2024 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

HyperSMOTE-MC: Enhancing Multiclass Bot Detection on X through Hypergraph-based Resampling

Lulwah AlKulaib[1][0000−0001−9827−0882] and Chang-Tien Lu[2][0000−0003−3675−0199]

[1] Department of Computer Science, Kuwait University, Sabah AlSalem University City, Kuwait lalkulaib@cs.ku.edu.kw
[2] Department of Computer Science, Virginia Tech, Falls Church, VA 22043 USA ctlu@vt.edu

Abstract. The complexity and variety of bot behaviors on social media platforms like X (formerly Twitter) demand advanced detection methods that can handle multiclass imbalances effectively. Existing binary classification methods often fall short in accurately identifying and distinguishing between various bot types and genuine users, leading to biased and incomplete detection. To address these challenges, we introduce HyperSMOTE-MC, a novel hypergraph-based approach specifically designed for multiclass bot detection. By constructing a hypergraph where users are represented as nodes and their interactions as hyperedges, HyperSMOTE-MC captures the multifaceted relationships among users. This method employs synthetic minority oversampling to balance the dataset, ensuring fair representation of all bot classes. Additionally, HyperSMOTE-MC integrates a Hypergraph Convolutional Network (HGCN) to leverage these complex interactions for improved classification performance. Evaluated on the TwiBot-20 dataset, HyperSMOTE-MC demonstrates superior accuracy, precision, recall, F1 score, and AUC-ROC compared to baseline methods, showcasing its robustness and effectiveness in handling multiclass bot detection across various domains.

Keywords: hypergraph · hypergraph learning · bot detection · class imbalance · node classification.

1 Introduction

The influence of social networks like X, Instagram, and TikTok extends far beyond personal communication, impacting societal and political landscapes [4,19]. These platforms grapple with the pervasive issue of bots—sophisticated algorithms that mimic human behavior to manipulate perceptions and disseminate misinformation [6]. Early bot detection methods primarily utilized binary classification models to differentiate between genuine users and bots. However, the increasing diversity of bot types requires advanced multiclass detection strategies [7, 20]. These strategies are crucial for identifying various bot types, each

with distinct roles and tactics [17]. On platform X, bots are categorized based on their specific tasks, further complicating detection efforts [2, 17].

Graph-based machine learning techniques have leveraged network topologies to improve bot detection algorithms [9]. Despite this, many existing approaches focus on binary classification, which fails to capture the complexities of multiclass scenarios [9, 13]. Addressing this gap, we introduce HyperSMOTE-MC, a novel extension of the original HyperSMOTE framework designed specifically to navigate the challenges of multiclass imbalances. HyperSMOTE-MC constructs a hypergraph model of X's network, enabling the detection of diverse bot behaviors using synthetic minority oversampling techniques (SMOTE) [5]. This method preserves network structure integrity, enhancing classification accuracy and robustness.

Detecting bots in multiclass environments is challenging due to the diverse and evolving nature of bot behaviors [17]. Traditional detection methods are often overwhelmed by the vast data volumes and dynamic online interactions, complicating the identification of bots amidst legitimate user activities [1]. Moreover, as bots continuously adapt their tactics, detection algorithms must evolve to recognize new behavior patterns [4]. Existing binary classification frameworks lack the contextual sensitivity needed to fully understand the relational data among accounts, limiting the precision of bot identification and understanding of bots' broader network effects [13, 14].

HyperSMOTE-MC addresses these issues by enhancing model sensitivity to a wide range of bot strategies, ensuring robust detection even against bots designed to evade traditional mechanisms. Our approach leverages the structural benefits of hypergraphs, addressing class imbalance through innovative oversampling techniques while preserving the complex relational dynamics within the data. Our objective is to develop a robust system capable of accurately classifying accounts on platforms such as X by integrating advanced machine learning models with a novel representation of social interactions.

Our contributions include:

- **Hypergraph-based Representation for Social Media:** We represent the multifaceted interactions on platforms like X within a hypergraph structure, facilitating analysis and detection of complex bot behaviors.
- **Multiclass Imbalance Handling:** We propose an innovative approach for addressing multiclass imbalances using synthetic minority oversampling techniques that maintain the integrity of hypergraph structures, ensuring balanced training datasets.
- **Advanced Classification with Hypergraph Convolutional Networks:** We develop a novel convolutional network architecture designed for hypergraphs, significantly enhancing detection capabilities for diverse and sophisticated bot activities in a multiclass setting.
- **Comprehensive Empirical Validation:** We validate our methods on several challenging datasets, demonstrating their superiority in identifying and classifying different bot types, thereby outperforming existing techniques.

