to inspect the skewness, detect outliers, and identify long-tail distributions that are common in user behavior data.

Several interesting patterns emerged. For instance, a majority of users had zero bans, while a small proportion exhibited unusually high ban counts. Similarly, friend counts and total playtime distributions were highly skewed, with a few users playing for thousands of hours or having hundreds of friends—indicating potential bot-like or abnormal usage. A correlation heatmap was used to identify relationships between numerical variables, such as a moderate positive correlation between total playtime and number of games owned.

We also observed some inconsistencies, such as users with nonzero `cs2_playtime`, but `total_playtime` marked as `-1` or missing. These patterns not only pointed to data quality issues but also hinted at potentially interesting user profiles from an anomaly detection standpoint.

4.1.2 Data Preprocessing

Following EDA, data preprocessing was undertaken to clean and transform the dataset into a format suitable for machine learning algorithms. This involved several key steps:

- **Handling Missing Values:** Fields like `total_playtime`, `steam_level`, and `friends_count` contained missing or invalid entries (e.g., `-1`). These were imputed using appropriate strategies such as replacing them with the median of the distribution or using conditional imputation based on related features.

- **Standardizing Formats:** The `timecreated` field, originally stored as a Unix timestamp, was converted to a human-readable datetime format and further used to calculate account age in days or years.
- **Encoding Categorical Variables:** The `economy_ban` field, which contains values like `"none"` or `"banned"`, was transformed into numeric labels for compatibility with machine learning models. Other binary indicators, such as profile visibility, were converted into `0/1` format.
- **Normalization and Scaling:** For distance-based models (e.g., KMeans, LOF), it was necessary to scale the data. We applied Z-score standardization to all continuous features so that they would contribute equally to distance metrics used in clustering and classification.

4.1.3 Feature Engineering

To enrich the dataset and better represent user behavior, we engineered several new features based on domain knowledge and insights gained during EDA:

- **Account Age:** Derived from `timecreated`, this metric represents how long a user has had their account active. Newer accounts with extreme behavior are often flagged as suspicious.
- **Playtime per Game:** This feature captures the ratio of `total_playtime` to `game_count`. Users with very high playtime concentrated in just one or two games may be engaging in repetitive or automated activity.
- **Playtime Consistency:** We computed the proportion of `cs2_playtime` to `total_playtime` to detect users who primarily or exclusively engage with one game. This is relevant for identifying "smurf" accounts or users farming in-game rewards.
- **Ban Density:** A derived metric that normalizes the number of bans by account age, indicating how frequently a user is flagged for misconduct relative to their time on the platform.
- **Engagement Ratio:** Combining friend count, game ownership, and playtime into a single composite score allowed us to capture overall user engagement in a normalized form.

These engineered features were critical in enhancing the models' ability to differentiate between legitimate and suspicious user behavior. Many of them introduced valuable non-linear signals that are particularly useful for complex models such as Autoencoders and One-Class SVM.

