RoDAC: A Robust Data-centric Anti-Cheat Framework for Fair Online Competitive Gaming

Minsu Kim*
KAIST
ms716@kaist.ac.kr

Junwoo Park
KAIST
junwoo.park@kaist.ac.kr

Chanho Lee
KRAFTON
kami98@krafton.com

Gibum Seo
KRAFTON
jseo0703@krafton.com

Steven Euijong Whang
KAIST
swhang@kaist.ac.kr

Hyuck Lee†
KRAFTON
dlgur0921@krafton.com

Abstract

Online gaming platforms face significant challenges from cheating behaviors that threaten fair play and user trust. In particular, first-person shooter (FPS) games are especially vulnerable to a wide range of sophisticated cheating techniques that directly undermine competitive balance. Among various cheating techniques in FPS games, this work specifically focuses on detecting Extra-Sensory Perception (ESP) cheats, which reveal hidden game information to users and are difficult to detect using traditional rule-based methods. We formulate the ESP cheat detection task as a binary classification problem that aims to distinguish between legitimate users and cheaters, using a tabular dataset extracted from user-match logs. In practice, the tabular dataset presents three key challenges, including label noise arising from mislabeling of legitimate users and cheaters, severe class imbalance between them, and distribution shifts caused by evolving user behaviors, all of which degrade model performance. To address these challenges, we propose RoDAC, a Robust Data-centric Anti-Cheat framework that sequentially applies tailored sample selection methods to mitigate label noise, class imbalance, and distribution shifts. We have deployed RoDAC in the PUBG: BATTLEGROUNDS competitive league and confirmed that the framework significantly improves detection accuracy and robustness compared to baseline methods. Our detailed analysis of computational efficiency and model compatibility highlights the practical benefits of our data-centric strategy for sustaining scalable and reliable anti-cheat systems in dynamic gaming environments.

Keywords

Anti-Cheat, Cheat Detection, Model Robustness, Data-centric AI

1 Introduction

Online gaming has rapidly evolved into a massive global industry, attracting millions of users and generating significant economic and cultural impact [20, 21, 59]. However, the popularity of these platforms has also made them prime targets for cheaters who seek unfair advantages through unauthorized tools, scripts, or exploits [46, 78, 79]. The proliferation of such cheating behaviors not only undermines the integrity of the gaming experience but also erodes user trust, and it can lead to significant financial losses for game publishers [7, 11, 48]. As a result, the development and deployment of reliable anti-cheat systems to detect and prevent cheating have become critical priorities for the gaming industry [3].

Figure 1: In-game screenshot of an ESP cheater in PUBG: BATTLEGROUNDS. The cheater leverages ESP to identify enemy positions and precisely aim at an enemy behind a tree, which would otherwise remain unseen to legitimate users.

While various forms of cheating exist in online games, including aimbots, recoil scripts, and macros [19, 80], we specifically focus on detecting *Extra-Sensory Perception (ESP)* cheats [80] in *PUBG: BATTLEGROUNDS* due to their high prevalence and significant impact on competitive integrity. ESP cheats allow users to gain unfair advantages by revealing hidden information such as the real-time locations of opponents and items as shown in Figure 1, thereby removing the uncertainty intended by the game design [17]. Unlike aimbots or recoil scripts that automate aiming, ESP cheats preserve manual control, making them harder to detect using simple rule-based or statistical anomaly detection.

Recently, anti-cheat systems increasingly rely on machine learning models to detect abnormal patterns indicative of cheating [1, 42, 77]. The models are trained on vast amounts of user behavioral data and learned to distinguish between behaviors of legitimate users and cheaters as a binary classification task. For example, suspicious movement patterns, looting behaviors, or combat engagements that deviate from legitimate user behavior can serve as signals for potential cheating. However, the dynamic nature of online games presents unique challenges that complicate the development of robust anti-cheat systems. As cheaters continuously refine their strategies to evade detection and legitimate users exhibit diverse behaviors, the resulting behavioral data of both groups become similar, making the classification task more challenging.

To train machine learning models for anti-cheat purposes, the first requirement is the availability of appropriate data. In the gaming domain, potential data modalities include tabular, image, video, text, and speech data [6, 83]. While image, video, text, and speech

*Work done during an internship at KRAFTON.
†Corresponding author.

data provide rich information, they impose significant storage demands and require complex, high-capacity models for effective training. Considering the complexity of both model development and deployment, we opt to utilize tabular data, which typically demands less memory and enables the use of lightweight tree-based models by leveraging the explicit semantic meaning of each column without requiring additional feature embeddings [15, 44]. We extract informative features from user-match log data to represent user behaviors in the game and construct a tabular dataset comprising millions of samples and hundreds of features. This choice facilitates the practical deployment of anti-cheat systems while preserving adequate detection performance [8, 65].

However, there are three main challenges when using the constructed tabular dataset for model training. First, it is hard to obtain ground-truth labels for both legitimate users and cheaters. This challenge fundamentally stems from the absence of a definitive system that can accurately determine whether a user is legitimate or a cheater. We can leverage existing labels generated for other purposes from user monitoring systems and a whitelist of verified users for a subset of samples. However, these labels may not precisely align with our target ESP cheats and may introduce label noise, since they were originally generated for different purposes based on different criteria. Second, even when ground-truth labels are available, there exists a severe class imbalance between legitimate users and cheaters in the dataset. In general, the number of legitimate users significantly exceeds that of cheaters. In general, models trained on the imbalanced dataset tend to bias their predictions toward the legitimate class, failing to accurately detect cheaters. Third, distribution shifts occur in the dataset as user behaviors and cheating strategies change over time. Such shifts are particularly evident during major events, including new season launches or game updates. When distribution shifts occur between the training and inference data, trained models may fail to make accurate predictions on the inference data.

While existing works have explored methods to address label noise, class imbalance, and distribution shifts independently, there is a lack of a holistic framework that tackles all three challenges specifically in the context of anti-cheat systems. We propose a holistic data-centric framework called RoDAC to address these challenges and improve the performance of anti-cheat models. As labeling newly collected training samples requires approximately one week, our anti-cheat system retrains the model on a weekly basis. After labeling, RoDAC sequentially performs a series of sample selection methods on the labeled training data, each designed to address label noise, class imbalance, and distribution shifts. To mitigate label noise, RoDAC selects training samples by verifying the alignment between the given labels and the predictions of the model trained in the previous week. RoDAC next performs coreset-based undersampling on the training samples with legitimate labels to mitigate the severe class imbalance between legitimate users and cheaters while minimizing information loss. In addition, to reduce the gap between the training and inference distributions, RoDAC removes out-of-distribution (OOD) training samples based on the distribution of inference samples. After completing the three sequential steps of sample selection followed by model training, RoDAC further applies a similar OOD data filtering strategy during inference phase that removes OOD inference samples based on the distribution of training samples. This strategy prevents the model from making predictions on ambiguous OOD inference samples.

RoDAC has been deployed in the PUBG: BATTLEGROUNDS competitive league, where it outperforms the sequential combination of compatible state-of-the-art methods in terms of post-launch performance. Our experimental analysis shows that each proposed sample selection method contributes to improving model performance. We further analyze computational time during the training and inference phases, compatibility with various backbone models, and the handling of users who appealed their bans after being detected as cheaters to provide a better understanding of our anti-cheat system. We believe that our findings highlight the necessity and effectiveness of data-centric strategies in sustaining robust model performance under challenging real-world environments.