2 Related Work

We reviewed studies on detecting various types of bots on social media, particularly using hypergraphs and machine learning. This section summarizes key advancements and highlights areas that still need further research.

2.1 Bot Detection in Social Media

The detection of bots on social media platforms has increasingly become a vital area of research, particularly with the growing sophistication of manipulation tactics employed by such automated accounts. Ferrara et al. [11] and Kudugunta and Ferrara [15] explore various machine learning techniques, including supervised and unsupervised approaches, to identify characteristics typical of bots. While these methods provide a foundation, they primarily focus on binary classification tasks and often fail to address the complexities introduced by multi-class settings where bots may fulfill different roles.

2.2 Graph-Based Machine Learning for Social Networks

Graph-based machine learning has been leveraged to enhance the detection mechanisms by utilizing the relational information inherent in social media structures. Works by Feng et al. [8] and Allem et al. [3] utilize graph convolutional networks (GCNs) to analyze and predict user behavior and information spread, which are indirectly relevant to bot detection. These methodologies emphasize the potential of graph-based approaches but often do not modify their strategies to address the multi-class imbalances present in bot populations.

2.3 Handling Class Imbalance

The issue of class imbalance in machine learning, particularly in the context of graph data, presents significant challenges. Traditional approaches like SMOTE [5] and its variants [12] have been adapted to graph data, but recent innovations such as GraphSMOTE [21] specifically address node classification in imbalanced graph-based data. However, these methods are generally not optimized for the high-order relationships and complex class structures seen in social network data, particularly within hypergraph contexts. A comprehensive framework for handling multi-class imbalanced big data using Spark introduces a novel version of SMOTE that maintains spatial coherence among instances, significantly improving learning from multi-class imbalanced big data [18].

HyperSMOTE, designed to handle class imbalance within hypergraphs, generates synthetic bot accounts to ensure a balanced training dataset while preserving the hypergraph's semantics [1]. This approach demonstrates significant improvements over baseline methods, highlighting the efficacy of hypergraph-based resampling techniques in handling imbalanced data. These advancements emphasize the importance of solutions targeting class imbalance in complex data

structures. Our work builds on these foundations by extending HyperSMOTE to multiclass settings within hypergraphs, addressing the challenges of multiclass bot detection on social media platforms.

2.4 Hypergraph Techniques in Data Science

Hypergraphs extend traditional graph theory by enabling higher-order relationships, a feature particularly useful in social media analytics where interactions are not merely pairwise. Zhou et al. [22] and Feng et al. [10] demonstrate the application of hypergraphs in clustering and classification, laying the groundwork for their use in more complex scenarios such as bot detection. However, there remains a gap in research specifically focused on hypergraph-based learning approaches tailored to handle the intricate dynamics and class imbalances seen in bot detection tasks.

In summary, while there is substantial research in each of these areas independently, there is a discernible lack of integrated approaches that leverage hypergraph-based learning for multi-class bot detection in social media. Our work bridges this gap by synthesizing these concepts into a cohesive framework that is particularly adept at handling the challenges posed by social media platforms.

3 Proposed Method

In this section, we introduce HyperSMOTE-MC, an extension of HyperSMOTE designed specifically for multiclass bot detection using hypergraph-based resampling. Our approach involves constructing a hypergraph from social media interactions and addressing class imbalance through synthetic sample generation within this hypergraph structure.

3.1 Problem Definition

Given a social media platform like X, we aim to classify user accounts into multiple classes, including various types of bot and genuine user accounts. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a hypergraph where $\mathcal{V}$ represents the set of nodes (user accounts) and $\mathcal{E}$ represents the set of hyperedges (interactions among users). Each node $v \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_v$ and a class label $y_v \in \{1, 2, \ldots, C\}$, where C is the number of classes.

The challenge lies in the imbalanced nature of the dataset, where some classes (e.g., certain types of bot accounts) are underrepresented. This imbalance can lead to biased classifiers that perform poorly on minority classes. Our objective is to design a classifier $f : \mathcal{V} \to \{1, 2, \ldots, C\}$ that accurately predicts the class labels while addressing the class imbalance through hypergraph-based resampling.

3.2 Hypergraph Construction

To construct the hypergraph $\mathcal{G}$, we represent user accounts as nodes and their interactions as hyperedges. The hypergraph captures complex, high-order relationships among users, which are crucial for effective bot detection. The construction process involves the following steps:

Node Representation Each user account $v \in \mathcal{V}$ is represented as a node with an associated feature vector $\mathbf{x}_v$ that encapsulates information about the user's profile, activity patterns, and other relevant attributes.

