Fan *et al. Cybersecurity* (2022) 5:19
https://doi.org/10.1186/s42400-022-00124-x

Cybersecurity

RESEARCH Open Access

# PUMD: a PU learning-based malicious domain detection framework

Zhaoshan Fan[1,2], Qing Wang[1,2], Haoran Jiao[1,2], Junrong Liu[1], Zelin Cui[1], Song Liu[1] and Yuling Liu[1,2*]

**Abstract**

Domain name system (DNS), as one of the most critical internet infrastructure, has been abused by various cyber attacks. Current malicious domain detection capabilities are limited by insufficient credible label information, severe class imbalance, and incompact distribution of domain samples in different malicious activities. This paper proposes a malicious domain detection framework named PUMD, which innovatively introduces Positive and Unlabeled (PU) learning solution to solve the problem of insufficient label information, adopts customized sample weight to improve the impact of class imbalance, and effectively constructs evidence features based on resource overlapping to reduce the intra-class distance of malicious samples. Besides, a feature selection strategy based on permutation importance and binning is proposed to screen the most informative detection features. Finally, we conduct experiments on the open source real DNS traffic dataset provided by QI-ANXIN Technology Group to evaluate the PUMD framework's ability to capture potential command and control (C&C) domains for malicious activities. The experimental results prove that PUMD can achieve the best detection performance under different label frequencies and class imbalance ratios.

**Keywords:** Malicious domain detection, Insufficient credible label information, Class imbalance, Incompact distribution, PU learning

## Introduction

As an important technical support of modern internet, DNS provides services for mapping domain name to IP address space. While providing users with convenient network services, domains have also been widely abused in malicious attacks, such as malware distribution, C&C communication, botnet control, phishing and spam.

The conduct of cyber attack activities often requires malicious domains as core resources to undertake the communication functions between the infected host and the attacker. Therefore, the detection of malicious domains is helpful to immediately block malicious activities and trace the source of attacks. In order to make the entire malicious infrastructure more robust, attackers often adopt resilient communication technologies such as Domain-flux or IP-flux. Thus, malicious domains are also different from benign domain names in characteristics of character level composition and communication traffic. In addition, due to the limited cost of attackers, they often register malicious domain names in batches or reuse malicious resources. Malicious domains can also be captured through domain registration information and the association among domains. In this paper, we focus on the detection scheme of active malicious domains that undertake communication functions in cyber attack activities, and capture malicious domains by analyzing character-level, traffic and registration characteristics, as well as the resource overlap of domains.

Most malicious domain detection solutions can be divided into three categories: black/white lists-based method (Sato et al. 2012; Kang and Lee 2007; Cao et al. 2008), knowledge-based method (Choi et al. 2007; Morales et al. 2009; Prieto et al. 2011; Villamarín-Salomón and Brustoloni 2009) , and machine learning-based method (Schiavoni et al. 2014; Yan et al. 2019; Sun et al. 2019; Shi et al. 2018; Liu et al. 2018; Wang et al.

*Correspondence: liuyuling@iie.ac.cn
[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
Full list of author information is available at the end of the article

2020; Tran et al. 2018; Tong et al. 2016; Du Peng 2020; Ma et al. 2014). In order to provide flexibility and resilience of communication, highly dynamic Domian-to-IP mapping was adopted in malicious infrastructure, static black/white list protection strategies are no longer applicable. Constructing malicious domain detection rules based on empirical knowledge is manpower consuming and easily bypassed. The mainstream detection methods use machine learning methods to automatically learn complex detection models. And the main problems faced by machine learning solutions include:

- Insufficient credible label information: Due to the high cost of manual labeling, labeled samples mostly come from public blacklists (The Spamhaus Project Ltd 2021; Phishtank 2021; Andre Correa 2021; SURBL.ORG 2021) and popular domain list (ALEXA-INTERNET 2021), which are insufficient and unreliable. Specifically, Stevanovic et al. (2015) cross-checks the Alexa Top 1M sites which are often used as benign domains and confirmed that about 15% of these popular domains had appeared in at least one blacklist, which makes the labeling of benign domains more challenging.
- Class imbalance: Malicious attack activities are hidden in voluminous normal DNS communications. Collecting real DNS traffic will build a data set with a severe imbalance between malicious and benign domains. Training directly on imbalanced data will make the classifier pay more attention to the majority class, while ignoring the ability to describe the minority malicious domains.
- Incompact distribution of domains adopted by different malicious activities: Indeed, the abnormal characteristics of malicious domains may differ from activity to activity, such as special communication modes or abnormal character-level combinations. This is because different malicious activities may adopt different malicious communication technologies and domain name composition skills. Therefore, malicious domain samples of different activities loosely distribute in feature space, which affects classification performance.

Considering the above problems, we propose a malicious domain detection framework named PUMD. And the specific solutions and contributions of this paper are summarized as follows:

- Propose a malicious domain detection framework based on PU learning. This framework can alleviate the problem of insufficient credible label information by using only a small number of labeled malicious domains to train classifier, and improve the impact of class imbalance via customizing sample weight to construct a cost-sensitive objective function.
- Construct detection features from two perspectives: the single domain and the domain association. In particular, we propose novel evidence features based on resource overlapping association to improve the incompact distribution of malicious domain samples. Besides, we also introduce a feature selection strategy based on permutation importance and binning to enhance the characterization capabilities of feature set.
- Compare PUMD with common machine learning methods and existing works on an open-source realistic imbalanced data set. Experiments prove that PUMD has the best detection performance and maintains system robustness under different label frequencies and class imbalance ratios.

The rest of this paper is organized as follows. "Background and motivation" section introduces the related works and explores the suitability of PU learning for malicious domain detection, "Proposed method" section describes the PUMD's framework and technical details, and "Experiments" section shows the experiments and results. We discuss the superiority of PUMD compared with existing works and the future work in "Discussion and future work" section, and summarize our work in "Conclusion" section.

## Background and motivation

### Malicious domain detection

This section briefly describes related malicious domain detection works and clarifies why the PUMD is better than existing solutions. A comprehensive comparison will be made in "Discussion and future work" section.

#### *Insufficient credible label information*

Some solutions are designed to reduce the required label information. Schiavoni et al. (2014) and Yan et al. (2019) both use a strategy of filtering first and then clustering to compress the required label information. Besides, HinDom (Sun et al. 2019) adopts metapath-based transduction classification on the heterogeneous information network of malicious domain communication associations, which can reduce the proportion of initial label samples. However, these works do not solve the unreliable problem of using popular domains as benign domain. In addition, some solutions set filtering rules to filter out benign domains with low confidence. For example, Shi et al. (2018) requires labeled benign domain persist for three months in the Alexa Top 10K. This kind of solutions limits the benign learning samples of the detection

model to popular domains, and lacks the ability to characterize unpopular benign domains. PUMD adopts PU learning model to solve the problem of insufficient credible label information, only a small amount of labeled malicious domains is required, and a wide range of unlabeled domain names are screened for credible benign domains. While compressing the required label information, PUMD enhances the ability to characterize the benign domains.

#### *Class imbalance*

The main solutions can be divided into two categories: (1) Data-level solutions, typically with data resampling strategy. Liu et al. (2018) undersamples the majority benign domains, which will lose part of the label information. While KSDom (Wang et al. 2020) oversamples the labeled malicious domains, which will increase the risk of overfitting, due to the similar construction of samples. (2) Algorithm-level solutions, typically with cost-sensitive methods. Tran et al. (2018) adopts cost-sensitive LSTM to alleviate the influence of class imbalance. However, considering the high calculation cost of the cost matrix, they set a fixed category weight, which requires prior knowledge of class imbalance ratio and not universal. PUMD adopts a customized sample weighting method, which effectively sets different sample weights according to the confidence each unlabeld sample labeling as benign receives and constructs a cost-sensitive objective function to solve the problem of class imbalance.

#### *Incompact distribution of domain samples in different malicious activities*

The malicious domain clusters with similar abnormal characteristics can be called "family" and there are three division schemes: DGA, malware, and malicious activity. The DGA family is identified by different DGA algorithms, mostly divided by character similarity (Schiavoni et al. 2014; Tong et al. 2016; Du Peng 2020). Besides, domain families determined by the malware type are detected by the similarity of malicious resources, such as the IP address or NS server (Ma et al. 2014). In addition, a malicious activity requires the coordination of different malicious domains, for example, a spam distribution activity involves the coordination of botnet C&C domains, malicious resource downloading domains, and spam content providing domains. It can be detected based on the characteristics of high co-occurrence probability and strong communication correlation (Sun et al. 2019). The existing binary classification works have not yet considered the incompact distribution of malicious domain samples. PUMD adopts novel evidence features based on resource overlapping association to effectively increase inter-class distance and reduce intra-class distance.

### PU learning

PU learning is a semi-supervised learning technique that builds a binary classifier based on positive samples and unlabeled samples, in order to predict unlabeled samples. PU learning is suitable for dealing with problems in binary classification where one category of data is impure or only one category of label is available. Early works (Lee and Liu 2003; Elkan and Noto 2008) have confirmed that PU learning can reach the performance of standard supervised learning. Since PU learning only requires one category of label information and has excellent performance, it has aroused widespread interest in the field of machine learning, and has been applied in practical scenarios such as knowledge base completion, medical diagnosis, financial risk control.

In recent years, the cyber security field has also considered introducing PU learning to solve a series of security threat discovery problems (Zhang et al. 2017; Sun et al. 2017; Luo et al. 2018; Wu et al. 2019; Dhamnani et al. 2021). A typical application is a malicious URL attack detection system POSTER (Zhang et al. 2017). Specifically, considering the highly dynamic composition of URLs, it is difficult to manually label a large number of URL requests, POSTER combines two-step and cost-sensitive PU learning strategies, and uses a small number of malicious URLs and a large number of unmarked URLs to train binary classifiers and help network security engineers effectively discover potential attack patterns. In addition, PU learning also has typical applications in malware detection problem. Researchers usually take apps downloaded from trusted sources as benign samples. Sun et al. (2017) pays attention to the behavior of cyber attackers publishing malware on trusted sources, and proposes PUDROID, an Android malware detection system based on PU learning. Experiments have proved that PUDROID can find nearly 100% of the mixed malware.

### Motivation

In this paper, we try to address an anomaly detection problem where the given dataset has the following three characteristics:

- Dataset only has one class of credible label.
- The number of credible annotated samples is rather small.
- Dataset has severe class imbalance.

Current malicious domain name detection works mostly adopt supervised, unsupervised and traditional semi-supervised learning schemes. However, since

both supervised and traditional semi-supervised learning schemes require the use of two classes of available label information, it is still necessary to use popular domains as benign domains, which leads to the noise problem in the labeled benign samples. Even though the malicious domains mixed in the popular domains can be eliminated by setting filtering rules, supervised and traditional semi-supervised learning schemes still limit the learning samples of benign domains to popular domains, resulting in the problem of poor modeling ability for non-popular benign domains. Beseides, the unsupervised learning schemes waste the guidance information of one class of credible label. Therefore, we can conclude that supervised, traditional semi-supervised and unsupervised learning schemes are not suitable for the research scenarios of this paper. In recent years, PU Learning schemes have agreat effect on dealing with one class of data that is impure and unavailable, which is very suitable for the characteristics of the dataset we introduced earlier. Therefore, we adopt the PU learning scheme to solve the problem of malicious domain name detection.