Summary of Contributions: (1) We identify three critical data challenges, namely label noise, class imbalance, and distribution shifts, that hinder robust anti-cheat model performance in online gaming; (2) We propose RoDAC, a holistic data-centric framework that effectively mitigates these data challenges through tailored sample selection methods; (3) We empirically validate the effectiveness and efficiency of RoDAC by deploying the framework as the live production anti-cheat system of the PUBG: BATTLEGROUNDS competitive league.

2 Related Work

Anti-Cheat Systems. Detecting and preventing cheating in online games has been an active area of research due to its critical importance for maintaining fair play and user engagement [7, 86]. Cheating behaviors in online games are diverse, including actions such as ESP cheats, aimbots, and exploiting unintended bugs or glitches for unfair advantages [19, 80]. Traditional anti-cheat systems primarily rely on rule-based detection, hardware attestation, and behavioral heuristics [14, 72, 80]. However, these approaches often struggle to adapt to the evolving nature of cheats, particularly in large-scale online environments. Recently, the use of machine learning for anti-cheat detection has gained traction, leveraging user behavior patterns to identify suspicious activities indicative of cheating [1, 42, 77, 81]. Vision-based approaches have also been explored to detect cheating behaviors by analyzing gameplay video streams and detecting overlays or suspicious in-game visual patterns [41, 85]. Despite these advancements, existing machine learning-based anti-cheat systems often overlook data challenges such as label noise, class imbalance, and distribution shifts in user behavior data, which are crucial for ensuring sustained detection performance in dynamic and adversarial environments.

Learning with Tabular Data. Tabular data is a prevalent modality in anti-cheat systems due to its structured nature and ease of integration with lightweight models that support real-time inference [1, 48]. Tree-based models, including Gradient Boosting Decision Trees (GBDT) [15, 23, 44, 63], are widely used for tabular data learning and often outperform deep learning methods on diverse tabular datasets [65]. Recent studies have explored enhancing deep learning performance on tabular data [4, 28, 33, 36, 82], but tree-based methods remain the preferred choice in many industrial settings due to their interpretability, efficiency, and scalability [8].

Following the established effectiveness of tree-based models in cheat detection [24, 69], we employ tree-based models to train the tabular dataset derived from our user-match log data. Specifically, we use structured features extracted from user behaviors, including movement, looting, and combat patterns.

Model Robustness. Ensuring the robustness of machine learning models is critical for their reliable deployment in real-world environments [9, 49]. In practice, model robustness can be undermined by challenging data conditions, including label noise, class imbalance, and distribution shifts. Label noise arises when the assigned labels fail to accurately reflect the true labels of samples due to annotation errors, sample ambiguity, or systematic labeling biases [22, 68]. To mitigate the adverse effects of label noise on model performance, conventional machine learning techniques [26, 57, 66, 74] and deep learning approaches [27, 31, 37, 61, 84] have been proposed.

Class imbalance is another common data challenge, referring to the disproportionate number of samples across classes. It can cause models to be biased toward the majority class and result in poor detection of the minority class [16, 40]. Existing work addressing class imbalance can be categorized into three groups [47]: data-level [12, 62], algorithm-level [45, 54, 76], and hybrid [2, 18, 34] approaches. Data-level methods rebalance distributions by adjusting the number of samples, algorithm-level methods adapt learning algorithms to mitigate the bias towards majority classes, and hybrid methods combine both strategies.

Distribution shifts refer to changes in the data distribution between the training and test phases, which can arise from domain differences, temporal variations, or evolving data generation processes. These shifts pose significant challenges in dynamic real-world scenarios, as models trained on the training data distribution often fail to generalize to the shifted test data distribution. To bridge the distribution gap and maintain model generalization, approaches such as domain generalization [13, 30, 51], domain adaptation [5, 25, 32, 39], and test-time adaptation [53, 70, 75] have been proposed.

As various data challenges frequently co-occur in real-world data [29], addressing them within a holistic framework is essential for sustaining robust model performance. However, methods for handling these data challenges are often investigated independently, and their integration and joint impact on anti-cheat systems remain underexplored. In contrast, our proposed simple yet effective data-centric approach sequentially addresses label noise, class imbalance, and distribution shifts problems within a holistic framework.

3 Problem Definition

We formulate the ESP cheat detection task in PUBG: BATTLEGROUNDS as a binary classification problem on large-scale competitive league match data streams, where the goal is to predict whether a user is a legitimate user or a cheater. Since new data streams arrive on a daily basis, it is necessary to periodically retrain the model rather than using a static model trained only once, as the game environment and user behavior patterns continuously evolve. In our anti-cheat system, we set the model retraining schedule to once per week. We describe the data pipeline for model training and inference each week as shown in Figure 2. Assuming that more recent data are more relevant to the current data, we use training data from the past 14 weeks and employ a subset of the training

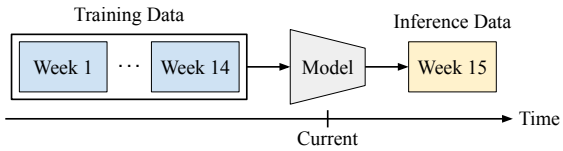


Figure 2: The data pipeline of our anti-cheat system.

data from the previous week for model validation. After training, we use the trained model to predict on the inference data from the subsequent days.

We set notations for data in our anti-cheat system. Let $\mathcal{D}_{train} = \{(X_i, y_i)\}_{i=1}^{n}$ denote the training dataset, where $X_i \in \mathbb{R}^d$ is the d-dimensional feature vector extracted from a match log of a user, $y_i \in \{0, 1, \texttt{NaN}\}$ indicates whether the user is a legitimate user ($y_i = 0$), a cheater ($y_i = 1$), or unlabeled ($y_i = \texttt{NaN}$) in the match, and n is the number of training samples. Similarly, we define the unlabeled inference dataset as $\mathcal{D}_{infer} = \{(X_j, y_j)\}_{j=1}^{m}$, where the size of the inference dataset is significantly larger than that of the training dataset (i.e., $m \gg n$). Since it is infeasible to obtain real-time labels for cheaters in the inference phase, only a subset of legitimate users can be labeled based on a whitelist of verified users (i.e., $y_j \in \{0, \texttt{NaN}\}$). The feature vectors X_i and X_j are composed of *in-game features* capturing user actions, movement patterns, and combat statistics at an early game phase to minimize harm to legitimate users, as well as *historical features* reflecting the past gameplay records of users, enabling robust detection under limited short-term observation of in-game features.

Given this environment setup, our goal is to train a model on the training data and make accurate predictions on the inference data. However, two practical challenges arise under the given environment setup, corresponding to the training and inference data, respectively. First, tree-based models are typically trained in a supervised manner using fully labeled data, whereas our training data consists of a mixture of labeled and unlabeled samples. Second, the true labels for the inference data are not fully available as only partial labels corresponding to legitimate users are provided, which makes evaluation challenging. To address these challenges, we leverage the statistical information contained in the unlabeled training data to improve the quality of the labeled training data. In addition, we evaluate the trained model on the inference data using an approximated False Discovery Rate (FDR), defined as the proportion of false positives among all positive predictions.

4 Framework

We describe the overall framework of RoDAC as shown in Figure 3. RoDAC first collects both in-game and historical user data from the corresponding database and joins them to generate the initial training data. RoDAC next sequentially addresses the data challenges of the training data in the order of label noise, distribution shifts, and class imbalance. Specifically, RoDAC performs three data selection methods corresponding to each challenge on the training data: label noise cleaning (LNC), OOD data filtering (ODF), and weighted under-sampling (WUS). The resulting refined training data is then used to train the model. In the inference phase, RoDAC applies similar OOD data filtering (ODF) to the inference data, making predictions only on the inference data that are similar to the refined training data. This series of methods within the framework

enables our anti-cheat system to remain robust in dynamic real-world environments by maintaining high-quality data over time. We describe the detailed methods applied in each step of RoDAC in the following sections.