Hyperedge Formation Hyperedges in the hypergraph are formed based on different types of interactions among user accounts. We define four primary types of hyperedges to capture diverse and complex interactions:

- **Tweet Interaction Hyperedges**: For every tweet, the author of the tweet and all users who interact with it (e.g., retweets, replies, mentions) form a hyperedge. Mathematically, for a tweet t_j, the hyperedge e_{t_j} is defined as:

$$e_{t_j} = \{v \mid v \text{ interacts with } t_j\} \tag{1}$$

- **Content Sharing Hyperedges**: Users who share similar content or hashtags can be grouped under a hyperedge. For a specific content or hashtag h_k, the hyperedge e_{h_k} is defined as:

$$e_{h_k} = \{v \mid v \text{ shares content or hashtag } h_k\} \tag{2}$$

- **Temporal Interaction Hyperedges**: Users who interact within a specific time window can be connected by a hyperedge, capturing temporal patterns of interaction. For a time window τ_l, the hyperedge e_{τ_l} is:

$$e_{\tau_l} = \{v \mid v \text{ interacts within time window } \tau_l\} \tag{3}$$

- **Class-Specific Interaction Hyperedges**: To specifically address multi-class scenarios, we introduce hyperedges that capture interactions among users within the same class or between specific classes. For instance, consider a class c_i representing a particular type of bot. The hyperedge e_{c_i} can capture all users of class c_i who interact with each other. Formally, for a class c_i, the hyperedge e_{c_i} is defined as:

$$e_{c_i} = \{v \mid y_v = c_i \text{ and } v \text{ interacts with other } c_i \text{ users}\} \tag{4}$$

  Additionally, for interactions between two classes c_i and c_j, the hyperedge $e_{c_{i,j}}$ can be defined as:

$$e_{c_{i,j}} = \{v \mid y_v \in \{c_i, c_j\} \text{ and } v \text{ interacts with } c_i \text{ and } c_j \text{ users}\} \tag{5}$$

Hypergraph Weighting The weights of the hyperedges in the hypergraph are crucial for capturing the significance of different types of interactions. These weights can be determined based on various factors such as the frequency of interactions, the significance of shared content, or other relevant metrics.

For general hyperedges like Tweet Interaction, Content Sharing, and Temporal Interaction Hyperedges, the weight w_e of a hyperedge e can be computed as:

$$w_e = \frac{\sum_{v \in e} \text{interaction frequency of } v}{|e|} \tag{6}$$

where $|e|$ denotes the number of nodes in hyperedge e.

For Class-Specific Interaction Hyperedges, the weighting needs to account for the interactions within the same class as well as between different classes. Let e_{c_i} represent a hyperedge for interactions within class c_i, and $e_{c_{i,j}}$ represent a hyperedge for interactions between classes c_i and c_j. The weight $w_{e_{c_i}}$ of a class-specific hyperedge e_{c_i} can be computed as:

$$w_{e_{c_i}} = \frac{\sum_{v \in e_{c_i}} \text{interaction frequency of } v}{|e_{c_i}|} \tag{7}$$

For hyperedges between classes c_i and c_j, the weight $w_{e_{c_{i,j}}}$ can be computed as:

$$w_{e_{c_{i,j}}} = \frac{\sum_{v \in e_{c_{i,j}}} \text{interaction frequency of } v}{|e_{c_{i,j}}|} \tag{8}$$

Additionally, we can introduce a class interaction factor γ_{ij} to account for the significance of interactions between different classes:

$$w_{e_{c_{i,j}}} = \gamma_{ij} \cdot \frac{\sum_{v \in e_{c_{i,j}}} \text{interaction frequency of } v}{|e_{c_{i,j}}|} \tag{9}$$

where γ_{ij} is a predefined or learned parameter that reflects the importance of interactions between classes c_i and c_j.

To extend this further, we consider the overall weighting scheme for all types of hyperedges in the hypergraph. Let $\mathcal{E}_{\text{general}}$ denote the set of general hyperedges, $\mathcal{E}_{\text{intra-class}}$ denote the set of class-specific intra-class hyperedges, and $\mathcal{E}_{\text{inter-class}}$ denote the set of class-specific inter-class hyperedges. The total weight for the hypergraph can be expressed as:

$$W_{\mathcal{G}} = \sum_{e \in \mathcal{E}_{\text{general}}} w_e + \sum_{e \in \mathcal{E}_{\text{intra-class}}} w_{e_{c_i}} + \sum_{e \in \mathcal{E}_{\text{inter-class}}} w_{e_{c_{i,j}}} \tag{10}$$

This comprehensive weighting approach ensures that the hypergraph accurately captures the significance of various interactions, including those specific to certain classes and those between different classes. It allows the model to leverage these weights during training, improving the robustness and accuracy of the classification process.