We further discuss why PU learning is effective for malicious domain detection problems. As we stated in "Introduction" section, constructing a benign domain ground truth is a difficult task, because we cannot identify a domain as a benign sample, even if it never appears in any known blacklist, but a batch of high-quality malicious domain labels can be obtained through pubilic blacklists or analysis by security researchers. This is just in line with the applicable scenario of PU learning, that is, the situation where one category of label information is hard to obtain. Therefore, we can use known malicious domains as labeled positive samples, and the domains to be detected as unlabeled samples, so as to obtain a malicious domain detection model through PU learning. Solve the problem of insufficient credible label information from the perspective of labeling dataset composition.

## Proposed method

This section will introduce the framework and technical details of PUMD, as shown in Fig. 1, there are three main modules:

- "Feature Extraction" module: Extract four types of typical features from two perspectives of the single domain and associations between domains, and a novel evidence feature is proposed to alleviate the influence of incompactly distributed malicious domains on detection performance.
- "Feature Selection" module: Perform a feature selection strategy based on permutation importance and binning to find the most discriminative and informative feature subset among plenty of original features, which enhances the characterization capabilities of feature set.
- "Two-Step PU Learning" module: Perform two-step PU learning processing to solve the problem of insufficient credible label information and class imbalance, in which step one is to obtain reliable negative(RN) samples based on iForest algorithm, and step two is to customize sample weight and train the classifier on a cost-sensitive objective function.

### Feature extraction

Existing works either extract local detection features from the character composition, the traffic behavior or auxiliary information of a single domain (Shi et al. 2018; Liu et al. 2018; Schüppen et al. 2018; Almashhadani et al. 2020; Antonakakis et al. 2010; Huang et al. 2010), or extract global detection features from the perspective of correlation and resource overlaping among domain names (Sun et al. 2019; He et al. 2019). We combine two

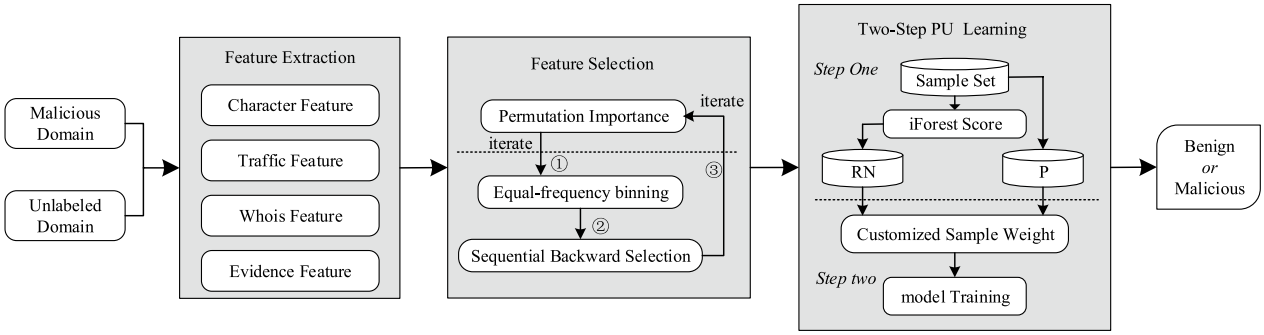



**Fig. 1** High-level overview of PUMD

feature extraction ideas to extract character, traffic, and whois features from single domain and propose a novel evidence feature based on resource overlapping association, capturing the association between the domains to be detected and the known malicious domain families. This section introduces the four types of typical malicious domain identification features and explains why these features can characterize malicious nature.

#### *Character feature*

Distinguish benign domains and malicious domains from the perspective of domain name character-level composition, see Table 1. **Structural features** focus on the structural attributes of the domain name: Since most short domain names have been registered, attackers usually increase the domain name length and the number of subdomains to obtain a larger domain name structure space (Shi et al. 2018; Liu et al. 2018), so we extract two structural features: domain name length (F1), number of subdomains (F2). **Linguistic features** capture the deviation of domain names' language mode: Considering that malicious domains are not memorable and readable, we extract five commonly used statistics (Schüppen et al. 2018; Almashhadani et al. 2020)to analyze the randomness of domain name characters: the number of special characters (F3), the number of numeric characters (F4), the conversion frequency of numeric and alphabetic characters (F5), the number of dictionary words (F6), and the number of unique length dicitonary words (F7).

#### *Traffic feature*

Capture abnormal characteristics from DNS traffic and resource record (RR), see Table 2.

For 91 days of DNS traffic data, we perform feature extraction according to two schemes: byday[*] and allday[†], that is, each day and the entire time span are seperately used as the observation period. Take F8 as example, we count the number of unique IPs resolved per day for each domain sample, and extract a total of 91 days as the byday feature. At the same time, we count the number of unique IPs resolved for each domain sample within 91 days as the allday feature. For 91-day byday features, further calculate statistics(sum, min, max, std, var, mean, 25%, 50%, 75%) to characterize the traffic distribution of domain sample and effectively compress the size of feature set.

**Domain resolution response features** analyze the RRs changes of the domain, focusing on typical

**Table 1** Character feature summarize

| Feature set | ID | Feature name | Mali DN profile |
|---|---|---|---|
| Structural feature | 1 | Domain name length | Longer |
| | 2 | Number of subdomains | Greater |
| Linguistic feature | 3 | Number of special characters | Greater |
| | 4 | Number of numeric characters | Greater |
| | 5 | Conversion frequency of numeric and alphabetic characters | Higher |
| | 6 | Number of dictionary words | Lesser |
| | 7 | Number of unique length dicitonary words | Lesser |

**Table 2** Traffic feature summarize

| Feature set | ID | Type | Feature name | Mali DN profile |
|---|---|---|---|---|
| Domain resolution response feature | 8 | */† | Number of unique IP addresses | Greater |
| | 9 | */† | Number of unique RR types | Greater |
| | 10 | */† | Range of unique RR parsing times | Lesser |
| | 11 | † | Number of unique domains hosted on IP pool | Greater |
| | 12 | † | Number of unique countries that IP pool belongs to | Greater |
| | 13 | † | Number of unique subvisions that IP pool belongs to | Greater |
| Client resolution request Feature | 14 | */† | Number of unique client | Lesser |
| | 15 | */† | Number of request count | Lesser |
| | 16 | † | Number of domains requested by client pool | Greater |
| | 17 | * | Ratio of the most frequent request hour to all-day requests | Greater |
| | 18 | * | Number of request count in different request period | Greater |

*byday (count sum, min, max, std, var, mean, 25%, 50%, 75%) , †allday

characteristics of malicious domains such as highly dynamic changes in RRs (Antonakakis et al. 2010), resource reuse (Sun et al. 2019), and loose geographic spatial distribution (Huang et al. 2010), the number of IPs (F8), the number of RRs (F9), the extreme difference in the parsing times of RRs (F10), the number of domains hosted on proxy IP pool (F11) and the distribution of IP pools in countries (F12) and regions (F13) are extracted respectively. **Client resolution request features** analyze the changes in the number of domain resolution requests, focusing on the discontinuity, long-term repetitive patterns and unusually active time of malicious domain resolution requests (Choi et al. 2009; Zhou et al. 2013). We extract the number of requesting clients (F14) and the number of parsing requests (F15) as statistical features. Besides, we count the total number of domains requested by the client pool (F16), where the client pool refers to all clients requesting the domain. Finally, we calculate the ratio of the most frequent request hour to all-day requests (F17), and count the client request volume in each period (F18) by dividing whole day into 6 periods: 0–3, 4–7, 8–11, 12–15, 16–19, 20–23.

#### *Whois feature*

Whois records provide detailed information about registered domains, which can assist in identifying malicious domains (Felegyhazi et al. 2010; Ma et al. 2009; Curtin et al. 2019; Hao et al. 2016). Normally, the valid period of malicious domain registration is short (F19), and the whois record usually lacks date information (F20). Besides, considering malicious domains have the characteristics of resource reuse, we calculate the similarity score of the name server(NS) domain names based on edit distance (F21), see Table 3.

#### *Evidence feature*

Considering above feature extraction schemes are all belongs to local detection features, malicious domains may be severely separated from each other in feature space due to inconsistent abnormal characteristics, we use the prior label knowledge of malicious domain family to construct effective evidence feature based on the overlapping associations of DNS communication and whois information resources, which could increase inter-class distance as well as reduce intra-class distance, see Table 4. The domain association construction scheme can refer to the Jaccard coefficient scheme proposed by He et al. (2019), and our work makes the following improvements:

1. Incorporate priori information of malicious domain family: a single domain is respectively associated with the resource set mapped by each known malicious domain family, and *no* represents associating with *no-th* family.
2. Construct new association rules: extend the association scheme based only on IP records to multiple resource associations, use relatedResource (abbreviated as *reRe*) to indicate the specific resource type associated.

The proposed evidence feature can be formalized as:

$$Jaccard\_no\_reRe(d_i) = J(d_i, D_{no}) = \frac{\left|reRe(d_i) \cap reRe(D_{no})\right|}{\left|reRe(d_i) \cup reRe(D_{no})\right|}$$

*Jaccard_no_reRe* represents the association between sample $d_i$ and known malicious domain family $D_{no}$ on *reRe* type resource, and *reRe*() represents the collection of *reRe* type resource collection. This paper mainly considers the overlap of DNS communication traffic (F22~F29) and whois information resources (F30~F36). The multiple types of resources mapped by domain $d_i$ are respectively associated with each family category (dimension = 9), and the resulting feature set is very large ($num(reRe) * dimension = 135$), which indeed needs a large amount of calculation. However, due to the different association characteristics of each malicious family, for example, some malicious families tend to map the same malicious IP address, while other malicious families tend to use the same registrar. We need to ensure the comprehensiveness of the association scheme in advance, then eliminate interference and invalid associations in "Feature selection" section, and achieve feature dimensionality reduction.

**Table 3** Whois feature summarize

| Feature set | ID | Feature name | Mali DN profile |
|---|---|---|---|
| Whois feature | 19 | Validity period of domain | Shorter |
| | 20 | Missing any of three whois dates (registration, update, expiration) | True |
| | 21 | Similarity Score of NS domain names | Higher |

### Feature selection

The current feature set is huge (33 groups, 299 features). The direct use of full features to predict malicious domains will undoubtedly lead to additional calculation and storage costs, increase the complexity of the algorithm, besides, the noise features will reduce the detection performance of the algorithm. Therefore, this paper proposes an automated feature selection strategy based on permutation importance and binning to select the most distinguishable detection feature set.

Common feature selection methods can be roughly divided into three categories: filter, wrapper, and embedding method. Filter method selects features based on divergence or correlation, independent of model training, fast calculation speed, but low degree of fit to the target problem, and poor feature selection effect. Embedding method automatically calculates the weight coefficients of features during the model training process, and selects features from large to small according to the weight, which fits the target problem well, but may face the risk of overfitting. Besides, the training model must support the calculation of feature weights, such as feature_importances of tree-based models. The feature selection in this paper adopts the wrapper method, in which features are continuously added or deleted to find the optimal feature subset according to the model prediction effect score. The wrapper method has a better fit to the target problem because of the participation of the model evaluating, and the non-integrated feature extraction scheme can reduce the risk of overfitting, and at the same time, it does not restrict the model trainer. However, the model must be retrained every time when a feature is picked, so the computational cost is high. This paper uses the binning strategy to effectively reduce the computational cost of feature selection in the packaged scheme. Specifically, this paper proposes an automated feature selection strategy based on Permutation Importance and binning, including the following two key steps:

#### *Subset evaluation step based on permutation importance*

The subset evaluation aims to determine the next candidate feature subset through some evaluation indicators. In this paper, the features with higher Pearson correlation coefficient(Corr) value and lower Permutation Importance(PI) value are selected to form the candidate set. Corr formula is as follows, where $f_i$ represents the *i-th* feature vector, $f_{ik}$ represents the $k-th$ dimension of $f_i$, a total of n dimensions, $\bar{f}_i$, $\sigma_i$ respectively represent mean and variance of $f_i$.

$$Corr(f_i, f_j) = \frac{\sum \left(f_{ik} - \bar{f}_i\right)\left(f_{jk} - \bar{f}_j\right)}{(n-1)\sigma_i\sigma_j}$$

And the pseudo code of PI scheme is shown in Algorithm 1:

**Algorithm 1** CALC_PI_SCORE

**Input:** $feature\_set$
**Output:** $PI_{score}$

```
baseline ←EVAL(feature_set)
for each feature f_i in feature_set do
    perm_fs ←permute f_i across feature_set
    PI_score[f_i] ← baseline−EVAL(perm_fs)
end for
```

#### *Sequential backward search step based on binning*

Considering the sequential backward search (SBS) has a high computational complexity, we propose a SBS strategy based on binning: for a given candidate feature set, it is divided into multiple feature bins of fixed size, and eliminate a feature bin for each backward search to reduce the computational overhead, as shown in Algorithm 2, we set different bin size in experiments.