4.1 Feature Extraction

User data of PUBG: BATTLEGROUNDS are primarily divided into in-game data and historical data, each stored in separate databases. The in-game data are extracted in real time from match logs, whereas the historical data are aggregated on a daily basis from user statistics. For a subset of users, labels indicating legitimate users or cheaters are assigned to these data on a weekly basis. Based on this data pipeline, we extract attributes related to the in-game behavior and historical record of users, along with their corresponding labels, from the two databases using Spark SQL queries. The extracted attributes are then transformed into meaningful in-game and historical features for model training. During this process, we employ a feature validator to filter out samples with values outside the expected range for each feature. For certain features that exhibit a substantial number of missing values, we address this issue through data imputation using either domain knowledge to assign specific values or feature-wise mean values. As a final step, the in-game and historical feature sets are merged to construct a comprehensive training data that represents user characteristics. We note that feature correlations are low by our prior feature selection process.

4.2 Label Noise Cleaning (LNC)

To identify the presence of label noise in the training data, we employ a simple and effective method that leverages our model trained in the previous week, motivated by prior works that utilize predictions of previously trained models [38, 60]. To verify the effectiveness of using predicted probabilities as a criterion, we present the distributions of the labels and the predicted probabilities in Figure 4a. The labels are binary (0 or 1), while the predicted probabilities are continuous values between 0 and 1. We observe that most training samples are concentrated near the extremes of the probability range, specifically within [0, 0.2] and [0.8, 1]. The overall alignment between the distributions of labels and predicted probabilities suggests that the predicted probabilities can accurately reflect labels in most cases.

Based on these observations, we identify training samples with label noise by detecting the misalignment between the predicted probabilities and the labels using a prediction threshold. Since we generally set the threshold for model predictions higher than the midpoint probability of 0.5 to more confidently identify cheaters, we use an arbitrary threshold of 0.8 and present the distribution of training samples for which the model predictions disagree with the labels as shown in Figure 4b. As a result, 1,995 cheater samples exhibit relatively low predicted probabilities (red bars), while 355 legitimate user samples show relatively high predicted probabilities (blue bars) out of the 210,918 labeled training samples. In the first case (red bars), a subset of labeled cheaters may not actually use ESP cheats in the game (e.g., those identified through unrelated hardware manipulations), resulting in behavior similar to that of legitimate users and consequently leading to low predicted probabilities. In the second case (blue bars), certain verified legitimate users may demonstrate gameplay so proficient that it mimics the

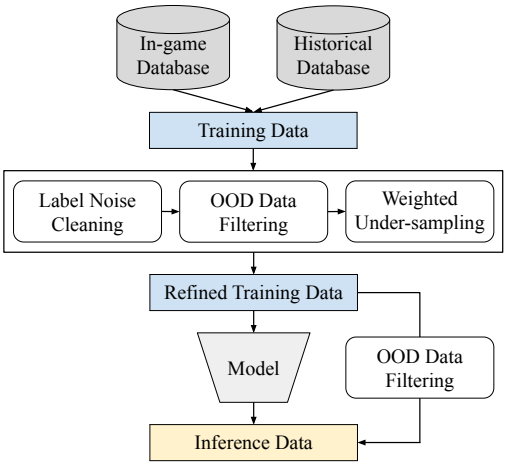
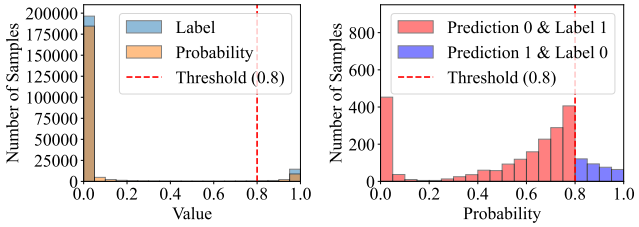


Figure 3: The overall framework of RoDAC.



(a) Label and predicted probability distribution. **(b) Noisy label distribution based on the prediction threshold.**

Figure 4: Noisy label detection results based on the difference between label and predicted probability.

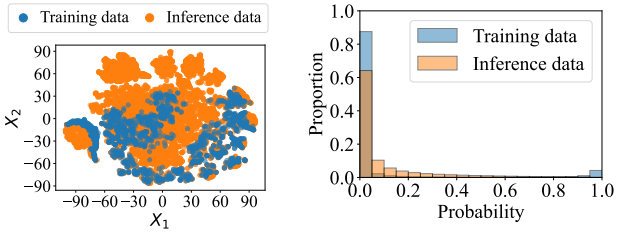
behavior of cheaters (e.g., professional gamers), resulting in high predicted probabilities. We thus define the samples corresponding to the first case (red bars) as noisy data and remove them from the training data. Formally, we obtain the cleaned training dataset $\mathcal{D}'_{train}$ by filtering out noisy cheater samples as follows:

$$\mathcal{D}'_{train} = \{(X_i, y_i) \in \mathcal{D}_{train} \mid \neg(y_i = 1 \wedge \hat{p}_i < \tau)\}, \quad (1)$$

where $\hat{p}_i$ is the predicted probability and τ is the threshold. If data drift occurs and reduces overall model confidence, existing drift detection methods can identify the shift, while lowering the threshold helps prevent new cheating patterns from being discarded.

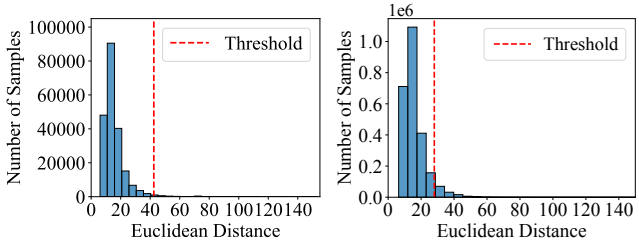
4.3 OOD Data Filtering (ODF) in Training Data

As labels of training data are typically obtained for users with certain status as legitimate or cheating, the labeled training data exhibits a different distribution relative to the overall distribution of inference data. Figure 5a illustrates the distribution shift between them using t-SNE [73]. To analyze how this shift affects model predictions, we compare the predicted probabilities on the labeled training data and the inference data using our model trained in the previous week as shown in Figure 5b. While the model produces confident predictions with probabilities close to 0 or 1 on the labeled training data, it yields more ambiguous probabilities on the inference data. We note that this pattern holds regardless of whether the labeled training data overlap with the training data of the trained model.



(a) t-SNE visualization. (b) Predicted probability.

Figure 5: Distribution shifts between labeled training and inference data.



(a) ODF in Training Data. (b) ODF in Inference Data.

Figure 6: Distance distributions of (a) training samples w.r.t. a centroid of proxy inference data and (b) inference samples w.r.t. a centroid of refined training data.