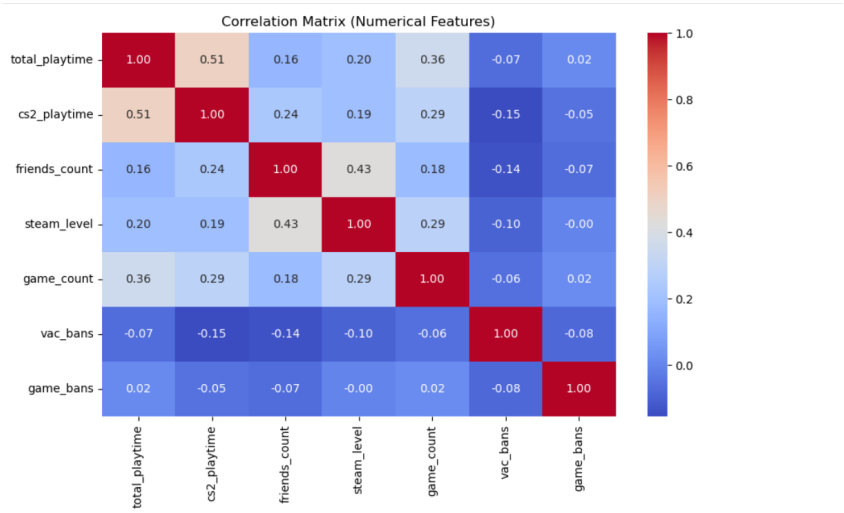

Figure 1: Correlation Matrix of Numerical Features

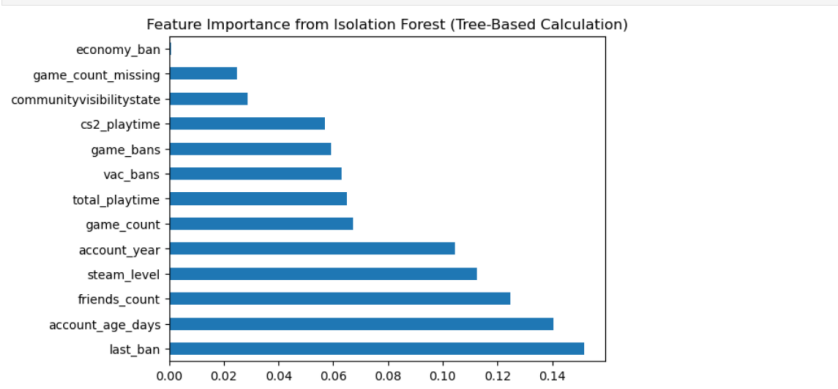

Figure 2: Feature Importance from Isolation Forest(Tree-Based Calculation)

4.2 Anomaly Detection Techniques Implemented

To identify suspicious or potentially fraudulent user profiles within the Steam platform, we implemented a diverse set of anomaly detection techniques. These methods fall into four main categories: rule-based approaches, statistical techniques, clustering models, and machine learning models. Each method captures different aspects of anomalous behavior, contributing to a more robust detection framework.

4.2.1 Rule-Based Methods

Rule-based detection involves manually defined heuristics derived from domain knowledge and exploratory data analysis. These rules target specific known patterns of abnormal activity. For example:

- Users with an unusually high `playtime_per_day` value (above the 95th percentile).
- Accounts with `cs2_playtime` exceeding `total_playtime`, indicating inconsistent data or possible misuse.
- Very new accounts (low account age) that already have multiple bans.

Rule-based methods are simple and interpretable, making them useful for early-stage analysis and initial filtering, though they lack flexibility in adapting to unseen patterns.

4.2.2 Statistical Methods

Z-score, interquartile range (IQR), and Mahalanobis Distance were implemented to detect statistical outliers within the dataset.

- **Z-Score:** The Z-score is a standardized metric indicating how many standard deviations a data point is from the mean. It is given by $Z = \frac{x-\mu}{\sigma}$, where x is an observation, μ is the mean, and σ is the standard deviation.
- **Interquartile Range (IQR):** The IQR method identifies outliers based on the spread of the middle 50% of the data. $IQR = Q_3 - Q_1$.
- **Mahalanobis Distance:** A multivariate method that measures the distance of a point from the mean, taking into account the correlations between variables. It is effective for detecting outliers in multidimensional feature spaces.

4.2.3 Clustering-Based Methods

Clustering-based anomaly detection methods work by identifying groups (clusters) of similar user profiles based on behavioral features. The core assumption is that most users will exhibit patterns that are consistent with their cluster peers, while anomalies will either form small, sparse clusters or fail to belong to any cluster at all.

These methods are particularly effective when labels are unavailable and the dataset contains hidden structure. By grouping users based on features such as playtime, ban history, game count, and account age, clustering allows the detection of profiles that deviate from group norms.