**Algorithm 2** BIN_SBS

**Input:** $bin\_size$, $candidate\_set$, $feature\_set$
**Output:** $best\_score, best\_fl$

```
del_list ←EQUAL-FREQ_BIN(candidate_set)
for each del_fs in del_list do
    PUSH(score_list, EVAL(feature_set − del_fs))
end for
best_score ←MAX(score_list)
del_fl ← del_list[ARGMAX(score_list)]
best_fl ← feature_set − del_fl
```

The candidate sets are seperately sorted according to two rules:

- Natural sorting, try to remove the features of similar detection idea batch by batch.
- PI value sorting, try to remove the features with low model dependency batch by batch.

The proposed feature selection scheme based on permutation importance and binning is iterative as shown in Algorithm 3.

**Algorithm 3** feature selection based on PI and binning

**Input:** $feature\_set$, $PI_{thr}$, $Corr_{thr}$, $bin\_size\_list$
**Output:** $feature\_selection$

1: **function** PI_FS_EVAL($feature\_set$, $PI_{thr}$, $Corr_{thr}$)
2: $PI_{score} \leftarrow$ CALC_PI_SCORE($feature\_set$)
3: $Corr_{coef} \leftarrow$ CALC_CORR_COEF($feature\_set$)
4: **for** each feature $f_i$ in $feature\_set$ **do**
5: **if** $PI_{score}[f_i] < PI_{thr}$ **or** $Corr_{coef}[f_i] > Corr_{thr}$ **then**
6: PUSH($candidate\_set$, $f_i$)
7: **end if**
8: **end for**
9: **return** $candidate\_set$
10: **end function**

11: **function** BIN_FS($feature\_set$, $candidate\_set$, $bin\_size\_list$)
12: $fl_{nat} \leftarrow$ NATSORT($candidate\_set$)
13: $fl_{pi} \leftarrow$ PISORT($candidate\_set$)
14: **for** each $bin\_size$ in $bin\_size\_list$ **do**
15: $Nat_score, Nat_fl \leftarrow$BIN_SBS ($bin\_size$, $fl_{nat}$, $feature\_set$)
16: $PI_score, PI_fl \leftarrow$BIN_SBS ($bin\_size$, $fl_{pi}$, $feature\_set$)
17: PUSH($score\_list$, $Nat\_score$, $PI\_score$)
18: PUSH($fl\_list$, $Nat\_fl$, $PI\_fl$)
19: **end for**
20: $best_score \leftarrow$MAX($score\_list$)
21: $best\_fl \leftarrow fl\_list$[ARGMAX($score\_list$)]
22: **return** $best\_score, best\_fl$
23: **end function**

24: $best\_score \leftarrow$ EVAL($feature\_set$)
25: $best\_fl \leftarrow feature\_set$
26: **repeat**
27: $baseline \leftarrow best\_score$
28: $feature\_set \leftarrow best\_fl$
29: PI_FS_EVAL($feature\_set$,$PI_{thr}$,$Corr_{thr}$)
30: BIN_FS($feature\_set$,$candidate\_set$, $bin\_size\_list$)
31: **until** $candidate\_set \leftarrow \emptyset$ **or** $best\_score < baseline$
32: $feature\_selection \leftarrow feature\_set$

Algorithm 3 describes an automated feature selection strategy based on permutation importance and binning. This algorithm takes the original feature set *feature_set*, permutation importance threshold $PI_{thr}$, Pearson correlation coefficient threshold $Corr_{thr}$, list of bin sizes *bin_size_list* as inputs, and outputs the selected feature set *feature_selection*. It firstly initializes the highest prediction score *best_score* and the best feature set *best_fl* respectively (line24~line 25). Then, it iteratively executes the function of feature subset evaluation based on permutation importance and the function of sequential backward search based on binning (line 26~line30). And the termination condition is that the candidate feature set is empty or the prediction score is no longer improved (line31). Finally, the currently saved feature set is output as the feature selection result (line32). The details of the two functions are described below.

- The function of feature subset evaluation based on permutation importance (line 1~line 10): It calculates the permutation importance score and Pearson correlation coefficient of all features on the feature set (line 2~line 3). It further selects the features according to $PI_{thr}$ and $Corr_{thr}$ to form a candidate feature set *candidate_set* (line 4~line 8) and takes *candidate_set* as the return value of the function(line 9).
- The function of sequential backward search based on binning (lines 11~line 23): It separately sorts the *candidate_set* according to natural order and permutation importance order (line 12~line 13). Then it traverses different bin sizes and executes the sequential backward search function based on binning (line14~line19). Finally, the highest prediction score and the corresponding feature set are assigned to *best_score* and *best_fl* and serve as the output value of the function (lines 20–23).

#### Two-step PU learning for malicious domain detection

We innovatively introduce PU learning method to solve the problem of malicious domain detection and make corresponding improvements to the two-step model based on the field knowledge. Malicious domain detection model is trained with a small number of labeled malicious domains and all unlabeled domains as input, which alleviate the problem of insufficient label information and uncredible benign domains labeling from the perspective of trainset construction. Besides, to improve the impact of the imbalanced trainset on the classifier, we construct a cost-sensitive objective function in PUMD by setting customized sample weights.

The two-step PU learning process adopted in PUMD was illustrated in Fig. 2. The method input is labeled malicious domains($P$) and unlabeled domains ($U$), the method output is a binary prediction(malicious or benign), and the two-step process can be summarized as:

- Step One: Process the iForest algorithm over entire sample set $D$ to calculate isolation score $IS(d_D)$, and

**Table 4** Evidence feature summarize

| Feature set | ID | Feature name | Dim | Mali DN profile |
|---|---|---|---|---|
| DNS communication related | 22 | Jaccard_no_IP | 9 | Higher |
| | 23 | Jaccard_no_CNAME | | |
| | 24 | Jaccard_no_MX | | |
| | 25 | Jaccard_no_NS | | |
| | 26 | Jaccard_no_RR | | |
| | 27 | Jaccard_no_country | | |
| | 28 | Jaccard_no_subvision | | |
| | 29 | Jaccard_no_client | | |
| Whois information related | 30 | Jaccard_no_ns | | |
| | 31 | Jaccard_no_admin_email | | |
| | 32 | Jaccard_no_registrant | | |
| | 33 | Jaccard_no_tech_email | | |
| | 34 | Jaccard_no_r_whoisserver | | |
| | 35 | Jaccard_no_whoisserver | | |
| | 36 | Jaccard_no_sponsoring | | |

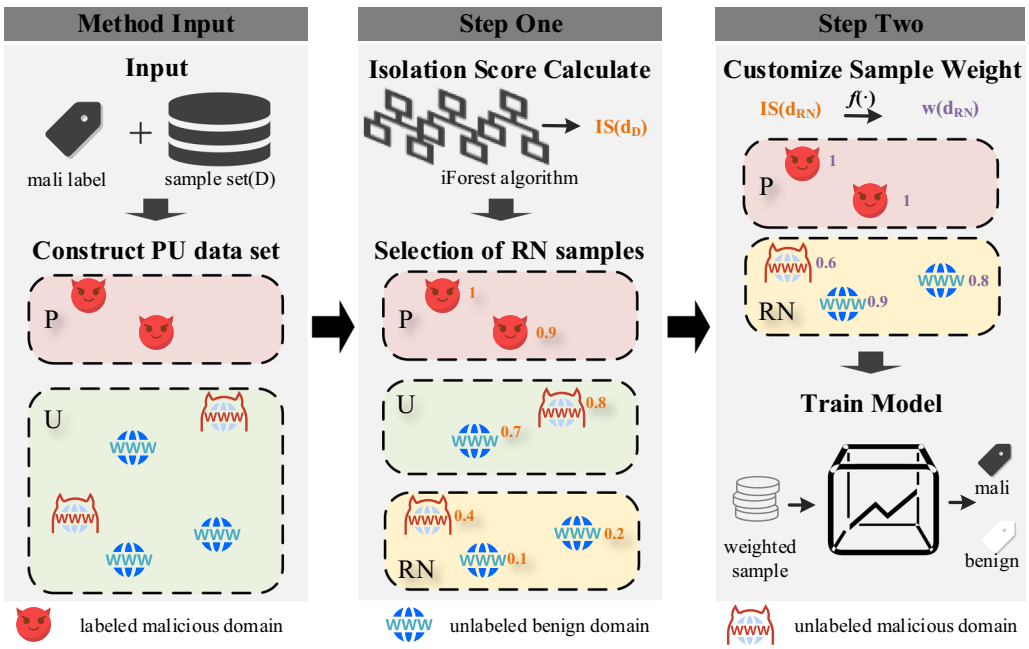



**Fig. 2** An overview of two-step PU learning process

filter unlabeled samples according to the isolation score threshold to obtain reliable benign domain set, called as reliable negative(*RN*).

- Step Two: Customize *RN* weight $w(d_{RN})$ according to the isolation score $IS(d_{RN})$, and further combine the labeled malicious domains to jointly train a supervised binary classification model based on cost-sensitive objective function.

***Problem statement and notations***

*Formal description:* In ordinary binary classification, we always adopt "true label"($y$) to indicate whether domain $d$ is malicious($y(d) = 1$) or benign($y(d) = 0$). In PU learning, the positive sample set $P$ corresponds to the labeled malicious domains, and the unlabeled sample set $U$ contains benign domains and potential malicious domains. Further define the "observation state"($s$) to identify whether a sample is labeled($s(d) = 1$) or not($s(d) = 0$).

*Construct PU data set:* Let $\chi = \mathbb{R}^n$ represent the feature space of all samples, $\gamma = \{0, 1\}$ represents the true label of the sample, and $\varsigma = \{0, 1\}$ represents the observation state of the sample, so the sample can be described as a tuple $\left(x(d_i) \in \chi, y(d_i) \in \gamma, s(d_i) \in \varsigma\right)$, The splitting scheme of sample set $D$ can be expressed as:

$$\begin{aligned} D =& P \cup U \\ =& \left\{ \left(x(d_i), y(d_i) = 1, s(d_i) = 1\right) | d_i \in P \right\} \\ & \cup \left\{ \left(x(d_i), y(d_i), s(d_i) = 0\right) | d_i \in U \right\} \end{aligned}$$

#### *Step one: obtain reliable normal samples*

As shown in Fig. 2, in Step one, we divide U into U and RN according to the isolation score, and only use RN containing fewer malicious domains as reliable negative samples for the next stage. The removal of U with more malicious domains will reduce false positives.

