

An Efficient Probabilistic Supervised Learning Model For Realtime Iot Attack Detection in Software-Defined Iot (SD-Iot) Networks

K.S.Niraja¹, Dr. Sabbineni Srinivasa Rao²

¹Research Scholar in department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, vaddeswaram,A.P, India, email: niraja.ksvce@gmail.com

²Deputy Registrar, Associate Professor in Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, vaddeswaram,A.P, India. Email: srinu1479cse@kluniversity.in

Abstract

As the size of an imbalanced database increases, the true positive rate of traditional prediction algorithms faces difficulties in improvement due to the high ratio of majority to minority classes and the substantial noise present in the data. Missing values, attribute noise, and imbalanced classes are some of the significant factors that can impact the quality of input data. The quality of imbalanced data significantly impacts the efficiency of classification approaches, making it necessary to ensure high-quality input data to achieve optimal results. Therefore, to ensure high-quality predictions on imbalanced datasets, traditional machine learning models need to be optimized. In this paper, a filtering approach, feature ranking, clustering, and ensemble classification model are proposed to address these issues in imbalanced SD-IoT datasets. A novel strategy is proposed to handle missing data, imbalanced classes, feature selection, probabilistic clustering, and ensemble classification approaches are optimized to improve the true positive rate and error rate on imbalanced SD-IoT databases. This work implements a hybrid probabilistic class membership and optimized Max-Hellinger based classification framework on imbalanced SD-IoT datasets to improve overall statistical metrics. Experimental results have demonstrated that the proposed approach outperforms conventional privacy-preserving techniques in terms of statistical metrics.

Keywords: Imbalance SD-IoT dataset, probabilistic clustering , support vector machine, decision tree, ensemble learning model.

1.Introduction

Classification is a critical task in machine learning and data mining that involves grouping objects into pre-defined categories based on their attributes. Numerous real-world applications, such as image and text classification, and medical diagnosis, rely on this fundamental task. However, the classification process becomes challenging when the data is imbalanced, and the current models often produce unreliable results. This is primarily due to the under-representation and reduced importance of the minority class in imbalanced datasets, which can negatively affect the model's accuracy [1]. Class imbalance occurs when one class has significantly more instances than the other class. Real-world applications such as credit scoring, fraud detection, and medical diagnostics frequently involve this type of imbalanced dataset [2-5]. On the other hand, class distribution imbalance describes a scenario where the number of instances of each class is not significantly different, but the distribution of the classes is unequal. This type of imbalanced dataset is also common in real-world applications like picture and text categorization. Additionally, concept drift is a situation where the distribution of classes changes over time, making it difficult for machine learning models to adapt. Covariate shift occurs when the distribution of the input features varies for each class. This type of imbalanced dataset arises when data is collected from multiple sources or at different times, making it challenging for machine learning models to accurately predict the class labels. Imbalanced datasets in machine learning algorithms can lead to poor accuracy, biased outputs, and overfitting. Existing models often overlook the minority class in imbalanced datasets, resulting in subpar performance and distorted outcomes [6]. Ensemble learning, which combines multiple machine learning models to produce a single prediction, is a useful method for improving the performance of machine learning models. Although this approach can be computationally expensive and requires significant resources, it has shown promising outcomes in practical applications [7]. Random oversampling involves replicating instances of the minority class at random until the class distribution is balanced. While this approach is straightforward and easy to understand, it can lead to overfitting and an increase in computation time. Synthetic oversampling involves creating synthetic instances of the minority class. Using techniques like bootstrapping and bagging [8], both oversampling techniques have their limitations and potential drawbacks. The appropriate technique depends on the specific problem and data characteristics. Nevertheless, these resampling techniques have been shown to effectively mitigate class imbalance in machine learning classification models. Imbalanced datasets in real-time machine learning systems can result in several issues, including predictive model bias, overfitting, and metric misrepresentation. Sampling techniques, such as oversampling or undersampling, can balance the class distribution in an imbalanced dataset, but they may also

result in information loss and an increase in noise in the data [9]. Preparing a dataset for machine learning involves several crucial processes, including data preprocessing, noise filtering, addressing missing values, and handling attribute noise and class noise. This may involve handling missing values, scaling features to the same range, and converting data into a numerical representation [10].