To reduce the distribution gap between labeled training data and inference data, we leverage the concept of out-of-distribution (OOD) detection to identify and discard OOD training samples from the perspective of the inference data. However, since the inference data is not available in advance, we use separate recent-date data, which includes both labeled and unlabeled samples, as a proxy for the inference data. Specifically, we employ a distance-based OOD detection method [35, 50, 71] that computes the Euclidean distance between each labeled training sample and the centroid of the proxy inference data. We can use multiple centroids for multimodal distributions, but our inference data is largely unimodal with most samples near the global centroid as shown in Figure 5a. The distance threshold for identifying OOD samples is adaptively determined from the resulting distance distribution based on its mean and standard deviation. Specifically, we define the OOD-filtered training dataset $\mathcal{D}''_{train}$ using the k-sigma rule as follows:

$$\mathcal{D}''_{train} = \{(X_i, y_i) \in \mathcal{D}'_{train} \mid \|X_i - \tilde{X}_c\|_2 \leq \mu + k\sigma\}, \quad (2)$$

where $\tilde{X}_c$ denotes the centroid of the proxy inference data, and μ and σ denote the mean and standard deviation of the distances, respectively. In practice, the results of OOD data filtering in the training data using the 3-sigma rule are shown in Figure 6a, where labeled training samples with distances exceeding the threshold set as three standard deviations above the mean are considered OOD. From this process, approximately 1% of the labeled training samples are identified as OOD and removed from the training data.

4.4 Weighted Under-sampling (WUS)

In gaming environments, legitimate users typically constitute a majority relative to cheaters, which results in substantial class imbalance in the training data. For example, the imbalance ratio between the two classes of majority class (legitimate user) and minority class (cheater) in the training data is about 10 in season 34 and 25 in season 35 as shown in Figure 7. Since class imbalance generally makes it difficult for the model to learn the minority class, data-level methods mitigate this issue by applying under-sampling to the majority class or over-sampling to the minority class to balance the classes. We initially apply various over-sampling techniques, but observe that duplicating samples yields no performance gain, while synthetic samples degrade model performance. In comparison, under-sampling techniques empirically shows more stable and consistent performance improvement, motivating us to adopt under-sampling in our approach.

We design two key components for under-sampling, namely the sample selection strategy and the target imbalance ratio. We propose a sample selection strategy that jointly accounts for relevance to the inference data and the diversity of the selected samples, motivated by distance-based weighted under-sampling [43]. Using the separate recent-date data as a proxy for the inference data, we assign pseudo-labels to its unlabeled samples with our model trained in the previous week. We then compute the Euclidean distance between each majority class (legitimate user) sample in the original training data and the centroid of the corresponding class samples in the proxy inference data. Using these distances as a metric of relevance, we perform weighted under-sampling that assigns sampling probabilities to each majority class (legitimate user) sample inversely proportional to their distance values. Specifically, we define the final balanced training dataset $\mathcal{D}'''_{train}$ as follows:

$$\mathcal{D}'''_{train} = \{(X_i, y_i) \in \mathcal{D}''_{train} \mid y_i = 1 \vee (y_i = 0 \wedge b_i = 1)\}, \quad (3)$$

where $b_i \sim \text{Bernoulli}(p_i)$ with probability $p_i = \min\left(1, \frac{\alpha}{\|X_i - \tilde{X}_{c,y=0}\|_2}\right)$. Here, $\tilde{X}_{c,y=0}$ denotes the majority class centroid of the proxy inference data, and α is a scaling factor adjusted to satisfy the target imbalance ratio. This approach reduces the number of majority class (legitimate user) samples by selecting samples that are more relevant to the inference data while maintaining diversity among the selected samples as shown in Figure 8.

Regarding the target imbalance ratio, achieving a low imbalance ratio requires removing a large number of majority class (legitimate user) samples, which can cause information loss for legitimate users and increase false positives. In contrast, a high imbalance ratio results in minority class (cheater) being underrepresented relative to majority class (legitimate user), which can prevent sufficient learning of cheaters and increase false negatives. We thus conduct a grid search over a range of target imbalance ratios to identify the value that generally yields the best model performance.

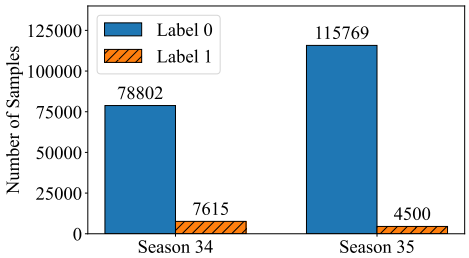
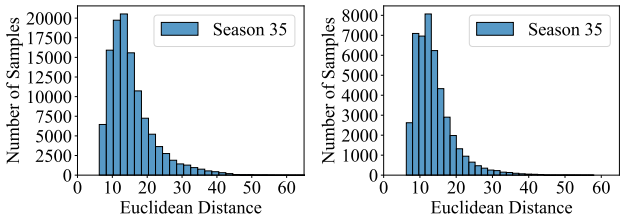


Figure 7: Class imbalance between label 0 (legitimate user) and label 1 (cheater).



(a) Original. (b) Under-sampled.

Figure 8: Distance distributions of majority class (legitimate user) training samples before and after under-sampling.

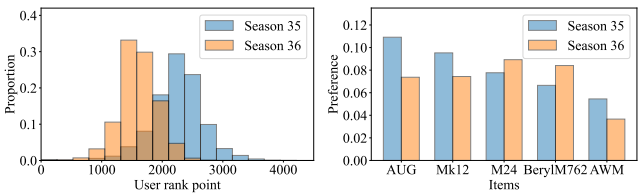
4.5 OOD Data Filtering (ODF) in Inference Data

Following a sequential sample selection process on the training data, the resulting refined training data is used for supervised learning to develop our anti-cheat model. Although we attempt to mitigate the distribution gap between the training data and the proxy inference data using OOD data filtering and weighted under-sampling, the actual inference data at test time may still exhibit a different distribution from the proxy inference data. In particular, when the training and inference data are collected from different game seasons, the distribution shift becomes more pronounced as shown in Figure 9. Test-time adaptation (TTA) [52] is a relevant research direction that addresses this problem by adapting a pre-trained model to unlabeled test data before making predictions. However, the applicability of test-time adaptation methods to our streaming tabular data scenarios is limited, since they are primarily developed for offline settings with deep learning models.

Instead, we suggest a simple data-centric approach that reuses the concept of OOD detection to abstain from making predictions on inference samples identified as OOD with respect to the refined training data. To this end, we compute the Euclidean distance between each inference sample and the centroid of the refined training data. Formally, we obtain the OOD-filtered inference dataset $\mathcal{D}'_{infer}$ as follows:

$$\mathcal{D}'_{infer} = \{(X_j, y_j) \in \mathcal{D}_{infer} \mid \|X_j - X_c\|_2 \leq \eta\}, \tag{4}$$

where X_c denotes the centroid of the refined training data, and η denotes a distance threshold. Since obtaining the overall distance distribution of the inference data is infeasible in a streaming scenario, we predefine the distance threshold for OOD detection using the 3-sigma rule applied to the distance distribution of the training data. As a result, predictions are withheld for approximately 5% of the inference samples identified as OOD as shown in Figure 6b. This can be interpreted as the model avoiding predictions on uncertain samples that differ from the training data, thereby reducing the risk of false positives.



(a) User rank point. (b) Item preference.

Figure 9: Distribution shifts with respect to game season.

Table 1: A competitive league dataset from PUBG: BATTLEGROUNDS, divided into two experimental settings. For each setting, we show the data type, duration, size, number of features, and number of classes.

Setting	Type	Duration	Size	#Ftrs	#Cls
Season 35 - Season 35	Training	98 Days	195,836	241	2
	Inference	5 Days	8,505,795	241	-
Season 35 - Season 36	Training	98 Days	210,918	241	2
	Inference	5 Days	14,031,270	241	-

5 Experiments

We have deployed RoDAC as the live production anti-cheat system of PUBG: BATTLEGROUNDS for over one year. We perform experiments to evaluate RoDAC using disjoint training and inference datasets. For each experiment, we report the mean and standard deviation of daily model performance across inference days. Our anti-cheat system is implemented in Python and PySpark on Databricks, and all experiments are conducted on Databricks clusters provisioned with r5d.16xlarge CPU instances.