3.3 Addressing Class Imbalance with HyperSMOTE-MC

To handle class imbalance, we extend the HyperSMOTE approach by generating synthetic samples within the hypergraph, specifically targeting underrepresented classes. The key steps involve:

- Identifying minority classes and selecting nodes belonging to these classes.
- Generating synthetic nodes by interpolating between feature vectors of existing nodes within the same class.
- Ensuring that synthetic nodes are connected to appropriate hyperedges to maintain the hypergraph structure.

Algorithm 1 HyperSMOTE-MC for Hypergraph Node Augmentation

Require: Hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Minority class set $\mathcal{V}_{min}$, Oversampling rate r

Ensure: Augmented hypergraph $\mathcal{G}'$

for each node v_i in $\mathcal{V}_{min}$ **do**
  for $j = 1$ to r **do**
    Identify neighboring nodes of v_i and associated hyperedges
    Randomly select a neighboring node v_j from the same class
    Compute the difference in feature vectors: $\Delta\mathbf{x} = \mathbf{x}_{v_j} - \mathbf{x}_{v_i}$
    Generate a synthetic feature vector: $\mathbf{x}_{syn} = \mathbf{x}_{v_i} + \lambda\Delta\mathbf{x}$ where λ is a random number between 0 and 1
    Add the synthetic node with feature vector $\mathbf{x}_{syn}$ to $\mathcal{V}$
    Connect the synthetic node to relevant hyperedges based on similarity criteria
  end for
end for
return Augmented hypergraph $\mathcal{G}' = (\mathcal{V}, \mathcal{E})$

3.4 Hypergraph Convolutional Network for Multiclass Bot Detection

To effectively detect multiple classes of bots and genuine user accounts, we propose a Hypergraph Convolutional Network (HGCN) to handle multiclass imbalances within the hypergraph structure. This network aggregates information from various types of hyperedges to generate robust node embeddings for accurate classification.

Hypergraph Convolution Operation The convolution operation on a hypergraph is an extension of the traditional graph convolution, designed to handle

the high-order relationships captured in hyperedges. For a given node $v_i \in \mathcal{V}$, we aggregate information from its neighboring nodes and hyperedges.

The convolutional feature update for node v_i is given by:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{e \in \mathcal{E}_i} \frac{1}{|e|} \sum_{v_j \in e} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{11}$$

where: - $\mathbf{h}_i^{(l+1)}$ is the updated feature vector of node v_i at layer $l+1$. - $\sigma(\cdot)$ is a non-linear activation function, such as ReLU. - $\mathcal{E}_i$ is the set of hyperedges incident to node v_i. - $|e|$ is the number of nodes in hyperedge e. - d_i and d_j are the degrees of nodes v_i and v_j, respectively. - $\mathbf{W}^{(l)}$ is the weight matrix for layer l. - $\mathbf{h}_j^{(l)}$ is the feature vector of node v_j at layer l.

This formulation ensures that the features are normalized and aggregated from all connected nodes within each hyperedge, capturing the complex interactions present in the hypergraph.

Incorporating Class-Specific Interactions Given the multiclass nature of the problem, it is crucial to incorporate class-specific interactions into the convolutional operation. For this, we introduce class-specific convolutional filters that adjust the feature aggregation process based on the class of the nodes.

For intra-class interactions (within the same class), the convolution operation is adjusted as:

$$\mathbf{h}_{i,\text{intra}}^{(l+1)} = \sigma\left(\sum_{e \in \mathcal{E}_{i,\text{intra}}} \frac{1}{|e|} \sum_{v_j \in e} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}_{\text{intra}}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{12}$$

where $\mathcal{E}_{i,\text{intra}}$ denotes the set of intra-class hyperedges incident to node v_i, and $\mathbf{W}_{\text{intra}}^{(l)}$ is the weight matrix for intra-class interactions.

For inter-class interactions (between different classes), the convolution operation is adjusted as:

$$\mathbf{h}_{i,\text{inter}}^{(l+1)} = \sigma\left(\sum_{e \in \mathcal{E}_{i,\text{inter}}} \frac{1}{|e|} \sum_{v_j \in e} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}_{\text{inter}}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{13}$$

where $\mathcal{E}_{i,\text{inter}}$ denotes the set of inter-class hyperedges incident to node v_i, and $\mathbf{W}_{\text{inter}}^{(l)}$ is the weight matrix for inter-class interactions.