Classification assigns items to fixed categories based on their attributes. It's used in areas like image labeling, document tagging, and medical evaluations. Uneven class distribution makes classification harder. When one class appears far less often, models often miss it, leading to lower accuracy. In medical datasets, for instance, rare disease cases are often underrepresented. This can cause models to favor the more common class, producing skewed results. Strategies like oversampling, undersampling, or assigning class-specific costs have been used. But these can lead to overfitting or loss of useful data, and they often demand heavy computational resources.

Types of Imbalance

- **Class Imbalance:** One class has many more instances than the other. Found in areas like fraud detection and health diagnostics.
- **Distribution Imbalance:** Classes are not too different in number, but the imbalance still affects learning, common in visual or text tasks.
- **Concept Drift:** Class distributions shift over time. It affects dynamic environments like market trends or weather predictions.
- **Covariate Shift:** Input features differ between classes. It often happens when data comes from various sources or time periods.

Let:

- α : Majority class set, $|\alpha| = m$
- β : Minority class set, $|\beta| = n$, where $n \ll m$
- $\gamma = \alpha \cup \beta$: Initial training set
- ϕ : Oversampling function, $\phi: \beta \rightarrow \beta'$
- ψ : Classifier trained on γ
- ψ' : Classifier trained on $\gamma' = \alpha \cup \beta'$
- Feature space: $\xi \in \mathbb{R}^d$
- Label space: $\zeta = \{0, 1\}$
- Learning function: $\Psi: \xi \rightarrow \zeta$
- Loss function: $\mathcal{L}(\Psi)$
- True distribution: $\delta(x, y)$
- Empirical risk: $\mathcal{R}_{\text{emp}}(\Psi) = \frac{1}{|\gamma|} \sum_{(x_i, y_i) \in \gamma} \ell(\Psi(x_i), y_i)$

Step 1: Dataset imbalance

We start with class distributions:

$$|\alpha| = m, |\beta| = n, \text{ where } n \ll m$$

This implies that during learning, Ψ minimizes:

$$\mathcal{R}_{\text{emp}}(\Psi) = \frac{1}{m+n} \left(\sum_{x \in \alpha} \ell(\Psi(x), 0) + \sum_{x \in \beta} \ell(\Psi(x), 1) \right)$$

Due to $m \gg n$, the model is biased toward class 0.

Step 2: Define oversampling strategy

We introduce a mapping $\phi: \beta \rightarrow \beta'$, such that:

$$|\beta'| = m \Rightarrow |\gamma'| = 2m$$

Now the empirical risk becomes:

$$\mathcal{R}'_{\text{emp}}(\Psi') = \frac{1}{2m} \left(\sum_{x \in \alpha} \ell(\Psi'(x), 0) + \sum_{x \in \beta'} \ell(\Psi'(x), 1) \right)$$

Step 3: Random Oversampling ϕ_r

Randomly choose elements:

$$\beta' = \beta \cup \{x_i \in \beta \mid i = 1, \dots, (m-n)\}$$

Drawbacks:

- $\exists x_i, x_j \in \beta': x_i = x_j \Rightarrow$ high variance in Ψ'
- Increases empirical risk variance:

$$\text{Var}(\mathcal{R}'_{emp}) \uparrow$$

Step 4: Synthetic Oversampling ϕ_s

Let:

$$x_i, x_j \in \beta, \lambda \sim U(0,1) \\ x_{syn} = x_i + \lambda(x_j - x_i)$$

Then:

$$\beta' = \beta \cup \{x_{syn}^k\}_{k=1}^{m-n}$$

Advantages:

- Introduces diversity in β'
- Reduces overfitting by smoothing the decision boundary

Step 5: Adaptive Synthetic Oversampling ϕ_a

Let:

- η_i : Density score of $x_i \in \beta$ based on K-nearest neighbors from $\alpha \cup \beta$

Generate:

$$x_{syn} = x_i + \lambda(x_j - x_i) \text{ where } x_j \in \text{neighbors}(x_i), \lambda \in (0,1)$$

Weighting function:

$$w(x_i) = \frac{\eta_i}{\sum \eta_i}$$

Generate $\phi_a(\beta)$ such that:

Harder-to-learn x_i generate more x_{syn}

Step 6: Regularization Term in Loss

We incorporate regularization $\Omega(\Psi')$ into loss:

$$\mathcal{L}_{total} = \mathcal{R}'_{emp}(\Psi') + \lambda\Omega(\Psi')$$

Where $\Omega(\Psi')$ controls model complexity. Oversampling influences this by affecting:

- Margin width
- Boundary smoothness

Step 7: Bias-Variance Trade-off

Let:

- Bias: $\text{Bias}^2(\Psi')$
- Variance: $\text{Var}(\Psi')$

Goal: Reduce bias toward α without inflating variance too much.