5.1 Experimental Settings

Metrics. Given that obtaining ground-truth labels is challenging as cheaters do not disclose their cheating and it requires extensive manual investigation, we employ evaluation metrics that are continuously monitored in our live production environment. We evaluate methods using the approximated number of false positive users (#FP), the number of detected users (#Detect), and the approximated False Discovery Rate (FDR), defined as the ratio of #FP to #Detect, on the inference data. Since the inference data lack cheater labels and only include partial legitimate user labels, conventional metrics such as accuracy or F1-score cannot be computed, and the number of false positives is approximated based on their observed proportion. Our practical metrics provide meaningful insight into model performance: a lower number of false positive users indicates more accurate predictions, a higher number of detected users reflects broader coverage, and a lower FDR denotes a smaller proportion of false positive users among the detected users. We note that reported metrics reflecting actual service performance.

Dataset. We use a competitive league dataset from PUBG: BATTLEGROUNDS, a large-scale online competitive game where users compete to survive while collecting items under a dynamically shrinking safe zone. We consider two experimental settings using the dataset as shown in Table 1: one where the training and inference phases occur within the same game season (e.g., Season 35 - Season 35), and another where the training and inference phases span different game seasons (e.g., Season 35 - Season 36). Since game updates between seasons can cause significant changes in the data distribution, predicting inference data from a subsequent season poses a more challenging task. Following our anti-cheat system pipeline, we train a model each week using labeled training data from the past 98 days and make predictions on inference data from the subsequent days.

Models. We primarily adopt XGBoost [15] as the backbone model due to its demonstrated effectiveness on tabular data [65]. In particular, XGBoost has been widely used as a strong baseline for tabular

Table 2: Average approximated number of false positive users (#FP), average number of detected users (#Detect), and the average approximated false discovery rate (FDR) on the PUBG: BATTLEGROUNDS competitive league dataset. We compare the performance by sequentially incorporating the proposed sample selection methods and compatible state-of-the-art baselines: label noise cleaning (LNC and MentorNet), OOD data filtering in training data (ODF (train) and JTT), weighted under-sampling (WUS and IHT), and OOD data filtering in inference data (ODF (infer) and iForest).

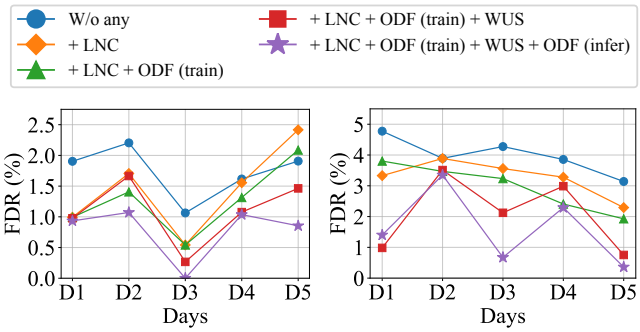
Method	Season 35 - Season 35			Season 35 - Season 36		
	#FP	#Detect	FDR (%)	#FP	#Detect	FDR (%)
W/o any sample selection methods	140.3±39.2	8204.6±1794.1	1.738±0.385	322.9±40.4	8146.6±778.1	3.987±0.537
+ LNC	123.6±68.9	8249.8±1977.7	1.442±0.639	265.5±37.9	**8191.2±783.6**	3.269±0.533
+ LNC + ODF (train)	108.8±57.2	8324.0±2020.2	1.268±0.508	237.1±40.4	8139.8±752.7	2.967±0.695
+ LNC + ODF (train) + WUS	91.8±48.2	**8373.2±2034.7**	1.089±0.481	168.1±87.6	8155.0±876.7	2.071±1.080
+ LNC + ODF (train) + WUS + ODF (infer) (RoDAC)	**63.5±38.2**	8251.0±1990.2	**0.778±0.396**	**127.8±82.8**	8154.6±856.4	**1.615±1.093**
+ MentorNet	133.5±51.4	8220.5±1810.3	1.624±0.592	294.1±45.1	8165.4±790.5	3.602±0.550
+ MentorNet + JTT	125.8±49.8	8235.1±1855.6	1.527±0.601	286.2±42.3	8158.2±785.2	3.508±0.545
+ MentorNet + JTT + IHT	113.4±55.2	8260.4±1910.1	1.372±0.645	253.9±60.5	8162.5±860.1	3.110±0.850
+ MentorNet + JTT + IHT + iForest	101.2±44.6	8245.8±1880.5	1.226±0.558	237.5±65.2	8155.9±855.8	2.912±0.910

data tasks owing to its scalability and generalization ability [8]. To evaluate the compatibility of our proposed data-centric framework across various backbone models, we further conduct experiments by replacing XGBoost with alternative models, including LightGBM [44], CatBoost [63], Random Forests [10], and Multi-Layer Perceptron (MLP) [64] in Sec. 5.3.

Hyperparameters. We conduct a grid search to optimize the hyperparameters of each model for consistently strong performance on our dataset. For the XGBoost model, we set the hyperparameters as follows: maximum depth = 10, minimum child weight = 10, gamma = 1, subsample = 0.7, column subsample by tree = 0.9, learning rate = 0.01, and the number of estimators = 2000. We set the target imbalance ratio for weighted under-sampling to 6 based on a grid search over {5, 6, 7, 8, 9}. To conservatively detect only users with high certainty of cheating, we adjust the prediction threshold from the default probability of 0.5 to a higher value. Specifically, we increase the threshold from 0.9 to 1.0 in increments of 0.0025 to observe the tendency of the approximated number of false positive users and the number of detected users as the threshold varies.

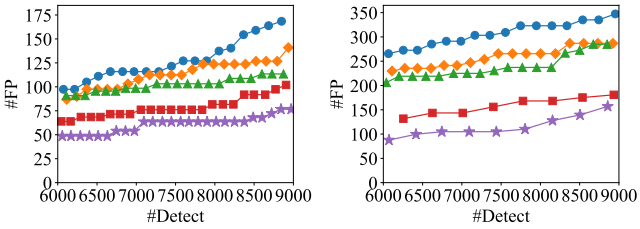
5.2 Experimental Results

To verify the effectiveness of each component in RoDAC, we design five ablation scenarios based on the sequential order of the four sample selection methods: (1) without any sample selection methods (W/o any), (2) with label noise cleaning (+ LNC), (3) with OOD data filtering in training data after LNC (+ LNC + ODF (train)), (4) with weighted under-sampling after LNC and ODF (train) (+ LNC + ODF (train) + WUS), and (5) with OOD data filtering in inference data after LNC, ODF (train), and WUS (+ LNC + ODF (train) + WUS + ODF (infer)). We compare the performance of these five ablation scenarios with corresponding state-of-the-art methods, using XGBoost as the backbone model, on the competitive league dataset from PUBG: BATTLEGROUNDS with respect to the average approximated number of false positive users (#FP), the average number of detected users (#Detect), and the average approximated false discovery rate (FDR) as shown in Table 2. Since the applicability of most state-of-the-art methods to tree-based models is limited, we adopt MentorNet [38], JTT [55], IHT [67], and iForest [56] as the compatible state-of-the-art methods for each respective component. For a fair comparison, we evaluate model performance in terms of #FP and FDR while aligning #Detect across methods. We also provide the corresponding sequential FDR results of RoDAC over five inference days as shown in Figure 10.