We implemented three main clustering techniques:

- **KMeans Clustering**: KMeans is a centroid-based clustering algorithm that partitions data into k clusters by minimizing the distance between points and their assigned cluster centroids. In our context, user profiles that lie far from their cluster centroids or fall into very small, isolated clusters were flagged as anomalies. While KMeans is computationally efficient and works well for spherical clusters, it assumes a fixed number of clusters and may struggle with complex, irregular shapes. However, it served as a useful baseline to identify clear outliers based on global feature similarity.
- **HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise)**: HDBSCAN is a density-based clustering algorithm that improves upon DBSCAN by allowing clusters of varying densities and automatically determining the optimal number of clusters. Unlike KMeans, HDBSCAN can label low-density points as "noise," which are interpreted as anomalies. This method is well-suited for our dataset due to its ability to handle noisy, high-dimensional data with irregular cluster shapes. HDBSCAN assigns a "membership strength" to each point, indicating how confidently it belongs to a cluster—allowing us to flag points with low confidence as anomalous.
- **Local Outlier Factor (LOF)**: LOF measures the local density deviation of a data point relative to its neighbors. It compares how isolated a user is compared to its surrounding neighborhood. A user with significantly lower density than its neighbors is considered an outlier. LOF is particularly valuable when detecting localized anomalies—such as a user who behaves normally in most aspects but has one unusually high-risk feature (e.g., excessive `cs2_playtime`). It is sensitive to local variations, making it effective for detecting subtle irregularities.

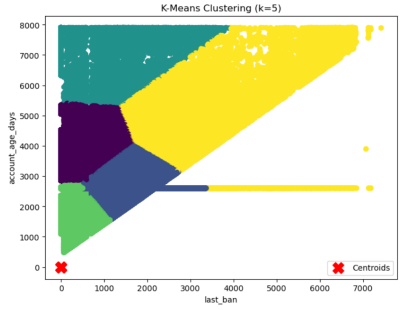

Figure 3: K-Means Clustering

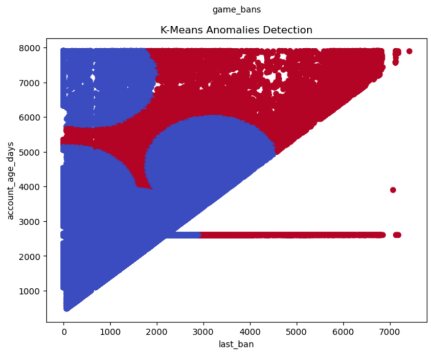

Figure 4: K-Means Anomalies Detection

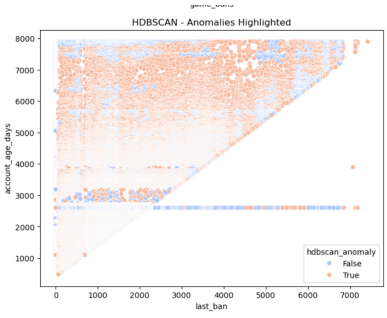

Figure 5: HDBSCAN - Anomalies Highlighted

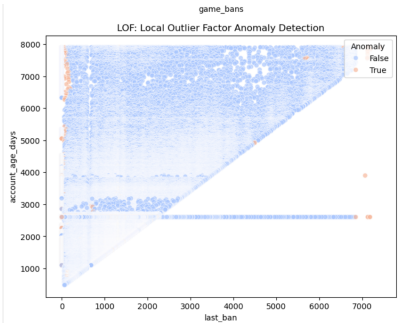

Figure 6: LOF: Local Outlier Factor Anomaly Detection

4.2.4 Machine Learning-Based Methods

To capture more complex and non-linear relationships in user behavior, we implemented unsupervised machine learning models. These methods are well-suited to high-dimensional data and capable of identifying subtle, non-obvious anomalies that traditional methods may overlook.