Oversampling:

- Reduces bias (more balanced training)
- Random/synthetic may increase variance
- Adaptive synthetic attempts optimal trade-off

Step 8: Generalization Bound

Using VC dimension h , and number of samples N , we apply:

$$\mathbb{P}(|\mathcal{R}(\Psi') - \mathcal{R}'_{emp}(\Psi')| > \epsilon) \leq 2\exp\left(-\frac{2N\epsilon^2}{(1+h)^2}\right)$$

More balanced $\gamma' \Rightarrow N \uparrow \Rightarrow \epsilon \downarrow$

Thus, oversampling improves generalization.

Step 9: Decision Boundary Behavior

Let:

$$B(\Psi) = \text{decision boundary}$$

For imbalanced γ , boundary $B(\Psi)$ is skewed toward β .

For γ' , $B(\Psi')$ is more centered, with:

$$\text{Margin width} \uparrow \Rightarrow \text{Generalization} \uparrow$$

Step 10: Class-wise Precision & Recall

Let:

$$\text{Recall}_{\beta}(\Psi') > \text{Recall}_{\beta}(\Psi) \\ \text{Precision}_{\beta}(\Psi') \approx \text{Precision}_{\beta}(\Psi)$$

Oversampling mainly boosts recall, slightly affecting precision.

Oversampling is a technique for handling imbalanced datasets by duplicating instances of the minority class to balance the distribution of the classes. This approach aims to increase the number of instances of the minority class so that the machine learning model has a better representation of the minority class. There are several

oversampling methods, including random oversampling, synthetic oversampling, and adaptive synthetic oversampling[10].

Random oversampling involves randomly duplicating instances of the minority class until the distribution of the classes is balanced. This approach is simple and straightforward, but it can lead to overfitting and an increase in the computation time. Synthetic oversampling involves generating synthetic instances of the minority class using techniques such as bootstrapping and bagging. This approach is effective in increasing the number of instances of the minority class, but it can also lead to overfitting and an increase in the computation time. Adaptive synthetic oversampling involves generating synthetic instances of the minority class based on the distribution of the input features. This approach is effective in reducing the risk of overfitting and improving the performance of the machine learning model. The synthetic instances are generated using techniques such as k-nearest neighbors and decision trees[11-13].

There are several issues associated with imbalanced datasets in real-time machine learning applications:

Predictive Model Bias: Imbalanced datasets can result in biased machine learning models that tend to favor the majority class. This can lead to poor performance on the minority class and result in a higher number of false negatives, or missed predictions.

Overfitting: Machine learning algorithms trained on imbalanced datasets may overfit to the majority class, resulting in poor generalization performance and poor accuracy on unseen data.

Metric Misrepresentation: Common performance metrics, such as accuracy, can be misleading in imbalanced datasets, as they tend to favor the majority class. Alternative metrics, such as F1-score or the area under the receiver operating characteristic (ROC) curve, may be more appropriate for evaluating the performance of machine learning models in imbalanced datasets.

Data Sampling Techniques: Sampling techniques, such as oversampling or undersampling, can be used to balance the class distribution in an imbalanced dataset, but these techniques can also result in a loss of information and increased noise in the data[14].

Data Generation Techniques: Synthetic data generation techniques, such as SMOTE (Synthetic Minority Over-sampling Technique), can be used to generate synthetic samples of the minority class, but these techniques can also result in overfitting and decreased model performance.

These are some of the key issues associated with imbalanced datasets in real-time machine learning applications, and it is important to carefully consider these issues and take appropriate steps to address them when building and deploying machine learning models in real-time applications.

Data preprocessing, noise filtering, handling missing values, and dealing with attribute noise and class noise are important steps in preparing a dataset for machine learning.

Data Preprocessing: Data preprocessing refers to the cleaning and transforming of raw data into a format that can be used for machine learning. This may involve converting data into a numerical format, scaling features to the same range, and handling missing values.

Noise Filtering: Noise filtering involves removing or correcting instances in the dataset that contain errors or inconsistencies, such as outliers or duplicate instances. This step can help to improve the performance of machine learning algorithms and prevent overfitting.

Handling Missing Values: Missing values can occur when instances in the dataset have missing or incomplete feature values. Handling missing values can involve either imputing the missing values with estimates (such as the mean or median) or removing instances with missing values entirely.