(a) Season 35 - Season 35. **(b) Season 35 - Season 36.**

Figure 10: Sequential FDR results over five inference days.



(a) Season 35 - Season 35. **(b) Season 35 - Season 36.**

Figure 11: #FP at various #Detect levels.

According to the results in Table 2, RoDAC achieves the best performance by obtaining the lowest #FP and FDR values while maintaining comparable #Detect values. The comparison with ablation scenarios suggests that addressing each data problem is critical for improving model performance. In addition, RoDAC outperforms the sequential combination of disjointed state-of-the-art methods, suggesting that it serves as a unified framework where each component is seamlessly integrated based on data distribution. We also

Table 3: Average approximated number of false positive users (#FP), average number of detected users (#Detect), and the average approximated false discovery rate (FDR) on the PUBG: BATTLEGROUNDS competitive league dataset when varying the backbone models. We use XGBoost, LightGBM, CatBoost, Random Forests, and MLP as backbone models.

Method	Season 35 - Season 35			Season 35 - Season 36		
	#FP	#Detect	FDR (%)	#FP	#Detect	FDR (%)
XGBoost	140.3$_{\pm 39.2}$	8204.6$_{\pm 1794.1}$	1.738$_{\pm 0.385}$	322.9$_{\pm 40.4}$	8146.6$_{\pm 778.1}$	3.987$_{\pm 0.537}$
+ RoDAC	**63.5**$_{\pm 38.2}$	**8251.0**$_{\pm 1990.2}$	**0.778**$_{\pm 0.396}$	**127.8**$_{\pm 82.8}$	**8154.6**$_{\pm 856.4}$	**1.615**$_{\pm 1.093}$
LightGBM	163.0$_{\pm 71.1}$	8189.2$_{\pm 1810.7}$	1.926$_{\pm 0.643}$	283.3$_{\pm 93.2}$	8060.4$_{\pm 748.5}$	3.590$_{\pm 1.354}$
+ RoDAC	**92.3**$_{\pm 35.7}$	**8270.6**$_{\pm 1962.1}$	**1.143**$_{\pm 0.361}$	**157.4**$_{\pm 54.6}$	**8071.8**$_{\pm 797.3}$	**1.982**$_{\pm 0.724}$
CatBoost	139.9$_{\pm 52.7}$	8271.6$_{\pm 2096.9}$	1.653$_{\pm 0.366}$	426.9$_{\pm 81.6}$	8225.0$_{\pm 679.7}$	5.241$_{\pm 1.191}$
+ RoDAC	**95.7**$_{\pm 39.4}$	**8276.8**$_{\pm 1987.3}$	**1.196**$_{\pm 0.439}$	**232.0**$_{\pm 28.3}$	**8450.2**$_{\pm 640.7}$	**2.763**$_{\pm 0.423}$
Random Forests	717.1$_{\pm 279.6}$	8145.0$_{\pm 1639.3}$	8.526$_{\pm 2.147}$	998.8$_{\pm 178.5}$	8326.2$_{\pm 553.1}$	12.015$_{\pm 2.162}$
+ RoDAC	**657.1**$_{\pm 184.7}$	**8174.8**$_{\pm 1635.0}$	**7.936**$_{\pm 0.879}$	**843.5**$_{\pm 45.7}$	**8500.8**$_{\pm 572.9}$	**9.956**$_{\pm 0.699}$
MLP	418.7$_{\pm 128.5}$	8150.7$_{\pm 1650.2}$	5.401$_{\pm 1.328}$	624.3$_{\pm 139.7}$	8167.3$_{\pm 720.5}$	7.931$_{\pm 1.247}$
+ RoDAC	**293.4**$_{\pm 115.2}$	**8172.5**$_{\pm 1724.8}$	**3.726**$_{\pm 0.962}$	**446.9**$_{\pm 121.6}$	**8185.6**$_{\pm 698.2}$	**5.497**$_{\pm 0.871}$

observe that RoDAC consistently improves model performance under both intra-season and inter-season experimental settings with different degrees of distribution shifts, suggesting that RoDAC is a generalizable data-centric framework. We provide additional results depicting #FP at various #Detect levels by varying the prediction threshold as shown in Figure 11, which demonstrates that RoDAC achieves the lowest #FP values compared to other ablation scenarios regardless of the #Detect values. We note that during one year of deployment in our anti-cheat system, RoDAC with XGBoost consistently maintained an average FDR below 2% while detecting about 8k users per day on average. In addition, we operate a ban appeal system that allows false positive users to contest their bans and request reinstatement as detailed in Sec. A.1.

5.3 Compatibility with Various Models

To show the compatibility of RoDAC with various models, we conduct experiments using XGBoost [15], LightGBM [44], CatBoost [63], Random Forests [10], and Multi-Layer Perceptron (MLP) [64] as backbone models. We present the results of model performance with and without RoDAC in Table 3. The results verify that applying RoDAC improves the performance of various models by improving the quality of both training and inference data. In addition, we can also observe that XGBoost achieves the best performance among all evaluated backbone models.

5.4 Computational Time Analysis

We analyze the computational time of RoDAC with XGBoost for each component, including model training and inference, as shown in Table 4. Since the sizes of training and inference data differ slightly between the two experimental settings (as described in Table 1), the computational time also varies accordingly. According to the results in Table 4, the computational overhead of the three sample selection methods applied to the training data (LNC, ODF (train), and WUS) is minimal, especially considering that training occurs only once per week. For inference, the overhead of ODF (infer) is negligible compared to the total inference time. We also observe that both training and inference on large-scale datasets can be completed within a few minutes by employing a simple yet effective data-centric framework with the tree-based model XGBoost, enabling the deployment of our efficient anti-cheat system.

Table 4: Computational time of RoDAC on the PUBG: BATTLEGROUNDS competitive league dataset.

Method	Season 35 - Season 35	Season 35 - Season 36
	Time	Time
LNC	3.6s	3.8s
ODF (train)	9.3s	9.6s
WUS	3.8s	4.2s
Training	56.2s	58.5s
ODF (infer)	10.5s	11.4s
Inference	366.8s	542.2s

6 Conclusion

We proposed RoDAC, a robust data-centric anti-cheat framework that effectively mitigates label noise, class imbalance, and distribution shifts in continuous tabular user-match logs. By introducing tailored sample selection methods for each of these challenges, RoDAC enhances the overall quality of both training and inference data. Through deployment on a large-scale real-world PUBG: BATTLEGROUNDS dataset, we demonstrated that RoDAC substantially improves detection performance and robustness while maintaining computational efficiency and compatibility with diverse backbone models. Based on these results, we expect our anti-cheat system to contribute to maintaining fairness in the competitive gaming environment. We believe that our simple yet effective framework enables seamless plug-and-play integration into a wide variety of real-world applications through its distribution-based adaptive design. As future work, we plan to extend our approach by exploring tabular foundation models, which have recently shown superior performance to traditional tree-based models on structured data. In addition, we aim to leverage vision-language foundation models to analyze replay videos of users for detecting subtle cheating behaviors beyond what tabular logs can reveal.

References

[1] Hashem Alayed, Fotos Frangoudes, and Clifford Neuman. 2013. Behavioral-based cheating detection in online first person shooters using machine learning techniques. In *CIG*. 1–8.

[2] Shin Ando and Chun-Yuan Huang. 2017. Deep Over-sampling Framework for Classifying Imbalanced Data. In *ECML/PKDD*, Vol. 10534. 770–785.