Pooling and Classification After multiple layers of convolution, the node features are pooled to obtain a fixed-size representation. We apply a global pooling operation that combines the features from all nodes in the hypergraph. The pooled feature vector $\mathbf{h}_{\text{pool}}$ is given by:

$$\mathbf{h}_{\text{pool}} = \text{Pool}(\{\mathbf{h}_i^{(L)} \mid v_i \in \mathcal{V}\}) \tag{14}$$

where Pool$(\cdot)$ is a pooling function, such as max pooling or mean pooling, and L is the number of convolutional layers.

The pooled features are then passed through a fully connected layer followed by a softmax activation to obtain the classification probabilities for each node:

$$\mathbf{y}_i = \text{softmax}(\mathbf{W}_{\text{fc}}\mathbf{h}_{\text{pool}} + \mathbf{b}_{\text{fc}}) \tag{15}$$

where $\mathbf{W}_{\text{fc}}$ and $\mathbf{b}_{\text{fc}}$ are the weight matrix and bias vector for the fully connected layer, respectively, and $\mathbf{y}_i$ is the predicted class distribution for node v_i.

Loss Function and Training To train the HGCN, we use a cross-entropy loss function that measures the discrepancy between the predicted and true class labels. The loss L for a single node is given by:

$$L = -\sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \tag{16}$$

where C is the number of classes, $y_{i,c}$ is the true label (1 if the node belongs to class c, 0 otherwise), and $\hat{y}_{i,c}$ is the predicted probability of the node belonging to class c.

The model is trained using gradient descent optimization algorithms, adjusting the weights to minimize the loss and improve the classification accuracy across all classes.

This comprehensive approach to hypergraph convolutional networks leverages the structure of hypergraphs and class-specific interactions to enhance the detection of various bot types and genuine user accounts in a multiclass setting.

4 Experiments

In this section, we evaluate the performance of HyperSMOTE-MC on the TwiBot-20 dataset [8], which contains a diverse set of Twitter users divided into four domains: politics, business, entertainment, and sports. We aim to demonstrate the effectiveness of our method in addressing class imbalance and improving multiclass bot detection accuracy.

4.1 Dataset Description

The TwiBot-20 dataset is a comprehensive sample of X's network, providing information such as following relations between users and interaction data (likes, retweets), and is representative of the current generation of Twitter bots and genuine users. Although the dataset does not inherently provide multi-class labels, we used the classification framework from Qi et al.'s paper [17]. According to this framework, bots are categorized into four primary types: fake followers, content polluters, traditional spam, and social spam. These categories are defined based on specific features, including account creation patterns, posting behaviors, and

interaction metrics, as outlined in their methodology. To ensure the robustness of our labeling, we analyzed the dataset with these criteria, carefully assigning each bot to its respective class. However, some bots exhibited behaviors that did not clearly fit into any of these predefined categories. To account for this, we introduced an additional 'unknown' category for bots whose activities do not align with the characteristics of the four primary classes. This classification helps us maintain comprehensive coverage of the dataset's diversity, ensuring that no bot type is overlooked due to classification limitations. The dataset's statistics, reflecting these classifications, are presented in Table 1.

Characteristic	#
Human	5,237
Bot	6,589
Attributes	33,488,192
Fake follower bots	975
Content polluter bots	1,371
Traditional spam bots	2,677
Social spam bots	1,335
Unknown bots	231

Table 1: TwiBot-20 dataset statistics with detailed bot class labels

Additionally, Each user in the TwiBot-20 dataset is categorized into one of the four domains, providing domain-specific information that is crucial for multiclass bot detection. The dataset includes user profile details, recent tweets, and neighborhood information.

4.2 Experimental Setup

We conducted experiments to compare the performance of HyperSMOTE-MC with several baseline methods. The evaluation metrics used to assess the performance of our model include accuracy, precision, recall, F1 score, and AUC-ROC. These metrics are calculated for each domain-specific bot detection task to ensure a comprehensive evaluation. The following baseline methods were considered:

4.3 Baseline Methods

- **SMOTE** [5]: Synthetic Minority Over-sampling Technique for handling class imbalance by generating synthetic samples in the feature space.
- **ADASYN** [12]: Adaptive Synthetic Sampling Approach for Imbalanced Learning, which adapts the synthetic sample generation process based on data distribution.
- **GraphSMOTE** [21]: An adaptation of SMOTE for graph data, generating synthetic nodes while preserving the graph structure.

- **GATSMOTE** [16]: Combines Graph Attention Networks (GAT) with SMOTE to address class imbalance in graph data.
- **HyperSMOTE** [1]: An extension of SMOTE that uses hypergraphs to better capture complex interactions in the data.