Attribute Noise: Attribute noise refers to errors or inconsistencies in the features of a dataset. This can occur when features are measured or recorded inaccurately, or when they contain irrelevant information. Attribute noise can be reduced by removing irrelevant features, correcting errors in the data, or normalizing features to the same scale

Let us define the original dataset $\gamma = \{(\xi_i, \zeta_i)\}_{i=1}^{m+n}$ over a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where $\xi_i \in \mathbb{R}^d \subset \mathcal{X}$, $\zeta_i \in \{0,1\}$, and $\gamma = \alpha \cup \beta$, such that $|\alpha| = m \gg n = |\beta|$. The dataset is sampled from an unknown distribution $\mathbb{P}_{\xi, \zeta}$ with class imbalance.

We define a transformation $\phi: \mathcal{X}^N \rightarrow \mathcal{X}^{N'}$ with sub-operators:

- $\phi_o: \beta \rightarrow \beta'$, oversampling such that $|\beta'| \approx |\alpha|$,
 - $\phi_u: \alpha \rightarrow \alpha'$, undersampling such that $|\alpha'| \approx |\beta|$,
- and

$$\gamma' = \phi(\gamma) = \phi_o(\beta) \cup \phi_u(\alpha)$$

Introduce a synthetic function $\theta: \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{X}$ defined by:

$$\theta(\xi_i, \xi_j; \lambda) = \xi_i + \lambda(\xi_j - \xi_i), \lambda \sim \mathcal{U}(0,1)$$

Extended to $\theta_k: \mathcal{X} \rightarrow \mathcal{X}$ via interpolation among k -nearest neighbors:

$$\theta_k(\xi_i) = \frac{1}{k} \sum_{j=1}^k \theta(\xi_i, \xi_j; \lambda_j)$$

which generates a synthetic class $\beta_{syn} = \{\theta_k(\xi_i) \mid \xi_i \in \beta\}$, such that:

$$\gamma_{aug} = \alpha \cup \beta \cup \beta_{syn}$$

Define the learning model as a functional operator:

$$\Psi: \mathcal{X} \rightarrow \mathcal{Y}, \Psi \in \mathcal{H}_K \text{ (RKHS space)}$$

Let $\mathcal{L}_{emp}(\Psi, \gamma_{aug}) = \frac{1}{|\gamma_{aug}|} \sum_{i=1}^{|\gamma_{aug}|} \ell(\Psi(\xi_i), \zeta_i)$, where ℓ is a convex surrogate loss.

Overfitting is characterized by:

$$\mathbb{E}_{(\xi, \zeta) \sim \mathbb{P}}[\ell(\Psi(\xi), \zeta)] \gg \mathcal{L}_{emp}(\Psi, \gamma_{aug}) \Rightarrow \text{High generalization error}$$

Introduce noise $\epsilon_i \sim \mathcal{D}_{noise} \subset \mathbb{R}^d$, and define:

$$\mathcal{N}(\xi_i) = \xi_i + \epsilon_i$$

Noise filtering operator $\mathcal{F}_N: \mathcal{X}^N \rightarrow \mathcal{X}^{N''}$, where:

$$\mathcal{F}_N(\gamma) = \{(\xi_i, \zeta_i) \in \gamma \mid \|\epsilon_i\|_2 \leq \tau\}$$

Let missing values be indicated by function $\mu: \xi_i \rightarrow \{0, 1\}^d$, with $\mu_j(\xi_i) = 1 \Rightarrow \text{missing value}$. Define the imputation operator:

$$\mathcal{I}(\xi_i) = (\tilde{\xi}_{i1}, \dots, \tilde{\xi}_{id}), \tilde{\xi}_{ij} = \begin{cases} \xi_{ij} & \text{if } \mu_j(\xi_i) = 0 \\ \mathbb{E}[\xi_{\cdot j}] & \text{if } \mu_j(\xi_i) = 1 \end{cases}$$

Now, feature normalization is applied:

$$\mathcal{S}: \xi \mapsto \xi', \mathcal{S}(\xi_j) = \frac{\xi_j - \min(\xi_j)}{\max(\xi_j) - \min(\xi_j)}$$

Define total preprocessing pipeline $\mathcal{P} = \mathcal{S} \circ \mathcal{I} \circ \mathcal{F}_N$, such that:

$$\gamma_{clean} = \mathcal{P}(\gamma)$$

We now minimize regularized risk:

$$\Psi^* = \arg \min_{\Psi \in \mathcal{H}_K} [\mathcal{L}_{emp}(\Psi, \gamma_{clean}) + \lambda \|\Psi\|_{\mathcal{H}_K}^2]$$

The Rademacher complexity $\mathfrak{R}_n(\mathcal{H}_K)$ is reduced by clean preprocessing, and this directly increases the model's generalization margin γ_{gen} .