*Isolation Score Caculate:* iForest (Liu et al. 2008) is an unsupervised anomaly detection method, which uses a forest algorithm to model the isolation degree of samples. Specifically, the isolation score can be caculated by the path length from leaf sample to root node on each tree, as follows.

$$IS(d) = 2^{-\frac{E(h(d))}{c(m)}}$$

In which, $E(h(d))$ represents the average path length of sample $d$ on the iForest. For the malicious domain detection data set $D$, we have $Size(D) = m$, thus the average path length of all samples on a binary search tree is $c(m) = 2H(m) - 2(m-1)/m$, which is used to standardize $E(h(d))$ , and $H(m)$ is the harmonic number, can be estimated by $ln(n) + 0.5772156649$ (Euler's constant). The value range of Isolation Score $IS(d)$ is (0, 1) . The larger the $IS(d)$ value, the more abnormal the domain sample.

*Selection of RN samples: RN* samples are extracted from Unlabled set($U$), which should be quite different from the labeled malicious domains in the Positive set($P$). This paper calculates isolation score for all samples in malicious domain detection data set $D$, and selects *RN* samples by calculating the potential malicious and benign domains isolation score threshold. The steps are as follows:

- Caculate the average isolation score of samples in set $P$ as the threshold score of potential malicious domains:

$$\alpha = \frac{1}{l}\sum_{i=1}^{l} IS(d_i), Size(P) = l \tag{1}$$

- Calculate the average isolation score of samples in set $D$ as the threshold score of potential benign domains:

$$is = \frac{1}{m}\sum_{i=1}^{m} IS(d_i), Size(D) = m \tag{2}$$

- The original value of threshold $\beta$ is set as follows:

$$\beta_{org} = \begin{cases} is & is < \alpha \\ \alpha & is > \alpha \end{cases} \tag{3}$$

The *RN* sample set can be expressed as

$$RN = \left\{ d_i | IS(d_i) < \beta, d_i \in D \right\} \tag{4}$$

- Threshold $\beta$ directly affects the number of RN samples for model training. Considering the imbalance problem in set $D$, There is a relationship $Size(Benign) >> Size(Mali)$, set adjustable hyperparameters:

$$ad\_\beta \in \left( 0, \frac{\underset{d_i \in D}{max}(IS(d_i))}{\beta_{org}} \right] \tag{5}$$

Let $\beta = \beta_{org} \cdot ad\_\beta$ to control the number of negative samples participating in step two weighted classifier training.

#### *Step two: train weighted binary classifier for malicious domains detection*

As shown in Fig. 2, in Step two, we customize the sample weights for the samples in P and RN, and train the detection model based on the weighted training sample set.

*Customize sample weight:* Firstly, set the weights for all selected training samples: for the labeled malicious domains in set $P$, the weights are uniformly set to 1. And for the RN samples obtained in step one, the weights are customized according to their isolation scores. The design idea is that for domain samples with lower isolation scores, the possibility of abnormality is less, and the credibility of benignity is higher, so their weights as RN samples should be higher. Conversely, for domain samples with higher isolation scores, their weights should be quite low. The original formula for the weight $w(d)$ of the *RN* sample is as follows:

$$w_{org}(d_j) = \frac{\max\limits_{d_i \in D}(IS(d_i)) - IS(d_j)}{\max\limits_{d_i \in D}(IS(d_i)) - \min\limits_{d_i \in D}(IS(d_i))}, \quad d_j \in RN$$

In order to solve the severe class imbalance in the malicious domain data set $D$, we set the weight adjustment parameter $ad\_weight = \frac{l}{\sum_{d_j \in RN} w_{org}(d_j)} \cdot ratio_{P/RN}$ to balance the weight ratio of positive samples and negative samples for training a weighted binary classifier. Let $w(d) = w_{org}(d) \cdot ad\_weight$, where $l$ is the total weight of the positive sample, and $ratio_{P/RN}$ represents the best ratio coefficient of the positive sample weight and the negative sample weight, which is the second adjustable hyperparameter.

*Train model:* According to the above training samples and their weights, a weighted binary classification model can be trained to distinguish between malicious domains and benign domains, the following objective function is minimized:

$$\sum_{d_i \in D} w(d_i) \cdot L\big(y(d_i), f_\theta(x(d_i))\big) + \lambda R(\theta)$$

In which, $L\big(y(d_i), f_\theta(x(d_i))\big)$ is the misclassification loss of sample $d_i$, $w(d_i)$ represents the sample weight of $d_i$. The first item of the objective function $\sum_{d_i \in D} w(d_i) \cdot L\big(y(d_i), f_\theta(x(d_i))\big)$ means calculating cost-sensitive misclassification loss of all training samples, the second term represents the construction of a regular loss for the model parameter $\theta$, which is used to reduce the complexity of the model.

So far, we have completely established a malicious domain detection framework based on two-step PU learning. In our specific experiments, we mainly compared the performance of multiple commonly used machine learning classifier algorithms applied to the PUMD model, including: Random Forest (RF), Xgboost, Support Vector Machine (SVM), Logistic Regression (LR).

## Experiments

### Experiments design

The experiments are designed to test whether PUMD can effectively solve the three problems faced by malicious domain detection field:

- Insufficient credible label information: Use label frequency as an experimental hyperparameter to control label information ratio, and compare methods using only malicious domain label information, which include PUMD and other PU learning solutions, with the ordinary machine learning solutions using both benign and malicious domain labels, to test whether PUMD can achieve superior and robust detection performance under a small amount of credible malicious domain label information, see "Evaluating the PUMD" section.
- Class imbalance: Use label frequency to adjust the class imbalance ratio of train and test set, and test whether PUMD can achieve excellent and robust detection performance on imbalanced real traffic, see "Evaluating the PUMD" section.
- Incompact distribution of malicious domain samples: Compare the detection performance of PUMD on the three groups of feature sets, and test whether the evidence feature and feature selection scheme can enhance the characterization capabilities of feature set and effectively increase inter-class distance and reduce intra-class distance, see "Evaluating the feature extraction and selection" section.

Finally, we compare our PUMD with other existing malicious domain detection methods, and comprehensive experiments verify the effectiveness and superiority of our proposed method, see "Quantitative comparison with related works" section. And we published all source code on GitHub (https://github.com/fzs-git/PUMD) for other researchers to reproduce experiments and verify the performance of PUMD.

### Dataset summarize

#### *Data set*

The data set used in this paper comes from the open source 91-day anonymized DNS communication traffic from 2020-3-01 to 2020-05-31 on the large-scale real-world network provided by QI-ANXIN Technology Group, involving 20512 domains and 431130 IP addresses, both have undergone encoding anonymization treatment. In addition, we also get the whois information of the domains and the geographic information of the ip addresses. Table 5 provides an overall overview of the open source malicious domain data set, which is currently published by QI-ANXIN Technology Group (QI-ANXIN 2007), and this paper has been licensed for use.

#### *Ground truth and train-test split*

The ground truth is manually annotated by QIANXIN's researchers and contains 952 malicious C&C domains. And the malicious domains can be divided into 9 malicious domain families according to different advanced persistent threat activities. Considering the sensitivity of the information, the details of the specific apt are anonymized, which are referred to as class_1 to class_9 in this article. In order to carry out

**Table 5** Data set summarize

| Domains | Resolution times | RR records | |
|---|---|---|---|
| 20512 | 1987609 | 66048 | |
| **Clients** | **Request times** | **Ipv4** | **Ipv6** |
| 370101 | 4163432 | 419155 | 11975 |

**Table 6** Data set details

| NO | Type | | Size | Label frequency | Train | Test |
|---|---|---|---|---|---|---|
| 1 | **Mali** | Class_1 | 672 | $c$ | $672 \cdot c$ | $672 \cdot (1-c)$ |
| 2 | | Class_2 | 12 | | $12 \cdot c$ | $12 \cdot (1-c)$ |
| 3 | | Class_3 | 36 | | $36 \cdot c$ | $36 \cdot (1-c)$ |
| 4 | | Class_4 | 34 | | $34 \cdot c$ | $34 \cdot (1-c)$ |
| 5 | | Class_5 | 47 | | $47 \cdot c$ | $47 \cdot (1-c)$ |
| 6 | | Class_6 | 16 | | $16 \cdot c$ | $16 \cdot (1-c)$ |
| 7 | | Class_7 | 115 | | $115 \cdot c$ | $115 \cdot (1-c)$ |
| 8 | | Class_8 | 11 | | $11 \cdot c$ | $11 \cdot (1-c)$ |
| 9 | | Class_9 | 9 | | $9 \cdot c$ | $9 \cdot (1-c)$ |
| – | | (Total) | 952 | | $952 \cdot c$ | $952 \cdot (1-c)$ |
| 10 | **Benign** | (Total) | 19560 | – | 0 | 19560 |
| – | **DataSet** | (All) | 20512 | – | $952 \cdot c$ | $20512 - 952 \cdot c$ |

the experiments, we divide the labeled data set into two parts: train set and test set. In PU learning, the train set corresponds to the labeled malicious domain set ($P$) and the test set corresponds to the unlabeled domain set ($U$). The detailed information is shown in Table 6.

In which, "label frequency" indicates the proportion of the number of labeled malicious domains to all malicious domains in the data set, and the label frequncy value is denoted as c, thus we have the equation: $c = Size(P)/Size(Mali)$.

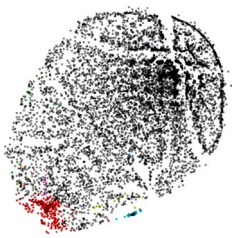


**Fig. 3** feature distribution display via t-sne

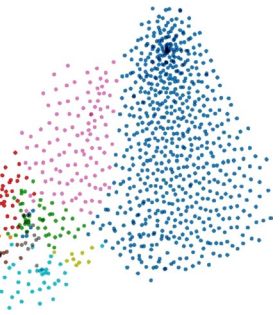


**Fig. 4** feature distribution of malicious domain families display via t-sne

#### *Feature spatial distribution*

In order to be intuitive, we use T-SNE (Van der Maaten and Hinton 2008) to visualize the feature spatial distribution of the sample set at 50% label frequency. As is shown in Fig. 3, benign domains are represented by black markers, while malicious domains belonging to different families are drawn in different colors and densely clustered in the lower left corner, which can effectively distinguish malicious samples from benign samples. And the feature space of 9 malicious domain clusters is further visualized in Fig. 4.

## Evaluating the classifiers and evaluation metrics

### *Classifiers and evaluation metrics selection*

We adopt various commonly used evaluation metrics(see Table 7) to test the performance of PUMD on several machine learning algorithms including: RF, Xgboost, SVM and LR. We use label frequency as the control variable to adjust the label information ratio and sample category imbalance rate, experimental results see Fig. 5. Experiments have proven that ACC, ROC, AUC, Precision and Recall are not suitable for imbalanced data set(for further details, see "Appendix" section): ACC is biased to consider the performance of the majority category. Precision or Recall is one-sided, separately consider the ability of detect correctly or completely. In ROC, since the number of negative samples is huge, FPR grows slowly, and the AUC actually contains a large area of no interest. We finally retain the F1 (Fig. 5d) and MCC (Fig. 5f) metrics. Among them, F1 comprehensively considers the Recall and Precision of the classifier, while MCC describes the correlation coefficient between the predicted results and the actual results ,often used in the case of imbalanced data set. It can be seen that the performance of the RF on both metrics maintains the best results.