[3] Md Sakib Anwar, Chaoshun Zuo, Carter Yagemann, and Zhiqiang Lin. 2023. Extracting Threat Intelligence From Cheat Binaries For Anti-Cheating. In *RAID*. 17–31.

[4] Sercan Ö. Arik and Tomas Pfister. 2021. TabNet: Attentive Interpretable Tabular Learning. In *AAAI*. 6679–6687.

[5] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. 2019. Regularized Learning for Domain Adaptation under Label Shifts. In *ICLR*.

[6] Matthew Barthet, Maria Kaselimi, Kosmas Pinitas, Konstantinos Makantasis, Antonios Liapis, and Georgios N. Yannakakis. 2024. GameVibe: A Multimodal Affective Game Corpus. *CoRR* abs/2407.12787 (2024).

[7] Jeremy Blackburn, Nicolas Kourtellis, John Skvoretz, Matei Ripeanu, and Adriana Iamnitchi. 2014. Cheating in Online Games: A Social Network Perspective. *ACM Trans. Internet Techn.* 13, 3 (2014), 9:1–9:25.

[8] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2024. Deep Neural Networks and Tabular Data: A Survey. *IEEE Trans. Neural Networks Learn. Syst.* 35, 6 (2024), 7499–7519.

[9] Houssem Ben Braiek and Foutse Khomh. 2024. Machine Learning Robustness: A Primer. *CoRR* abs/2404.00897 (2024).

[10] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (2001), 5–32.

[11] Phillip J. Brooke, Richard F. Paige, John A. Clark, and Susan Stepney. 2004. Playing the game: cheating, loopholes, and virtual identity. *SIGCAS Comput. Soc.* 34, 2 (2004), 3.

[12] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106 (2018), 249–259.

[13] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. 2019. Domain Generalization by Solving Jigsaw Puzzles. In *CVPR*. 2229–2238.

[14] Laetitia Chapel, Dmitri Botvich, and David Malone. 2010. Probabilistic approaches to cheating detection in online games. In *CIG*. 195–201.

[15] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *SIGKDD*. 785–794.

[16] Wuxing Chen, Kaixiang Yang, Zhiwen Yu, Yifan Shi, and C. L. Philip Chen. 2024. A survey on imbalanced learning: latest research, applications and future directions. *Artif. Intell. Rev.* 57, 6 (2024), 137.

[17] Sam Collins, Alex Poulopoulos, Marius Muench, and Tom Chothia. 2024. AntiCheat: Attacks and the Effectiveness of Client-Side Defences. In *CCS*. 30–43.

[18] Qi Dong, Shaogang Gong, and Xiatian Zhu. 2019. Imbalanced Deep Learning by Minority Class Incremental Rectification. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 6 (2019), 1367–1381.

[19] Henry Been-Lirn Duh and Vivian Hsueh-hua Chen. 2009. Cheating Behaviors in Online Gaming. In *HCI*, Vol. 5621. 567–573.

[20] Benjamin Engelstätter and Michael R. Ward. 2022. Video games become more mainstream. *Entertain. Comput.* 42 (2022), 100494.

[21] Ge Fan, Chaoyun Zhang, Kai Wang, Yingjie Li, Junyang Chen, and Zenglin Xu. 2024. CUPID: Improving Battle Fairness and Position Satisfaction in Online MOBA Games with a Re-matchmaking System. *Proc. ACM Hum. Comput. Interact.* 8, CSCW2 (2024), 1–39.

[22] Benoît Frénay and Michel Verleysen. 2014. Classification in the Presence of Label Noise: A Survey. *IEEE Trans. Neural Networks Learn. Syst.* 25, 5 (2014), 845–869.

[23] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, 5 (2001), 1189 – 1232.

[24] Luca Galli, Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. 2011. A cheating detection framework for Unreal Tournament III: A machine learning approach. In *CIG*. 266–272.

[25] Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *ICML*, Vol. 37. 1180–1189.

[26] Aritra Ghosh, Naresh Manwani, and P. S. Sastry. 2017. On the Robustness of Decision Tree Learning Under Label Noise. In *PAKDD*, Vol. 10234. 685–697.

[27] Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

[28] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting Deep Learning Models for Tabular Data. In *NeurIPS*. 18932–18943.

[29] Frank Grimberg, Petra Maria Asprion, Bettina Schneider, Enkelejda Miho, Lmar Babrak, and Ali Habbabeh. 2021. The Real-World Data Challenges Radar: A Review on the Challenges and Risks regarding the Use of Real-World Data. *Digital Biomarkers* 5, 2 (2021), 148–157.

[30] Ishaan Gulrajani and David Lopez-Paz. 2021. In Search of Lost Domain Generalization. In *ICLR*.

[31] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*. 8536–8546.

[32] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *ICML*, Vol. 80. 1994–2003.

[33] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *ICLR*.

[34] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning Deep Representation for Imbalanced Classification. In *CVPR*. 5375–5384.

[35] Haiwen Huang, Zhihan Li, Lulu Wang, Sishuo Chen, Xinyu Zhou, and Bin Dong. 2021. Feature Space Singularity for Out-of-Distribution Detection. In *AAAI*, Vol. 2808.

[36] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. 2020. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *CoRR* abs/2012.06678 (2020).

[37] Simon Jenni and Paolo Favaro. 2018. Deep Bilevel Learning. In *ECCV*, Vol. 11214. 632–648.

[38] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *ICML*, Vol. 80. 2309–2318.

[39] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. 2020. Minimum Class Confusion for Versatile Domain Adaptation. In *ECCV*, Vol. 12366. 464–480.

[40] Justin M. Johnson and Taghi M. Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *J. Big Data* 6 (2019), 27.

[41] Aditya Jonnalagadda, Iuri Frosio, Seth Schneider, Morgan McGuire, and Joohwan Kim. 2021. Robust Vision-Based Cheat Detection in Competitive Gaming. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 1 (2021), 7:1–7:18.

[42] Ah Reum Kang, Seong Hoon Jeong, Aziz Mohaisen, and Huy Kang Kim. 2016. Multimodal Game Bot Detection using User Behavioral Characteristics. *CoRR* abs/1606.01426 (2016).

[43] Qi Kang, Lei Shi, MengChu Zhou, XueSong Wang, Qidi Wu, and Zhi Wei. 2018. A Distance-Based Weighted Undersampling Scheme for Support Vector Machines and its Application to Imbalanced Classification. *IEEE Trans. Neural Networks Learn. Syst.* 29, 9 (2018), 4152–4165.

[44] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *NIPS*. 3146–3154.

[45] Salman H. Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous Ahmed Sohel, and Roberto Togneri. 2018. Cost-Sensitive Learning of Deep Feature Representations From Imbalanced Data. *IEEE Trans. Neural Networks Learn. Syst.* 29, 8 (2018), 3573–3587.

[46] Ji Eun Kim and Milena Tsvetkova. 2021. Cheating in online gaming spreads through observation and victimization. *Netw. Sci.* 9, 4 (2021), 425–442.

[47] Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. *Prog. Artif. Intell.* 5, 4 (2016), 221–232.

[48] Peter Laurens, Richard F. Paige, Phillip J. Brooke, and Howard Chivers. 2007. A Novel Approach to the Detection of Cheating in Multiplayer Online Games. In *ICECCS*. 97–106.

[49] Jae-Gil Lee, Yuji Roh, Hwanjun Song, and Steven Euijong Whang. 2021. Machine Learning Robustness, Fairness, and their Convergence. In *SIGKDD*. 4046–4047.