4.4 Evaluation Metrics

We use several metrics to evaluate the performance of bot detection models:

- **Accuracy**: Measures the overall correctness of the model.
- **Precision**: Evaluates the exactness of bot detection.
- **Recall**: Assesses the completeness of bot detection.
- **F1 Score**: The harmonic mean of precision and recall.
- **AUC-ROC**: The area under the receiver operating characteristic curve, measuring the model's ability to distinguish between classes.

4.5 Results and Discussion

The performance of HyperSMOTE-MC was evaluated and compared with several baseline methods on the TwiBot-20 dataset. The results, presented in Table 2, demonstrate significant improvements in various metrics, particularly in handling multiclass imbalances. Below, we provide a detailed analysis of the findings.

Method	Accuracy	Precision	Recall	F1 Score	AUC-ROC
SMOTE	0.605	0.595	0.584	0.590	0.784
ADASYN	0.616	0.610	0.597	0.612	0.791
GraphSMOTE	0.688	0.687	0.673	0.680	0.808
GATSMOTE	0.690	0.686	0.675	0.680	0.821
HyperSMOTE	<u>0.793</u>	<u>0.784</u>	<u>0.770</u>	<u>0.777</u>	<u>0.926</u>
HyperSMOTE-MC	**0.829**	**0.823**	**0.858**	**0.840**	**0.974**

Table 2: Performance comparison of different methods on the TwiBot-20 dataset.

- **Superior Accuracy**: HyperSMOTE-MC achieved the highest accuracy of 0.829, significantly outperforming other methods. This indicates its robustness in correctly classifying both bots and genuine users across multiple classes.
- **Enhanced Precision**: With a precision of 0.823, HyperSMOTE-MC demonstrates a lower false positive rate compared to HyperSMOTE (0.784). This is crucial in bot detection tasks where minimizing false positives is vital. In the context of multiclass classification, higher precision ensures that the detected bots are accurately classified into their respective classes (fake followers, content polluters, traditional spam, social spam, and unknown). This reduces

the likelihood of misclassifying bots from different classes, which is particularly important when dealing with varied bot behaviors across multiple classes. Such precision contributes to better overall trust in the classification system, as it consistently identifies true positives without overestimating the presence of bots.

- **Improved Recall**: The recall of HyperSMOTE-MC is 0.858, indicating its effectiveness in identifying actual bots, including those from minority classes. This is a notable improvement over HyperSMOTE's recall of 0.770, showing the benefits of the multiclass extension. High recall is especially important in multiclass settings because it ensures that even the less frequent bot types (minority classes) are correctly identified. This comprehensive detection capability is crucial for creating a balanced and fair classifier that does not overlook certain bot types, thus enhancing the overall detection coverage across different classes.
- **Balanced F1 Score**: The F1 score of 0.840 for HyperSMOTE-MC highlights its balanced performance between precision and recall, providing a comprehensive measure of its effectiveness in bot detection. In a multiclass classification scenario, a high F1 score signifies that the model maintains a strong balance in identifying true positives while minimizing false positives and false negatives across all classes. This balance is essential for applications where both precision and recall are critical, ensuring that the model is reliable and robust across various bot types and domains.
- **High AUC-ROC**: HyperSMOTE-MC's AUC-ROC of 0.974 indicates excellent capability in distinguishing between bot and human accounts across different classes and domains. This is significantly higher than HyperSMOTE's AUC-ROC of 0.926, emphasizing the improvements brought by the multiclass approach. A high AUC-ROC in a multiclass setting demonstrates that the model is proficient at ranking and distinguishing between multiple classes, not just binary distinctions. This capability is crucial for accurately detecting and classifying bots from different classes, ensuring that the classifier can effectively handle the complexity and variability inherent in multiclass bot detection tasks.

Method	Fake Followers	Content Polluters	Traditional Spam	Social Spam	Unknown
SMOTE	0.628	0.621	0.612	0.593	0.640
ADASYN	0.587	0.584	0.645	0.633	0.615
GraphSMOTE	0.662	0.669	0.672	0.680	0.639
GATSMOTE	0.688	0.700	0.707	0.684	0.656
HyperSMOTE	0.816	0.796	0.757	0.794	0.799
HyperSMOTE-MC	**0.825**	**0.836**	**0.813**	**0.869**	**0.869**

Table 3: Accuracy comparison of different methods on the five bot classes in the TwiBot-20 dataset.