**Table 7** Metric summarize

| Metric | Description |
|---|---|
| TP | Malicious domains labeled as malicious |
| FP | Benign domains labeled as malicious |
| TN | Benign domains labeled as benign |
| FN | Malicious domains labeled as benign |
| ACC | $(TP + TN)/(TP + FP + TN + FN)$ |
| Precision | $TP/(TP + FP)$ |
| Recall | $TP/(TP + FN)$ |
| F1 | $2 \times (precision \cdot recall)/(precision + recall)$ |
| TPR | $TP/(TP + FN)$ |
| FPR | $FP/(FP + TN)$ |
| ROC | A curve plotting TPR against FPR with various thresholds |
| AUC | Area under the ROC curve |
| MCC | $\frac{(TP \cdot TN - FP \cdot FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$ |

### *Analysis detection performance trends*

Observing the change trend of the two metrics (F1 and MCC) on the label frequency, as the amount of labeled samples increases, the detection performance decreases, which is inconsistent with common sense. This is because the increase in labeling frequency leads to more severe class imbalance in the Test set (expressed as Ratio) as the numbers of benign samples remain stable at 19560 while the malicious samples keep decreasing from 852 to 91, which has become the main factor affecting the performance of the classifier, see Table 8 for details. More specifically, focusing on the evaluation results on the test set, severe class imbalance causes the TP samples to decrease proportionally while the FP samples

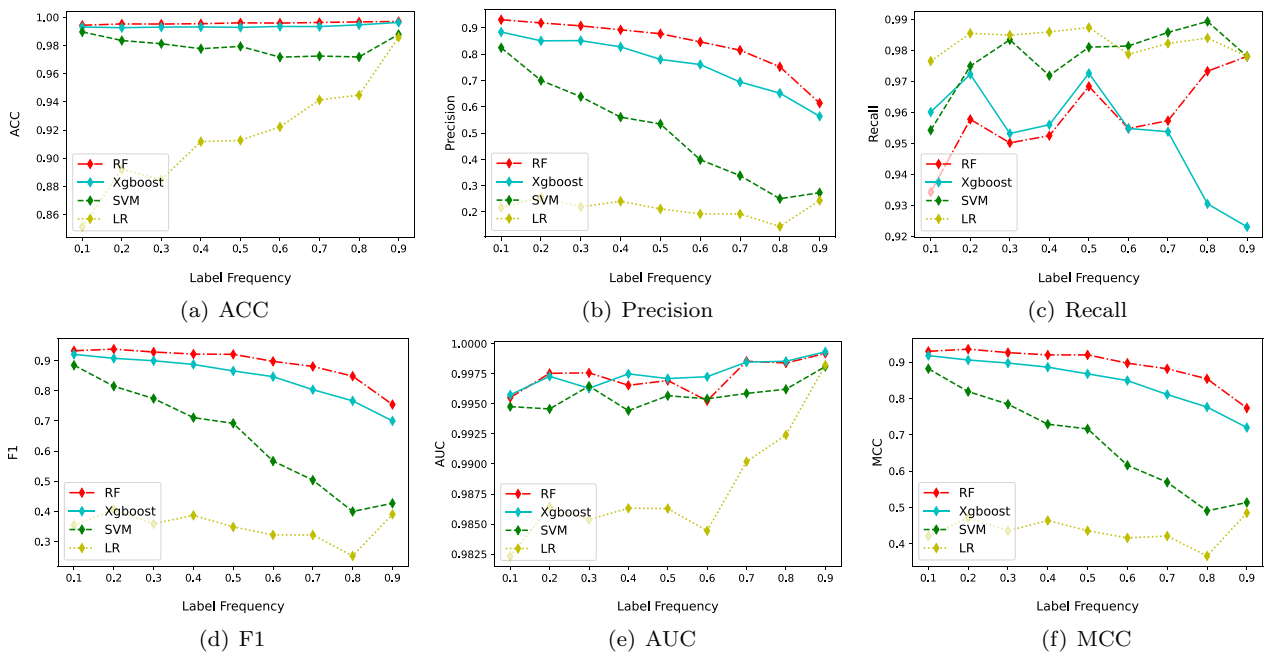



**Fig. 5** Metrics evaluate score with different label frequencies. (**a**) Prediction ACC changes. (**b**) Prediction Precision changes. (**c**) Prediction Recall changes. (**d**) Prediction F1 changes. (**e**) Prediction AUC changes. (**f**) Prediction MCC changes

**Table 8** Test data summerize

| Label frequency | Train (Mali) | Test | | | Result |
|---|---|---|---|---|---|
| | | Mali | Benign | Ratio | TP–FP–TN–FN |
| 0.1 | 100 | 852 | 19560 | 1:22.9 | 796–59–19501–56 |
| 0.2 | 195 | 757 | | 1:25.8 | 725–64–19496–32 |
| 0.3 | 290 | 662 | | 1:29.5 | 629–64–19496–33 |
| 0.4 | 384 | 568 | | 1:34.4 | 541–65–19495–27 |
| 0.5 | 478 | 474 | | 1:41.2 | 459–64–19496–15 |
| 0.6 | 576 | 376 | | 1:52.0 | 359–65–19495–17 |
| 0.7 | 671 | 281 | | 1:69.6 | 269–61–19499–12 |
| 0.8 | 765 | 187 | | 1:104.5 | 182–60–19500–5 |
| 0.9 | 861 | 91 | | 1:214.9 | 89–56–19504–2 |

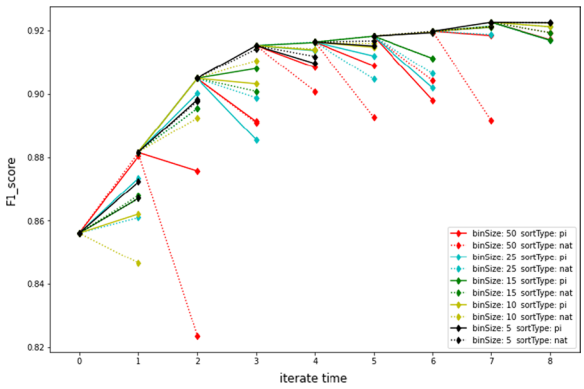



**Fig. 6** Visualization of feature selection process based on PI and binning

remains stable, thus the fault tolerance of FP decreases multiplicatively(regarded as Precision), resulting in a downward trend in overall performance. However, the overall decline of FN and the increasing ratio of TP/Mali (regarded as Recall) show that the classifier's ability to characterize malicious domains is actually enhanced with more malicious domain label information, which is reasonable.

#### Evaluating the feature extraction and selection

This section further discusses the effectiveness of the novel evidence feature and the feature selection strategy proposed. Three control groups are set for experiments, as shown in Table 9:

- Original group represents general features extracted from single domain, including character, traffic and whois features.
- Feature_Append group appends evidence features on the basis of Original group. Specifically, multiple resource types mapped by domain($num(reRe) = 15$) are respectively associated with each known family($dim = 9$), and the resulting evidence feature set is large ($num(reRe) * dim = 135$).
- Feature_Selection group performs the feature selection process described in "Feature selection" section over Feature_Append group. Practically, we initialize $PI_{thr} = 7E{-}6$, $Corr_{thr} = 0.95$ and $bin\_size\_list = [5, 10, 15, 25, 50]$, iterate the whole feature selection process under different label frequencies and different evaluation metrics(F1 or MCC), and finally we get the 92 features. Take the feature selection process under label frequency = 0.5 and F1 mtrics for an example, see Fig. 6.

We can see that the overall number of iterations in Fig. 6 is 8 times, and the effective iteration is the first 7 times. The 8th feature screening failed to improve the detection performance, and the feature selection process is terminated. As we can see, the nat sorting in the early stage of the iteration is better than the PI sorting. At the first iteration, the feature is removed from the perspective of feature extraction. Thereafter, the removal of features with low model dependency depends on PI score. In addition, the bin_size chosed in the early stage of the iteration is generally larger, and the size of the subsequent bin is gradually reduced, which realizes an effective feature selection scheme of first large-scale screening and then refined screening.

According to the optimal base classifier RF determined in the previous section and two evaluation metrics F1 and MCC, the performance histograms are drawn (Fig. 7).

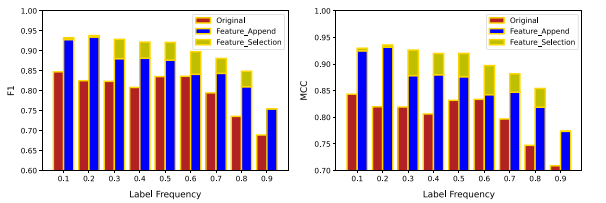
Comparing the Original and Feature_Append groups can analyze the effectiveness of the evidence features. It can be seen that the evidence feature can definately improve the detection performance of the model. Comparing the Feature_Append and Feature_Selection groups can analyze the effectiveness of the feature selection scheme. It can be seen that the feature selection operation helps to keep the performance of the model stable. The experimental results prove that the proposed novel evidence feature and the feature selection strategy are applicable to the field of malicious domain detection, **which can improve the incompact distribution of malicious domains and enhance the characterization capabilities of feature set**. 92 features of Feature_Selection group are ultimately used in PUMD framework.

#### Evaluating the PUMD

As the base classifier, evaluation metrics, and feature set have been determined above, this section mainly compares the PUMD with other PU learning schemes and ordinary machine learning schemes (see Table 10 for

**Table 9** Feature set summarize

| Feature set | Character feature | Traffic feature | Whois feature | Evidence feature | Feature select | Size |
|---|---|---|---|---|---|---|
| Original | ✓ | ✓ | ✓ | – | – | 164 |
| Feature_Append | ✓ | ✓ | ✓ | ✓ | – | 299 |
| Feature_Selection | ✓ | ✓ | ✓ | ✓ | ✓ | 92 |




(a) F1 (b) MCC

**Fig. 7** Feature set evaluation performance histogram. (**a**) Prediction F1 changes. (**b**) Prediction MCC changes

details) and illustrates the superiority of PUMD. Specifically, we adopt label frequency as an experimental hyperparameter to control label information ratio and adjust the class imbalance ratio of data set, and test whether PUMD can solve the problem of insufficient credible label information and class imbalance. Experimental results are shown in Fig. 8 and Table 11.