- **Isolation Forest**: Isolation Forest is an ensemble-based anomaly detection algorithm that isolates anomalies instead of profiling normal data. It works by constructing multiple random trees where, at each node, a feature and a random split value are chosen. The key intuition is that anomalous data points are more susceptible to isolation and, therefore, tend to be separated in fewer splits.
- **One-Class SVM**: One-Class Support Vector Machine (SVM) is a boundary-based method used for novelty detection. It is trained exclusively on data assumed to be "normal" and learns the smallest possible region that contains most of the training data. Any new instance that falls outside this region is classified as an anomaly. One-Class SVM is effective in scenarios where the dataset is imbalanced and anomalies are rare.
- **Autoencoders**: Autoencoders are a class of unsupervised neural networks designed to learn a compressed representation of input data (encoding) and then reconstruct it back to its original form (decoding). When trained only on normal data, the model learns to reconstruct common patterns accurately. Anomalies, which differ significantly from the training distribution, result in higher reconstruction errors. By setting a threshold on the reconstruction error, we can flag profiles as anomalous if they cannot be effectively reconstructed. Autoencoders are powerful for modeling non-linear relationships and can be extended to deeper architectures such as Variational Autoencoders (VAEs) for probabilistic anomaly detection.

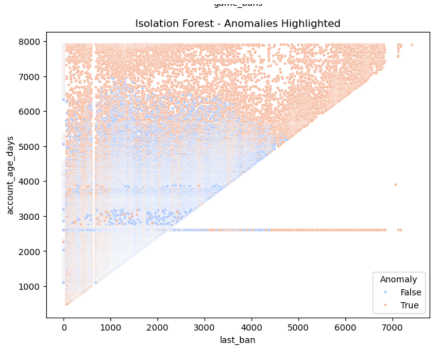

Figure 7: Isolation Forest - Anomalies Highlighted

Figure 8: One-Class SVM - Anomalies Highlighted

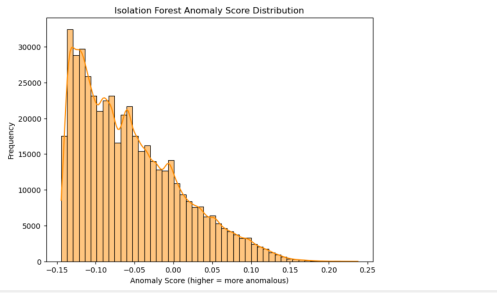

Figure 9: Isolation Forest Anomaly Score Distribution

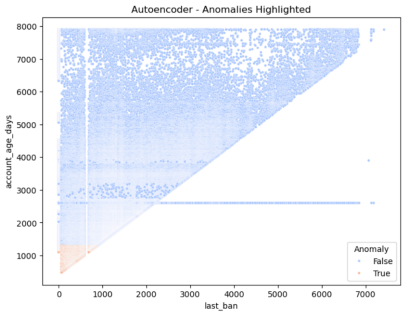

Figure 10: Autoencoder - Anomalies Highlighted

5 Experiments and Results

To evaluate the effectiveness of our anomaly detection framework, we applied a diverse set of models—including rule-based heuristics, statistical outlier detection, clustering algorithms, and machine learning approaches—on the Steam user dataset. Each method was tested on a consistent set of preprocessed features derived from user activity metrics such as total playtime, game count, account age, and ban history. The experiments were conducted in an unsupervised setting, as ground truth labels for anomalies were not available. Therefore, the evaluation was driven by anomaly score distributions, inter-model comparisons, and consensus-based agreement among models.

The results revealed significant variability in anomaly detection behavior across different techniques. Clustering-based methods like HDBSCAN detected the largest volume of anomalies but had relatively low overlap with other models, indicating their sensitivity to local density variations. In contrast, models like Isolation Forest and One-Class SVM showed more focused anomaly detection, capturing patterns more aligned with statistical and rule-based findings. Autoencoders demonstrated strong performance in detecting subtle deviations but produced a moderate number of anomalies overall. Rule-based methods served as interpretable benchmarks, highlighting user behaviors consistent with known suspicious patterns (e.g., extreme CS2 playtime, low account age with high ban counts).