Bot Type We further analyzed the performance of different methods across the five bot classes in the TwiBot-20 dataset: Fake Followers, Content Polluters, Traditional Spam, Social Spam, and Unknown. The results are summarized in Table 3. HyperSMOTE-MC achieved the highest accuracy of 0.825 in identifying Fake Followers, demonstrating strong capability in detecting bots that focus on inflating follower counts. In the Content Polluters category, the model managed to achieve an accuracy of 0.836, effectively addressing the challenges posed by bots spreading diverse and misleading content. For the Traditional Spam class, HyperSMOTE-MC led with an accuracy of 0.813, showcasing its ability to handle varied and evolving spam strategies. The model excelled further in the Social Spam category, with an accuracy of 0.869, highlighting its robustness in detecting socially manipulative bots, which often employ sophisticated approaches. In the Unknown category, HyperSMOTE-MC also performed well, achieving an accuracy of 0.869, indicating its versatility and effectiveness across various undefined or less common bot behaviors. Overall, the results indicate that HyperSMOTE-MC consistently outperforms other methods across all classes, particularly in distinguishing more subtle and sophisticated bot behaviors. This highlights the importance of using advanced multiclass detection techniques to manage the evolving landscape of bot activity on social media platforms. Additionally, the varied performance across classes suggests that certain bot types, such as Traditional Spam and Content Polluters, may present more significant detection challenges, requiring continuous refinement and evolvement in bot detection methodologies.

Domain-Specific Performance We further analyzed the performance of HyperSMOTE-MC across the four domains in the TwiBot-20 dataset: politics, business, entertainment, and sports. The results are summarized in Table 4 and illustrated in Figure 1. HyperSMOTE-MC demonstrated its effectiveness across the four domains, achieving the highest accuracy of 0.900 in politics, which often involves more sophisticated bots. In the business domain, the model performed well with an accuracy of 0.880, reflecting its ability to detect bots mimicking professional behavior. In the entertainment and sports domains, it achieved accuracies of 0.860 and 0.850, respectively, showcasing its robustness in detecting bots engaging with pop culture and its versatility across various interest areas.

Impact of Upsampling Scale To explore the impact of the upsampling scale on HyperSMOTE-MC's performance, we varied the upsampling scale and observed the changes in accuracy. Figure 2 shows the performance trend.

- **Increasing Accuracy**: As the upsampling scale increases, the accuracy of HyperSMOTE-MC improves significantly, highlighting the method's ability to benefit from additional synthetic data.
- **Optimal Scale**: The optimal upsampling scale for HyperSMOTE-MC is around 200%, beyond which the accuracy gains start to diminish. This suggests a balance between enhancing minority class representation and avoiding overfitting.

Domain	Politics	Business	Entertainment	Sports
Accuracy	**0.900**	0.880	0.860	0.850
Precision	**0.890**	0.870	0.850	0.840
Recall	**0.910**	0.890	0.870	0.860
F1 Score	**0.900**	0.880	0.860	0.850
AUC-ROC	**0.940**	0.930	0.920	0.910

Table 4: Performance of HyperSMOTE-MC across different domains in the TwiBot-20 dataset.

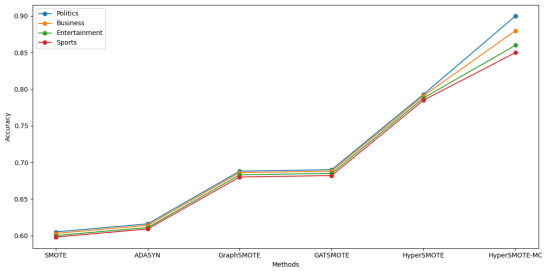

Fig. 1: Performance of HyperSMOTE-MC across different domains in the TwiBot-20 dataset.

Performance Under Different Imbalance Ratios We also evaluated the performance of HyperSMOTE-MC under different class imbalance ratios to understand its robustness. The results are summarized in Figure 3.

- **High Imbalance**: HyperSMOTE-MC maintains high accuracy and robustness even under severe class imbalance conditions, demonstrating its effectiveness in addressing minority class representation.
- **Low Imbalance**: The performance of HyperSMOTE-MC is consistent and robust across varying degrees of class imbalance, highlighting its adaptability to different data distributions.

5 Conclusion

In summary, HyperSMOTE-MC demonstrates superior performance in multiclass bot detection tasks on the TwiBot-20 dataset. By effectively handling class imbalances and leveraging hypergraph-based resampling, HyperSMOTE-MC achieves higher accuracy and robustness compared to existing methods. Our experiments confirm the efficacy of HyperSMOTE-MC in enhancing multiclass bot detection and offer insights into optimal configurations for its application.

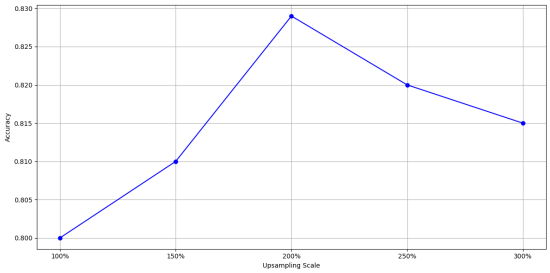

Fig. 2: Accuracy of HyperSMOTE-MC for different upsampling scales on the TwiBot-20 dataset.