As we can see, PUMD achieves the superior and robust detection performance under different label frequencies, and the formal description of the data set used by each scheme is shown in Table 12. PUMD compresses half of the label information required by ordinary machine learning schemes via discarding the unreliable benign domain label information, while achiving a excellent performance at 0.1 label frequency, which has proven that **PUMD can alleviate the problem of insufficient credible label information**. In addition, as shown in Table 8, there is a severe class imbalance at 0.9 label

**Table 10** PUMD and multiple comparison algorithms analysis

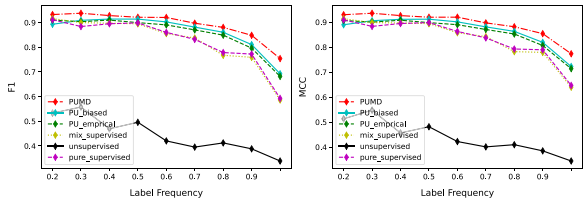
| Method | Strategy | Base classifier | Labeled sample | | Full SampleSet training | Sample weight | Customized sample weight |
|---|---|---|---|---|---|---|---|
| | | | Mali DN | Benign DN | | | |
| PUMD | Two-step | RF | ✓ | – | ✓ | ✓ | ✓ |
| PU_biased | Biased Learning | | ✓ | – | ✓ | ✓ | – |
| PU_empirical | Incorporation with Class Priori | | ✓ | – | ✓ | ✓ | – |
| mix_supervised* | Supervised | | ✓ | ✓(mix) | – | – | – |
| pure_supervised | | | ✓ | ✓(pure) | – | – | – |
| unsupervised | Unsupervised | iForest | – | – | ✓ | – | – |

*baseline

**Table 11** PUMD and multiple comparison algorithms performance evaluation

| Label frequency | PUMD | | PU_biased | | PU_empirical | | mix_supervised | | Pure_ supervised | | Unsupervised | | Improve* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | MCC | F1 | MCC | F1 | MCC | F1 | MCC | F1 | MCC | F1 | MCC | F1 | MCC |
| 0.1 | **0.9326** | **0.9297** | 0.8932 | 0.8895 | 0.9105 | 0.9066 | 0.9163 | 0.9137 | 0.9115 | 0.9091 | 0.5330 | 0.5131 | 0.0163 | 0.0160 |
| 0.2 | **0.9379** | **0.9357** | 0.9092 | 0.9058 | 0.9036 | 0.9001 | 0.9001 | 0.8984 | 0.8841 | 0.8831 | 0.5578 | 0.5479 | 0.0378 | 0.0373 |
| 0.3 | **0.9284** | **0.9262** | 0.9143 | 0.9114 | 0.9101 | 0.9078 | 0.8974 | 0.8965 | 0.8952 | 0.8945 | 0.4695 | 0.4566 | 0.0310 | 0.0297 |
| 0.4 | **0.9216** | **0.9198** | 0.9141 | 0.9118 | 0.8998 | 0.8980 | 0.8951 | 0.8941 | 0.8989 | 0.8983 | 0.4959 | 0.4816 | 0.0266 | 0.0257 |
| 0.5 | **0.9208** | **0.9199** | 0.9028 | 0.9012 | 0.8906 | 0.8899 | 0.8555 | 0.8578 | 0.8603 | 0.8628 | 0.4204 | 0.4223 | 0.0652 | 0.0621 |
| 0.6 | **0.8975** | **0.8971** | 0.8817 | 0.8811 | 0.8704 | 0.8707 | 0.8377 | 0.8422 | 0.8320 | 0.8370 | 0.3953 | 0.4009 | 0.0598 | 0.0549 |
| 0.7 | **0.8805** | **0.8816** | 0.8608 | 0.8631 | 0.8491 | 0.8526 | 0.7675 | 0.7820 | 0.7786 | 0.7925 | 0.4122 | 0.4096 | 0.1130 | 0.0996 |
| 0.8 | **0.8485** | **0.8541** | 0.8108 | 0.8192 | 0.7965 | 0.8066 | 0.7592 | 0.7790 | 0.7724 | 0.7894 | 0.3882 | 0.3849 | 0.0893 | 0.0750 |
| 0.9 | **0.7542** | **0.7736** | 0.6926 | 0.7226 | 0.6820 | 0.7140 | 0.5863 | 0.6398 | 0.5928 | 0.6470 | 0.3390 | 0.3444 | 0.1679 | 0.1338 |

**Underlined value**: overall best method, *count as *Score* (PUMD) − *Score* (mix_supervised(baseline))




(a) F1 (b) MCC

**Fig. 8** PUMD and multiple comparison algorithms performance evaluation. (**a**) Prediction F1 changes. (**b**) Prediction MCC changes

frequency(Ratio $\approx$ 200:1), and PUMD remain acceptable detection performance, which has proven that **PUMD can improve the impact of class imbalance**. And the following is a detailed analysis of the above experimental results.

#### *PU learning schemes*

Can be divided into three typical categories: two-step techniques, biased learning and the class prior incorporation (Bekker and Davis 2020). **PU_biased** belongs to the biased learning scheme, which takes all unlabeled samples as negative for training and adjusts the weight of the positive and negative samples as $W(P) : W(N) = Size(Benign) : Size(Mali)$ to deal with the noise in negative sample sets, which also can be regarded as a certain strategy for balancing the sample categories. **PU_empirical** belongs to the class prior incorporation scheme, which aims to derive the PU data based empirical risk minimization formula $\widehat{R}_{\prime}(g|x,s) = \frac{1}{|s|}\left(\sum_{x|s=1}\frac{1}{c}L^{+}(g(x)) + \left(1-\frac{1}{c}\right)L^{-}(g(x)) + \sum_{x|s=0}L^{-}(g(x))\right)$ via the prior knowledge of the category (see Bekker et al. (2019) for the derivation process), so as to construct a new data set for training the model: unlabeled samples are used as negative samples with weight 1, whereas each labeled sample is treated as a combination of positive sample with weigh $1/c$ and negative sample with weight $(1-1/c)$. Considering that there is no class balancing process in PU_empirical scheme, we use *class_weight* when training the classifier to further adjust the sample weight for better performance.

The common point of three PU learning schemes is that they do not need labeled benign domains and use the full sample set for classifier training. The advantage of PUMD over PU_biased and PU_empirical is that the sample weight is customized, and the confidence of each sample can be properly adjusted to train the classifier, while PU_biased and PU_empirical unify the sample weight.

#### *Ordinary machine learning schemes*

Considering that the proportion of mixed malicious domain samples in benign label is as high as 15% (Stevanovic et al. 2015). **Mix_supervised(baseline)** randomly samples Size(P) samples from the set U as negative samples (the mixed ratio is less than 4%), and trains the classification together with the set P. It is used as the baseline to reflect the performance improvement of the PUMD compared with the conventional detection scheme. **Pure_supervised** is the upper limit performance of the conventional malicious domain detection scheme, *Size*(*P*) negative samples are randomly sampled from the benign domains and the classifier is trained together with the set *P*.

Compared with mix_supervised and pure_supervised schemes, PUMD has the advantage that PU learning can use the full sample set for training, while the supervised scheme requires fully labeled samples and has the best performance with balanced data, which limit the number of samples participating in model training. Taking $label\_frequency = 0.5$ as an example, the train-set size ratio of PU learning and supervised learning is $20512 : 478 * 2 \approx 21{:}1$. More sample information significantly improves the classifier's ability to describe benign domains. By looking at the performance line graph, it can be seen that the PU learning schemes are overall higher than the supervised learning schemes.

In addition, because the PUMD adopts the unsupervised algorithm iForest, we use iForest as an **unsupervised scheme** to conduct a comparative experiment. Specifically, the entire data set is arranged in order according to the isolation score to find the best segmentation threshold and calculate the evaluation metrics. It can be seen from the performance curve that the performance of the unsupervised scheme is always lower than others. This is because unsupervised algorithm lacks the guidance of label information and cannot describe benign domains and malicious domains well.

In summary, the advantage of PUMD compared with ordinary solution is that it uses more sample information and label guidance information. Compared with other PU learning solutions, PUMD adopts customized sample weight, and the confidence of each unlabeled sample is considered separately. therefore, the classifier in PUMD can learn more accurate discrimination information and enhance the ability to characterize the benign and malicious domains.

### Quantitative comparison with related works

To further confirm the effectiveness of the PUMD scheme proposed in this paper, we select four related works listed in the "Background and motivation" section

**Table 12** Formal description of the data set

| Method | Labeled sample | | Unlabeled sample | TrainSet | | TestSet |
|---|---|---|---|---|---|---|
| | Mali DN | Benign DN | | P | N | |
| PUMD | $P = c * Mali$ | – | $U = D - P$ | $W(P)$ | $\cup w(U)$ | $U$ |
| PU_biased | | | | | $W(U)$ | |
| PU_empirical | | | | | $W(P) \cup W(U)$ | |
| mix_supervised* | | $N_{mix} = Sample_{size(P)}(D - P)$ | $D - P - N_{mix}$ | P | $N_{mix}$ | |
| pure_supervised | | $N_{pure} = Sample_{size(P)}(Benign)$ | $D - P - N_{pure}$ | | $N_{pure}$ | |
| unsupervised | – | – | D | D | | |

*baseline

**Table 13** Performance of different malicious domain detection methods

| Method | Macro-Recall | Macro-Precision | Macro-F1 | Recall | Precsion | F1 | ACC | FP-Rate | MCC |
|---|---|---|---|---|---|---|---|---|---|
| AULD | 0.5806 | **0.9471** | 0.6277 | 0.1618 | **0.9333** | 0.2757 | 0.9606 | **0.0006** | 0.3796 |
| ELM | 0.6827 | 0.7197 | 0.6993 | 0.3797 | 0.4592 | 0.4157 | 0.9668 | 0.0143 | 0.4007 |
| KSDom | 0.8911 | 0.7686 | 0.8172 | 0.8038 | 0.5435 | 0.6485 | 0.9729 | 0.0217 | 0.6482 |
| HAC_EasyEnsemble | 0.8241 | 0.5594 | 0.5559 | 0.8354 | 0.1252 | 0.2177 | 0.8134 | 0.1873 | 0.2774 |
| PUMD | **0.9825** | 0.9384 | **0.9594** | **0.9684** | 0.8776 | **0.9208** | **0.9961** | 0.0033 | **0.9199** |

**Underlined value**: overall best performance

for experiments, including AULD (Yan et al. 2019), ELM (Shi et al. 2018), KSDom (Wang et al. 2020), and HAC_EasyEnsemble (Liu et al. 2018). Since these works do not disclose their experimental datasets, we reproduce the detection schemes of these four works and conduct experiments based on the datasets of this paper.

- AULD. It proposes an unsupervised algorithm based suspicious APT domains detection framework, which combines the canopy and k-means algorithms. Besides, it extracts t ten important features from the host, domain name, and time from a large number of DNS log data. The evaluation metrics of AULD include accuracy, TP-Rate(Recall), and FP-Rate.
- ELM. As the first work to introduce Extreme Learning Machine (ELM) algorithm to detect C&C communication domains in APT attacks, it extracts 9 detection features from multiple data sources. The features are summarized as four categories, including construction-based, IP-based, TTL-based and WHOIS-based features. The evaluation metric adopted by ELM is the accuracy.
- KSDom. It proposes an imbalanced data processing scheme based on K-means and Smote algorithms, and applies it to the malicious domain detection framework. It extracts a total of 16 composite detection features, which can be summarized into three categories: domain name-based features, DNS answers-based features, and contextual features. In addition, it adopts the Catboost algorithm to train the malicious domain name detection model. KSDom's evaluation metrics include Accuracy, Precision and F1-Score.
- HAC_EasyEnsemble. It handles the sample imbalance problem in the field of malicious domain detection field by combining Hierarchical Agglomerative Clustering (HAC) and EasyEnsemble algorithms. It extracts 16 detection features from the characters of the domain name and dynamic DNS resolution data, and adopts Macro-precision, Macro-recall, and Macro-F1 as the evaluation metrics.

***Quantitative experiment 1***

In the case of label frequency = 0.5, the detection effects of the proposed PUMD scheme and four related works are compared, as shown in Table 13. In addition to the F1 and MCC adopted in this paper, the evaluation metrics used in Table also include the evaluation metrics of four related works. In addition, considering the processing capability of KSDom and HAC_EasyEnsemble for imbalanced data, according to the maximum imbalance ratio supported by their paper, imbalanced training sample sets for benign and malicious domains are respectively constructed (the imbalance ratio in KSDom is set to 10:1,

**Table 14** Performance of different malicious domain detection models

| Model | Macro-Recall | Macro-Precision | Macro-F1 | Recall | Precsion | F1 | ACC | FP-Rate | MCC |
|---|---|---|---|---|---|---|---|---|---|
| $AULD_{model}$ | 0.8581 | **0.963** | 0.9032 | 0.7185 | **0.9396** | 0.8143 | 0.9848 | **0.0022** | 0.8144 |
| $ELM_{model}$ | 0.9612 | 0.9414 | 0.951 | 0.9262 | 0.8851 | 0.9052 | 0.994 | 0.0039 | 0.9023 |
| $KSDom_{model}$ | 0.982 | 0.869 | 0.9175 | 0.9726 | 0.7388 | 0.8397 | 0.991 | 0.0085 | 0.8435 |
| $HAC\_EasyEnsemble_{model}$ | 0.9725 | 0.7846 | 0.8527 | 0.9684 | 0.5702 | 0.7177 | 0.9763 | 0.0234 | 0.7334 |
| PUMD | **0.9825** | 0.9384 | **0.9594** | **0.9684** | 0.8776 | **0.9208** | **0.9961** | 0.0033 | **0.9199** |

**Underlined value**: overall best performance

and the imbalance ratio in HAC_EasyEnsemble is set to 2.1:1).