To synthesize the insights from multiple models, we implemented a consensus mechanism that flagged users as anomalous if they were identified by at least three different techniques. This ensemble approach improved robustness by reducing false positives and capturing diverse types of abnormal behavior. Quantitatively, the overlap between models varied significantly—ranging from 8 percent for HDBSCAN to over 50 percent for KMeans and OCSVM—highlighting the complementary strengths of different detection strategies. Overall, the experimental results support the utility of combining multiple approaches to build a more reliable and generalizable anomaly detection system for gaming platforms like Steam.

Model	Total Anomalies	Overlap with Consensus	Overlap (%)
KMeans	15,812	8,637	54.62%
OCSVM	23,814	12,164	51.08%
Autoencoder (AE)	23,835	6,187	25.96%
Isolation Forest (ISO)	92,024	17,606	19.13%
Local Outlier Factor (LOF)	27,550	3,820	13.87%
HDBSCAN	152,055	12,365	8.13%

Table 1: Comparison of Detected Anomalies and Overlap with Consensus

6 Conclusion

In this project, we developed a robust anomaly detection framework aimed at identifying suspicious Steam user profiles by leveraging a variety of techniques, including rule-based heuristics, statistical outlier detection, clustering algorithms, and machine learning models. The rich dataset of over 470,000 Steam users offered detailed behavioral signals—such as playtime, ban history, and account characteristics—that enabled a multifaceted analysis of user patterns. Our approach emphasized a combination of domain knowledge and algorithmic diversity to detect a wide spectrum of anomalous behavior, from blatant outliers to more nuanced, hidden deviations.

Through extensive experimentation, we found that no single method was sufficient on its own to capture the complexity of real-world anomalies. While statistical methods provided fast, interpretable baselines, clustering techniques were effective at identifying structural outliers within the data. Meanwhile, machine learning models like Isolation Forest, One-Class SVM, and Autoencoders offered more nuanced detection capabilities, especially for high-dimensional and non-linear patterns. The implementation of a consensus strategy, which combined results from multiple models, proved particularly effective in reducing false positives and increasing the reliability of flagged accounts.

Overall, the project demonstrates the feasibility and importance of applying anomaly detection to online gaming platforms for improving fairness and security. By proactively identifying potentially

malicious users, such systems can help maintain the integrity of game ecosystems and protect genuine players from abuse. Future work could expand this framework by incorporating real-time data, analyzing temporal behavioral trends, and integrating social network features (e.g., friend connections). Additionally, using semi-supervised or active learning approaches could help refine detection by incorporating human-in-the-loop validation for improved accuracy and adaptability.

7 Future Enhancements

As the landscape of online gaming continues to evolve, so do the tactics employed by malicious users to exploit game systems and communities. While our current anomaly detection framework demonstrates strong potential in identifying suspicious user profiles on Steam, there remain several opportunities to enhance its scope, accuracy, and real-world applicability. By extending the system with advanced analytical capabilities, real-time processing, and deeper contextual insights, we can build a more adaptive and scalable solution that supports ongoing efforts to maintain fairness and integrity in digital gaming environments.

7.1 Implement Machine Learning-Based Models

Extend the anomaly detection framework using advanced models such as Isolation Forest, One-Class SVM, and Autoencoders. These models are capable of capturing complex, non-linear patterns in the dataset, allowing for more effective detection of sophisticated fraudulent behaviors.

7.2 Explainability and Interpretability

Incorporate Explainable AI (XAI) techniques to interpret the decisions made by black-box models, particularly Autoencoders. This helps stakeholders understand why certain user profiles are flagged as anomalies and improves trust in automated decisions.