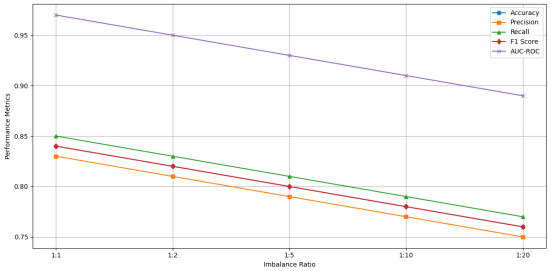

Fig. 3: Performance of HyperSMOTE-MC under different class imbalance ratios on the TwiBot-20 dataset.

References

1. Alkulaib, L., Lu, C.T.: Balancing the scales: Hypersmote for enhanced hypergraph classification. In: Proceedings of the IEEE International Conference on Big Data (2023)
2. AlKulaib, L.A.: Twitter Bots Multiclass Classification Using Bot-Like Behavior Features. Ph.D. thesis, The George Washington University (2018)
3. Allem, J.P., Ferrara, E.: Could social bots pose a threat to public health? American journal of public health **108**(8), 1005 (2018)
4. Broniatowski, D.A., Jamison, A.M., Qi, S., AlKulaib, L., Chen, T., Benton, A., Quinn, S.C., Dredze, M.: Weaponized health communication: Twitter bots and russian trolls amplify the vaccine debate. American journal of public health **108**(10), 1378–1384 (2018)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
6. Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S.: Who is tweeting on twitter: human, bot, or cyborg? In: Proceedings of the 26th annual computer security applications conference. pp. 21–30 (2010)
7. Davis, C.A., Varol, O., Ferrara, E., Flammini, A., Menczer, F.: Botornot: A system to evaluate social bots. In: Proceedings of the 25th international conference companion on world wide web. pp. 273–274 (2016)

8. Feng, S., Wan, H., Wang, N., Li, J., Luo, M.: Twibot-20: A comprehensive twitter bot detection benchmark. Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021)
9. Feng, S., Wan, H., Wang, N., Luo, M.: Botrgcn: Twitter bot detection with relational graph convolutional networks. In: Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. pp. 236–239 (2021)
10. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 3558–3565 (2019)
11. Ferrara, E., Varol, O., Davis, C., Menczer, F., Flammini, A.: The rise of social bots. Communications of the ACM **59**(7), 96–104 (2016)
12. He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). pp. 1322–1328. IEEE (2008)
13. Heidari, M., Jones, J.H.J., Uzuner, O.: An empirical study of machine learning algorithms for social media bot detection. In: 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). pp. 1–5 (2021). https://doi.org/10.1109/IEMTRONICS52119.2021.9422605
14. Khaund, T., Kirdemir, B., Agarwal, N., Liu, H., Morstatter, F.: Social bots and their coordination during online campaigns: A survey. IEEE Transactions on Computational Social Systems **9**(2), 530–545 (2022). https://doi.org/10.1109/TCSS.2021.3103515
15. Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. Information Sciences **467**, 312–322 (2018)
16. Liu, Y., Zhang, Z., Liu, Y., Zhu, Y.: Gatsmote: Improving imbalanced node classification on graphs via attention and homophily. Mathematics **10**(11), 1799 (2022)
17. Qi, S., AlKulaib, L., Broniatowski, D.A.: Detecting and characterizing bot-like behavior on twitter. In: Social, Cultural, and Behavioral Modeling: 11th International Conference, SBP-BRiMS 2018, Washington, DC, USA, July 10-13, 2018, Proceedings 11. pp. 228–232. Springer (2018)
18. Sleeman IV, W.C., Krawczyk, B.: Multi-class imbalanced big data classification on spark. Knowledge-Based Systems p. 106598 (2020)
19. Weeks, B.E., Ardèvol-Abreu, A., Gil de Zúñiga, H.: Online influence? social media use, opinion leadership, and political persuasion. International journal of public opinion research **29**(2), 214–239 (2017)
20. Wu, J., Teng, E., Cao, Z.: Twitter bot detection through unsupervised machine learning. In: 2022 IEEE International Conference on Big Data (Big Data). pp. 5833–5839 (2022). https://doi.org/10.1109/BigData55660.2022.10020983
21. Zhao, Tong, Z.X., Wang, S.: Graphsmote: Imbalanced node classification on graphs with graph neural networks. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining. pp. 833–841 (2021)
22. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. Advances in neural information processing systems **19** (2006)