It can be seen from Table 13 that the detection scheme PUMD proposed in this paper can achieve the best detection effect in most detection metrics. However, in the three metrics of Macro-Precision, Precision and FP rate, the performance of AULD is better than PUMD. Further analysis of other metrics of AULD, it is not difficult to find that its high Precision and low FP-Rate are mainly achieved at the expense of recall rate. In some comprehensive evaluation metrics, such as F1 and MCC, the performance of AULD scheme is poor. Therefore, we can conclude that AULD scheme lacks the ability to detect malicious domain names. Although we try our best to reproduce the relevant works according to the description of the original papers, we still find that there is a big gap in the detection effect between related works and the method proposed in this paper. We speculate that this may be due to the advantages of Feature Engineering, which makes the method proposed in this paper have better detection effect. The detection feature sets of relevant works are directly integrated according to the existing works, lacking innovation and pertinence. In addition, the scale of these feature sets is small, and the number ranges from 9 to 16, which can be regarded as a subset of the feature scheme adopted in this paper. Therefore, it is difficult to effectively distinguish malicious domains from benign domains by training classification models on these small-scale and lack of pertinence feature sets. In order to confirm this view, we design quantitative experiment 2.

#### *Quantitative experiment 2*

The common feature of these four related works is that the main contribution focuses on the construction of detection model, rather than the extraction of detection features. In order to confirm that the huge gap between PUMD and related works in detection performance is caused by feature engineering, we further designed quantitative experiment 2. Specifically, all the comparison schemes adopt the feature scheme proposed in this paper, and only the model training methods are compared. The experimental results are shown in Table 14.

It can be seen from the Table 14 that the detection effect of all related works have been greatly improved after adopting the feature scheme in this paper, which verifies the effectiveness of the feature scheme in this paper. At the same time, through further horizontal comparison of the detection effect of each evaluation metric, it can be seen that the PUMD scheme proposed in this paper maintains the optimal performace on most evaluation metrics, which confirms the effectiveness of the two-stage Pu learning scheme adopted by PUMD. AULD still leads the way in Macro-Precision, Precision and FP-Rate, but ranks fourth in the comprehensive evaluation metrics (F1 and MCC). Therefore, we can conclude that the malicious domain name detection ability of AULD is relatively poor.

In summary, through the above two quantitative experiments, it is fully confirmed that the PUMD scheme is superior to related works in both feature scheme and learning scheme, which further verifies the effectiveness of the proposed scheme in this paper.

## Discussion and future work

The purpose of this section is to make a qualitative comparison between the related works listed in "Background and motivation" section and PUMD, as shown in the Table 15, to further discuss the advantages of PUMD and the future direction of work.

### Qualitative comparison

**From the perspective of dataset**, PUMD uses comprehensive anonymization data, while most solutions do not consider the privacy of DNS traffic, such as Phoneix (Schiavoni et al. 2014), ELM (Shi et al. 2018), LSTM. MI(Tran et al. 2018), KSDom (Wang et al. 2020) , HAC_EasyEnsemble (Liu et al. 2018), which are not suitable for sensitive scene. Besides, PUMD only uses a small number of malicious domain labels manually annotated by security researchers, while existing works use the Alexa TOP sites as ground truth of benign domains, which are impure and will affect performance.

**Table 15** The properties of the malicious domain detection methods

| No. | Work | Object | Technique | Dataset | | Feature construction* | | | | | | Model training | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Data anonymize | Ground truth SRC | Handcraft | | | | Implicit | | Trainset size | | | Imbalance ratio |
| | | | | | | HC | T | W | E | IC | A | Mali | Benign | Unlabel | |
| 1 | PUMD(our) | Malicious activity DN(C&C) | PU learning: iForest + RF | DN, IP Addr | manual label | ✓ | ✓ | ✓ | ✓ | | ✓ | 100–861 | | 19651–20412 | 22.8–204.1 |
| 2 | Phoenix | DGA | mahalanobis distance + dbscan | | generate DN, blacklist, Alexa | ✓ | | | | | ✓ | ~ 100k | | | |
| 3 | AULD | Malicious activity DN | filter-rule + canopy + k-means | IP Addr | simulate DN, Alexa | ✓ | ✓ | ✓ | | | | | | | |
| 4 | HinDom | Malicious activity DN | HIN + transductive classify | IP Addr | blacklist, Alexa, whitelist | | | | | ✓ | ✓ | 0.02M–0.22M | 0.07M–0.63M | | 2.8 |
| 5 | ELM | Malicious activity DN | ELM | | blacklist, Alexa | ✓ | ✓ | ✓ | | | | ~ 20k | ~ 6k | | 0.3 |
| 6 | LSTM.MI | DGA | cost-sensitive lstm | | blacklist, Alexa | | | | | ✓ | | ~ 41 k (total) | ~ 44k | | 2–3534 |
| 7 | KSDom | Malicious activity DN | catboost+ kmSmote | | blacklist, Alexa, whitelist | ✓ | ✓ | ✓ | | | | 5.4k, 3.6k, 1.8k, 0.9k | 9k | | 1.6, 2.5, 5, 10 |
| 8 | HAC_Easy-Ensemble | Malicious activity DN | undersample +ensemble learning | | blacklist, Alexa | ✓ | ✓ | | | | | 2.7k | 5.76k | | 2.1 |

| No. | Work | Object | Technique | Model testing | | | Model output | |
|---|---|---|---|---|---|---|---|---|
| | | | | Testset size | | Imbalance ratio | Process | Class |
| | | | | Mali | Benign | | | |
| 1 | PUMD(our) | Malicious activity DN(C&C) | PU learning: iForest + RF | 91–852 | 19560 | 22.9–214.9 | Auto | 2 |
| 2 | Phoenix | DGA | mahalanobis distance + dbscan | ~ 1.3M | | | Threshold, Rule- match | 5 |
| 3 | AULD | Malicious activity DN | filter-rule + canopy + k-means | 1462 | 9068 | 6.2 | Threshold, Manual -analysis | 2 |
| 4 | HinDom | Malicious activity DN | HIN + transductive classify | ~ 0.25M | ~ 0.7M | 2.8 | Auto | 2 |
| 5 | ELM | Malicious activity DN | ELM | ~ 20k | ~ 6k | 0.3 | Auto | 2 |
| 6 | LSTM.MI | DGA | cost-sensitive lstm | ~ 41 k (total) | ~ 44k | 2–3534 | Auto | 38 |
| 7 | KSDom | Malicious activity DN | catboost+ kmSmote | 0.6k, 0.4k, 0.2k, 0.1k | 1k | 1.6,2.5, 5,10 | Auto | 2 |
| 8 | HAC_Easy Ensemble | Malicious activity DN | undersample +ensemble learning | 0.3k | 0.64k | 2.1 | Auto | 2 |

*HC: Handcraft Character, T: Traffic, W: Whois, E: Evidence, IC: Implicit Character, A: Associate

**From the perspective of feature construction**, PUMD comprehensively extracts multiple features. Among them, the character, traffic, and whois features are extracted from single domain, and the evidence features extracted based on resource overlapping association from domain association, which can also be considered as an implicit association feature, while existing works extract features from only one perspective, which is easy to be evaded.

**From the perspective of model training and testing**, compared with the existing supervised schemes, such as HinDom, ELM, PUMD reduce the labeling cost by half. Besides, other researches require at least one thousand domains to train the model, while the labeled samples in PUMD are kept between 100 and 861. In addition, this paper maintains a large imbalance ratio in both trainset and testset, and discusses the PUMD's ability to capture malicious domains in the real network environment. And **from the perspective of model output processing**, PUMD is an end-to-end framework that can automatically predict malicious domains without the need for manual analysis assistance or pre-set classification thresholds.

### Future direction

The future direction can be mainly divided into three aspects, including the design of whois data usage schemes, the optimization of the model, and the expansion of the experimental dataset.

**The design of whois data usage schemes:** We use whois information to extract three detection features in the feature engineering part, and associate the domains to be detected and the known malicious domains based on the whois information resource when extracting evidence features. Considering that whois information may not be released to the public in the future, this work cannot temporarily provide an effective solution to the problem that all relevant features of whois information will be invalid. However, our original feature set is huge, and there are some features in the original feature set that can make up for the impact of removing the whois information-related features and provide a robust malicious domain name detection capability. In addition, how to obtain more effective detection features to characterize malicious domain names is also one of our future work directions.

**The optimization of the model:** Considering PUMD is an basic framework for malicious domain detection, which supports the replacement of base classifiers with specific detection algorithms, we will try to integrate the existing solutions into the PUMD framework to further evaluate the generalization capability of PUMD. At the same time, for the problem that the existing detection features mostly use a statistical overview of traffic, but cannot fully characterize the activities of malicious domains, consider introducing time series features to model domain name cross-cycle activity changes. In addition, consider using a more complex graph network to model the association between the domain to be detected and the existing malicious domain family, and extract global evidence characteristics, so as to comprehensively capture the multiple resource reuse exceptions of the malicious domains.

**The expansion of the experimental dataset:** At present, due to the sensitivity of DNS data, there is no generally accepted large-scale malicious domain name detection dataset available in the field of malicious domain name detection. This paper chooses to conduct experiments on a small-scale open source DNS traffic dataset provided by Qi-ANXIN, in order to facilitate comparison with other schemes in the follow-up works. We will try to deploy the PUMD detection scheme in the real network in the future to test the model detection ability on real large-scale traffic.

## Conclusion

In this paper, we propose a malicious domain detection framework called PUMD. By adopting the two-step PU learning scheme, PUMD can use a small number of labeled malicious domains and a large number of unlabeled domains to construct a binary classifier. At the same time, PUMD employs novel feature engineering techniques, including evidence features based on resource overlapping and a feature selection strategy based on permutation importance and binning to improve malicious domain detection performance. This paper further conducts experiments to detect potential malicious C&C domains in open source real DNS data set provided by QI-ANXIN Technology Group. Experiments have separately proved that when the label information is extremely limited (100 labeled malicious domains), or the class category is severe imbalanced (Benign:Mali $\approx$ 200:1), and in the case of multiclass malicious domain activities(9 families), PUMD can maintain the optimal detection performance, which can solve the problems of insufficient credible label information, severe class imbalance, and incompact distribution of different malicious activities' domain samples in the malicious domain detection field.