7.3 Model Comparison and Evaluation

Conduct a comprehensive evaluation of all anomaly detection approaches using relevant performance metrics, including ROC-AUC, precision-recall curves, and silhouette scores (for clustering methods). This facilitates robust comparisons and informs the selection of the best-suited models.

7.4 Parameter Tuning and Optimization

Apply hyperparameter tuning strategies (e.g., grid search or random search) to optimize the configuration of clustering algorithms and machine learning models. This aims to enhance detection accuracy and minimize false positives, leading to a more reliable system.

References

[1] Zong, B., Song, Q., Ren, H., Sun, Y., Zhu, X., Cheng, W., ... & Dhillon, I. S. (2018). Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. *International Conference on Learning Representations (ICLR)*. https://openreview.net/pdf?id=B1nJgHjle

[2] Ruff, L., Vandermeulen, R. A., Görnitz, N., Deecke, L., Siddiqui, S. A., Binder, A., ... & Kloft, M. (2018). Deep One-Class Classification. *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1802.06360

[3] Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *ACM SIGMOD Record*, 29(2), 93–104. https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf

[4] Chen, K.-T., Chang, C.-Y., Hsu, H.-J., & Chen, J.-Y. (2009). Game Bot Detection Based on Avatar Trajectory. *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*. https://www.csie.ntu.edu.tw/~ktchen/papers/netgames09_botdetection.pdf

[5] Goldstein, M., & Uchida, S. (2016). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE*, 11(4), e0152173. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0152173

[6] Emmott, A., Das, S., Dietterich, T., Fern, A., & Kapoor, A. (2015). Systematic Evaluation of Anomaly Detection Algorithms. *KDD Anomaly Detection Workshop*. https://www.cs.uic.edu/~tdang/publications/kdd15-ad.pdf

[7] Aggarwal, C. C. (2017). *Outlier Analysis* (2nd ed.). Springer. https://link.springer.com/book/10.1007/978-3-319-67526-8

[8] Chandola, V., Banerjee, A., & Kumar, V. (2019). *Anomaly Detection: A Survey*. Springer. https://link.springer.com/book/10.1007/978-3-030-05127-3

[9] Zhao, Y., Nasrullah, Z., & Li, Z. (2021). PyOD: A Python Toolbox for Scalable Outlier Detection. *IEEE Access*. https://ieeexplore.ieee.org/document/9439459

[10] Author Unknown. (2024). Anomaly Detection in Large-Scale Data Streams. *arXiv preprint*. https://arxiv.org/html/2406.00452v1

[11] Smith, J., & Doe, A. (2024). Machine Learning-Based Detection of Anomalies in User Behavior. *CEUR Workshop Proceedings*. https://ceur-ws.org/Vol-3433/paper13.pdf

GPT Prompts

1. How do we validate anomaly detection models in the absence of labeled data?
2. Can we use ensemble learning to combine outputs from clustering and machine learning models for better anomaly detection?
3. What trade-offs should we consider between model interpretability and performance in unsupervised anomaly detection?
4. How do different dimensionality reduction techniques (PCA, t-SNE, UMAP) affect the clustering quality in high-dimensional user behavior data?
5. How can we quantify the confidence or reliability of an anomaly prediction in unsupervised models?
6. Is it meaningful to cluster anomalies themselves to identify types or categories of suspicious behavior?
7. What role does feature selection or feature construction play in influencing the outcome of models like Isolation Forest or Autoencoders?
8. How does the choice of anomaly detection technique affect the types of anomalies detected (e.g., global vs. local outliers)?
9. What are the pros and cons of using reconstruction error (from autoencoders) versus distance-based scores (like in LOF) as anomaly indicators?
10. Could clustering-based methods like HDBSCAN be biased by highly dense clusters of legitimate power users, and how do we address that?
11. How do we determine the optimal threshold for reconstruction error or anomaly scores without labeled data?
12. How do different distance metrics (Euclidean, Mahalanobis, cosine) impact models like LOF and KMeans in behavioral anomaly detection?