[50] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *NeurIPS*. 7167–7177.

[51] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. 2018. Learning to Generalize: Meta-Learning for Domain Generalization. In *AAAI*. 3490–3497.

[52] Jian Liang, Ran He, and Tieniu Tan. 2025. A Comprehensive Survey on Test-Time Adaptation Under Distribution Shifts. *Int. J. Comput. Vis.* 133, 1 (2025), 31–64.

[53] Jian Liang, Dapeng Hu, and Jiashi Feng. 2020. Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation. In *ICML*, Vol. 119. 6028–6039.

[54] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. In *ICCV*. 2999–3007.

[55] Evan Zheran Liu, Behzad Haghgoo, Annie S. Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. 2021. Just Train Twice: Improving Group Robustness without Training Group Information. In *ICML*, Vol. 139. 6781–6792.

[56] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM*. 413–422.

[57] Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A Soft-label Method for Noise-tolerant Distantly Supervised Relation Extraction. In *EMNLP*. 1790–1795.

[58] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NIPS*. 4765–4774.

[59] Shannon McCreary and K. Claffy. 2000. Trends in wide area IP traffic patterns - A view from Ames Internet Exchange. *Internet Traffic Measurement and Modeling* (2000).

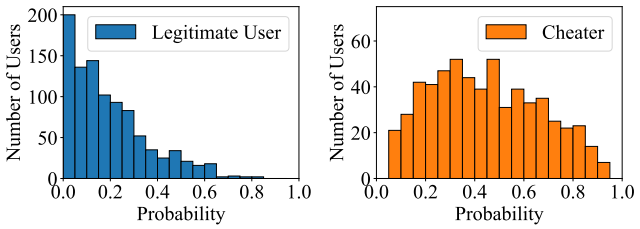
[60] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. 2021. Confident Learning: Estimating Uncertainty in Dataset Labels. *J. Artif. Intell. Res.* 70 (2021), 1373–1411.
[61] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach. In *CVPR*. 2233–2241.
[62] Samira Pouyanfar, Yudong Tao, Anup Mohan, Haiman Tian, Ahmed S. Kaseb, Kent Gauen, Ryan Dailey, Sarah Aghajanzadeh, Yung-Hsiang Lu, Shu-Ching Chen, and Mei-Ling Shyu. 2018. Dynamic Sampling in Convolutional Neural Networks for Imbalanced Data Classification. In *MIPR*. 112–117.
[63] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *NeurIPS*. 6639–6649.
[64] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533–536.
[65] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Inf. Fusion* 81 (2022), 84–90.
[66] Borut Sluban, Dragan Gamberger, and Nada Lavrac. 2014. Ensemble-based noise detection: noise ranking and visual performance evaluation. *Data Min. Knowl. Discov.* 28, 2 (2014), 265–303.
[67] Michael R. Smith, Tony R. Martinez, and Christophe G. Giraud-Carrier. 2014. An instance level analysis of data complexity. *Mach. Learn.* 95, 2 (2014), 225–256.
[68] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2023. Learning From Noisy Labels With Deep Neural Networks: A Survey. *IEEE Trans. Neural Networks Learn. Syst.* 34, 11 (2023), 8135–8153.
[69] Ruan Spijkerman and Elizabeth Marie Ehlers. 2020. Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools. In *CIIS*. 87–92.
[70] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In *ICML*, Vol. 119. 9229–9248.
[71] Engkarat Techapanurak, Masanori Suganuma, and Takayuki Okatani. 2020. Hyperparameter-Free Out-of-Distribution Detection Using Cosine Similarity. In *ACCV*, Vol. 12625. 53–69.
[72] HaiYun Tian, Phillip J. Brooke, and Anne-Gwenn Bosser. 2012. Behaviour-Based Cheat Detection in Multiplayer Games with Event-B. In *IFM*, Vol. 7321. 206–220.
[73] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605.
[74] Brendan van Rooyen, Aditya Krishna Menon, and Robert C. Williamson. 2015. Learning with Symmetric Label Noise: The Importance of Being Unhinged. In *NIPS*. 10–18.
[75] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. 2021. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *ICLR*.
[76] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J. Kennedy. 2016. Training deep neural networks on imbalanced data sets. In *IJCNN*. 4368–4374.
[77] Martin Willman. 2020. Machine Learning to identify cheaters in online games. , 44 pages.
[78] Jeff Jianxin Yan. 2003. Security Design in Online Games. In *ACSAC*. 286–295.
[79] Jeff Jianxin Yan and Hyun-Jin Choi. 2002. Security issues in online games. *Electron. Libr.* 20, 2 (2002), 125–133.
[80] Jeff Jianxin Yan and Brian Randell. 2005. A systematic classification of cheating in online games. In *NETGAMES*. 1–9.
[81] Siu Fung Yeung, John C. S. Lui, Jiangchuan Liu, and Jeff Yan. 2006. Detecting cheaters for multiplayer games: theory, design and implementation[1]. In *CCNC*. 1178–1182.
[82] Yuchen Zeng, Tuan Dinh, Wonjun Kang, and Andreas C. Mueller. 2025. TabFlex: Scaling Tabular Learning to Millions with Linear Attention. *CoRR* abs/2506.05584 (2025).
[83] Longxiang Zhang and Wenping Wang. 2023. Multi-Modal Machine Learning for Assessing Gaming Skills in Online Streaming: A Case Study with CS: GO. *CoRR* abs/2307.12236 (2023).
[84] Zhilu Zhang and Mert R. Sabuncu. 2018. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *NeurIPS*. 8792–8802.
[85] Shiwei Zhao, Jiaheng Qi, Zhipeng Hu, Han Yan, Runze Wu, Xudong Shen, Tangjie Lv, and Changjie Fan. 2024. VESPA: A General System for Vision-Based Extrasensory Perception Anticheating in Online FPS Games. *IEEE Trans. Games* 16, 3 (2024), 611–620.
[86] Xiang Zuo, Clayton Gandy, John Skovertz, and Adriana Iamnitchi. 2016. Bad Apples Spoil the Fun: Quantifying Cheating in Online Gaming. In *ICWSM*. 496–506.

A Appendix

A.1 Ban Appeal System

Our anti-cheat system imposes a temporary ban on users classified as cheaters by our model, preventing them from playing the game for a fixed period. However, since our model may mistakenly classify legitimate users as cheaters and impose a ban, we operate a ban appeal system that allows falsely banned users to contest their bans and request reinstatement. According to our internal model interpretation analysis using SHAP (SHapley Additive exPlanations) [58], we observe that our model relies, to some extent, on historical features when detecting cheaters. Since our goal is to detect whether a user actually employed ESP cheats in a match, we suggest reinstating users who were banned primarily due to their historical features.

Based on this observation, we replace the historical features of cheaters with the average historical features of legitimate users and then compare the predicted probability distributions for legitimate users and cheaters using our model as shown in Figure 12. By replacing the historical features of cheaters with those of legitimate users, their predicted probabilities shift toward lower values, while a subset of cases still exhibits high predicted probabilities. Based on the distributional differences between legitimate users and cheaters with replaced historical features, we set a new threshold to distinguish between cheaters who should be reinstated and those whose bans should be upheld. As a result, on average, about 20% of users who appealed their bans are reinstated, while the remaining 80% have their bans upheld. We expect our ban-appeal system to mitigate the negative effects of erroneous bans on users.



(a) Legitimate users. **(b) Cheaters.**

Figure 12: Comparison of predicted probability distributions for legitimate users and cheaters.