## Appendix

**Evaluate ACC** (Fig. 5a). ACC is also not suitable for imbalanced data set. It only focuses on the number of samples that are correctly classified, and does not distinguish the

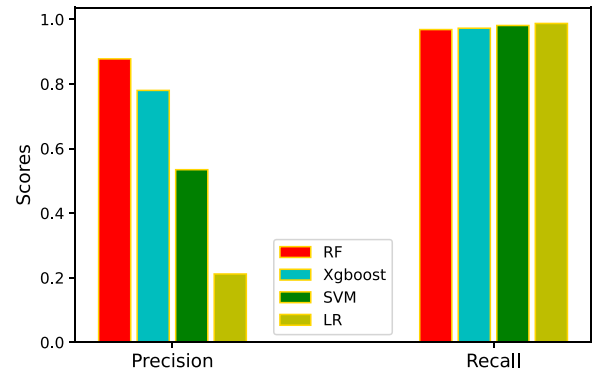
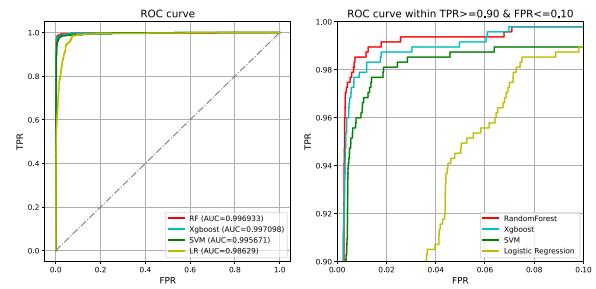



**Fig. 9** Precision & recall analysis




(a) ROC (b) ROC_local

**Fig. 10** ROC and AUC analysis. (**a**) Prediction ROC changes. (**b**) Prediction ROC changes within the TPR>=0.90 and FPR<=0.10 zone

categories of the samples, thus it will be biased to consider the performance of the majority category, that is the benign domains, which is inconsistent with the detection target of this paper.

**Evaluate precision and recall** (Fig. 5b–c). These two evaluation indicators are not suitable for analyzing imbalanced data set separately. Take the performance on label frequency c=0.5 as an example (see Fig. 9). LR has the worst performance on the Precision, while the best performance on Recall. This is because Precision only focuses on the ability to detect correctly, while Recall only pays attention to the ability to detect completely.

**Evaluate ROC and AUC** (Fig. 5e). ROC and AUC are not suitable for imbalanced data set, especially when the negative sample set is large, FPR grows slowly, and the ROC curve is generally close to the upper left corner. Taking the ROC curve on c = 0.5 as an example (see Fig. 10a), AUC actually contains a large area of no interest, such as the $FPR > 0.1$ area, where the number of FPs exceeds four times that of malicious domains. Only observe the $FPR \in [0, 0.1]$ interval (see Fig. 10b), the effective area is ranked as $RF > Xgboost > SVM > LR$, which is inconsistent with the AUC value ranking.

**Acknowledgements**
The authors would like to thank the QI-ANXIN Technology Group for open-source malicious domain dataset. We also thank the anonymous reviewers for their time and efforts in reviewing our manuscript and providing constructive comments.

**Author contributions**
ZF: investigation, conceptualization, methodology, materials, writing, editing, experiment, validation, review, resources, supervision. QW: resources, supervision, discussion. Haoran Jiao: discussion, conceptualization. JL: resources, discussion, supervision, funding acquisition. ZC: resources, discussion. SL: resources, discussion. YL: resources, discussion, review, supervision, funding acquisition. All authors read and approved the final manuscript.

**Funding**
This research is supported by National Key Research and Development Program of China (Nos. 2021YFF0307203, 2019QY1300), Youth Innovation Promotion Association CAS (No. 2021156), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDC02040100) and National Natural Science Foundation of China (No. 61802404). This work is also supported by the Program of Key Laboratory of Network Assessment Technology, the Chinese Academy of Sciences, Program of Beijing Key Laboratory of Network Security and Protection Technology.

**Availability of data and materials**
Online open-source malicious domain dataset: The dataset supporting the conclusions of this article is available in the DataCon Opendata repository, [http://datacon.qianxin.com/opendata/dns]. Online project source code: Project name: PUMD; Project home page: https://github.com/fzs-git/PUMD; Operating system: Platform independent; Programming language: Python; Other requirements: Python 3.8; License: FreeBSD; Any restrictions to use by non-academics: licence needed

## Declarations

**Competing interests**
The authors declare that they have no competing interests.

**Author details**
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. [2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China.

Received: 7 September 2021 Accepted: 27 March 2022
Published online: 01 October 2022

**References**

ALEXA-INTERNET: Alexa topsites (2021). https://www.alexa.com/topsites. Accessed 20 Aug 2021
Almashhadani AO, Kaiiali M, Carlin D, Sezer S (2020) Maldomdetector: a system for detecting algorithmically generated domain names with machine learning. Comput Secur 93:101787
Andre Correa: Malware Patrol (2021). https://www.malwarepatrol.net/. Accessed 20 Aug 2021
Antonakakis M, Perdisci R, Dagon D, Lee W, Feamster N (2010) Building a dynamic reputation system for DNS. In: USENIX security symposium, pp 273–290
Bekker J, Davis J (2020) Learning from positive and unlabeled data: a survey. Mach Learn 109(4):719–760
Bekker J, Robberechts P, Davis J (2019) Beyond the selected completely at random assumption for learning from positive and unlabeled data. In: Joint

European conference on machine learning and knowledge discovery in databases. Springer, pp 71–85
Cao Y, Han W, Le Y (2008) Anti-phishing based on automated individual white-list. In: Proceedings of the 4th ACM workshop on digital identity management, pp 51–60
Choi H, Lee H, Kim H (2009) BotGAD: detecting botnets by capturing group activities in network traffic. In: Proceedings of the fourth international ICST conference on COMmunication System softWAre and middlewaRE, pp 1–8
Choi H, Lee H, Lee H, Kim H (2007) Botnet detection by monitoring group activities in DNS traffic. In: 7th IEEE international conference on computer and information technology (CIT 2007). IEEE, pp 715–720
Curtin RR, Gardner AB, Grzonkowski S, Kleymenov A, Mosquera A (2019) Detecting DGA domains with recurrent neural networks and side information. In: Proceedings of the 14th international conference on availability, reliability and security, pp 1–10
Dhamnani S, Sinha R, Vinay V, Kumari L, Savova M (2021) Botcha: detecting malicious non-human traffic in the wild. arXiv preprint arXiv:2103.01428
Du Peng DS (2020) A DGA domain name detection method based on deep learning models with mixed word embedding. J Comput Res Dev 57(2):433
Elkan C, Noto K (2008) Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 213–220
Felegyhazi M, Kreibich C, Paxson V (2010) On the potential of proactive domain blacklisting. LEET 10:6
Hao S, Kantchelian A, Miller B, Paxson V, Feamster N (2016) Predator: proactive recognition and elimination of domain abuse at time-of-registration. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 1568–1579
He W, Gou G, Kang C, Liu C, Li Z, Xiong G (2019) Malicious domain detection via domain relationship and graph models. In: 2019 IEEE 38th international performance computing and communications conference (IPCCC). IEEE, pp 1–8
Huang S-Y, Mao C-H, Lee H-M (2010) Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In: Proceedings of the 5th ACM symposium on information, computer and communications security, pp 101–111
Kang J, Lee D (2007) Advanced white list approach for preventing access to phishing sites. In: 2007 international conference on convergence information technology (ICCIT 2007). IEEE, pp 491–496
Lee WS, Liu B (2003) Learning with positive and unlabeled examples using weighted logistic regression. In: ICML, vol 3, pp 448–455
Liu FT, Ting KM, Zhou Z-H (2008) Isolation forest. In: 2008 eighth IEEE international conference on data mining. IEEE, pp 413–422
Liu Z, Zeng Y, Zhang P, Xue J, Zhang J, Liu J (2018) An imbalanced malicious domains detection method based on passive DNS traffic analysis. Secur Commun Netw 2018:1–7
Luo Y, Cheng S, Liu C, Jiang F (2018) PU learning in payload-based web anomaly detection. In: 2018 third international conference on security of smart cities, industrial control system and communications (SSIC). IEEE, pp 1–5
Ma J, Saul LK, Savage S, Voelker GM (2009) Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1245–1254
Ma X, Zhang J, Tao J, Li J, Tian J, Guan X (2014) DNSRadar: outsourcing malicious domain detection based on distributed cache-footprints. IEEE Trans Inf Forensics Secur 9(11):1906–1921
Morales JA, Al-Bataineh A, Xu S, Sandhu R (2009) Analyzing DNS activities of bot processes. In: 2009 4th international conference on malicious and unwanted software (MALWARE). IEEE, pp 98–103
Phishtank: PhishTank open api (2021). https://www.phishtank.com. Accessed 20 Aug 2021
Prieto I, Magaña E, Morató D, Izal M (2011) Botnet detection based on DNS records and active probing. In: Proceedings of the international conference on security and cryptography. IEEE, pp 307–316
QI-ANXIN: DataCon opendata (2020). https://datacon.qianxin.com/opendata/dns. Accessed 20 Feb 2007
Sato K, Ishibashi K, Toyono T, Hasegawa H, Yoshino H (2012) Extending black domain name list by using co-occurrence relation between DNS queries. IEICE Trans Commun 95(3):794–802
Schiavoni S, Maggi F, Cavallaro L, Zanero S (2014) Phoenix: DGA-based botnet tracking and intelligence. In: International conference on detection of intrusions and malware, and vulnerability assessment. Springer, pp 192–211
Schüppen S, Teubert D, Herrmann P, Meyer U (2018) {FANCI}: Feature-based automated NXDomain classification and intelligence. In: 27th {USENIX} security symposium ({USENIX} security 18), pp 1165–1181
Shi Y, Chen G, Li J (2018) Malicious domain name detection based on extreme machine learning. Neural Process Lett 48(3):1347–1357
Stevanovic M, Pedersen JM, D'Alconzo A, Ruehrup S, Berger A (2015) On the ground truth problem of malicious DNS traffic analysis. Comput Secur 55:142–158
Sun L, Wei X, Zhang J, He L, Philip SY, Srisa-an W (2017) Contaminant removal for android malware detection systems. In: 2017 IEEE international conference on big data (big data). IEEE, pp 1053–1062
Sun X, Tong M, Yang J, Xinran L, Heng L (2019) {HinDom}: a robust malicious domain detection system based on heterogeneous information network with transductive classification. In: 22nd international symposium on research in attacks, intrusions and defenses ({RAID} 2019), pp 399–412
SURBL.ORG: SURBL-URI Reputation Data (2021). http://www.surbl.org/. Accessed 20 Aug 2021
The Spamhaus Project Ltd: The Domain Block List (2021). https://www.spamhaus.org/dbl/. Accessed 20 Aug 2021
Tong V, Nguyen G (2016) A method for detecting DGA botnet based on semantic and cluster analysis. In: Proceedings of the seventh symposium on information and communication technology, pp 272–277
Tran D, Mac H, Tong V, Tran HA, Nguyen LG (2018) A LSTM based framework for handling multiclass imbalance in DGA botnet detection. Neurocomputing 275:2401–2413
Van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(11):2579–2605
Villamarín-Salomón R, Brustoloni JC (2009) Bayesian bot detection based on dns traffic similarity. In: Proceedings of the 2009 ACM symposium on applied computing, pp 2035–2041
Wang Q, Li L, Jiang B, Lu Z, Liu J, Jian S (2020) Malicious domain detection based on k-means and smote. In: International conference on computational science. Springer, pp 468–481
Wu S, Fulton J, Liu N, Feng C, Zhang L (2019) Risky host detection with bias reduced semi-supervised learning. In: Proceedings of the 2019 international conference on artificial intelligence and computer science, pp 34–40
Yan G, Li Q, Guo D, Li B (2019) AULD: large scale suspicious DNS activities detection via unsupervised learning in advanced persistent threats. Sensors 19(14):3180
Zhang Y-L, Li L, Zhou J, Li X, Liu Y, Zhang Y, Zhou Z-H (2017) POSTER: a PU learning based system for potential malicious URL detection. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 2599–2601
Zhou Y, Li Q-S, Miao Q, Yim K (2013) DGA-based botnet detection using DNS traffic. J Internet Serv Inf Secur 3(3/4):116–123

## Publisher's Note