We evaluate via:

- Precision $P = \frac{TP}{TP+FP}$
- Recall $R = \frac{TP}{TP+FN}$
- F1 score: $\mathcal{F}_1 = \frac{2PR}{P+R}$
- AUC: $\mathcal{R}_{auc} = \int_0^1 TPR(x) dx$

For real-time models, redefine:

$$\phi_t, \theta_t: \mathcal{X}_t \rightarrow \mathcal{X}'_t, \text{ evolving adaptively with data stream over } t \in \mathbb{R}_+$$

Then final model optimization becomes:

$$\Psi_t^* = \arg \min_{\Psi} [\mathcal{L}(\Psi(\mathcal{P}(\phi_t(\gamma_t)))) + \lambda_t \Omega(\Psi)]$$

under non-stationary distribution shift: $\mathbb{P}_t(\xi, \zeta) \neq \mathbb{P}_{t+1}(\xi, \zeta)$.

This article is presented in the following manner. Section 2 contains an overview of the related work. Section 3 discusses the proposed work technique after that. The experimental results of the effectiveness of the proposed work are presented in Section 4. Finally, section 5 has the conclusion.

• Literature Survey

The problem of imbalanced data has led some to tackle it by assigning varying costs to classification errors or utilizing sampling techniques to balance class distribution. Although these methods can be advantageous, they do not always lead to improved model performance. Fortunately, a new active learning framework has emerged as a more efficient and effective solution[15]. By prioritizing high prediction accuracy above overall classification accuracy, this framework addresses the prevalent issue of misclassification of minority classes that frequently hampers machine learning algorithms. To improve the effectiveness of classifiers, one promising approach is to assign greater weight to positive class instances compared to negative ones. For instance, in the domain of cancer detection, misclassifying non-cancerous cells may require further medical procedures, but misidentifying cancerous cells can pose significant health risks. Similarly, in the realm of monitoring video feeds for suspicious activities, misclassifying a regular event may only result in an increase in security measures, while misclassifying a life-threatening event can have catastrophic ramifications. As such, it is crucial to ensure that the classification

model is designed to prioritize the accurate identification of instances within the minority class. However, this strategy requires careful calibration of penalty parameters to deal with misclassification issues. Another method involves adjusting the training dataset by either increasing underrepresented classes or decreasing overrepresented ones until a balanced class distribution is achieved. However, both techniques involve additional computational costs for data preprocessing, and oversampling can be particularly challenging for large datasets[16]. Undersampling has been suggested as a potential solution to enhance classifier sensitivity, but it may result in the loss of some data. Therefore, the choice of the most appropriate method should be based on a careful consideration of the specific dataset and the goals of the analysis. To enhance the learning process, it is essential to identify data that has the potential to be useful. However, removing irrelevant information during undersampling can lead to a significant decrease in prediction accuracy. Although a flexible ensemble learning approach has been suggested in [16] to tackle the issue of discarding redundant examples, its application is restricted to certain algorithm types as it lacks an external preprocessing stage[17]. On the other hand, oversampling involves generating synthetic positive instances from existing samples to increase class representation. In datasets with class imbalance, the most prominent characteristic is the skewed distribution of data. The effectiveness of a classification model is pivotal to its success, but this can be restricted by a limited sample size when addressing class imbalance. Referring to Ref. 40, it is evident that relying on patterns discovered in a small class can be hazardous. Therefore, the sample size is an indispensable factor in determining the model's ability to handle class imbalance. Augmenting the training set's size can considerably enrich the model's accuracy, specifically in identifying patterns in the small class[18]. Classification modeling is a practical approach to address the challenge of imbalanced class distribution by distinguishing rare and common samples. They suggest utilizing a significant dataset to offset the imbalance, given that the data is readily available and the learning time required for a large dataset is feasible. However, in some classification scenarios, a single class may contain diverse subclasses or subconcepts with varying numbers of instances, collectively referred to as within-class concepts. These subclasses can add an extra layer of complexity to classification models, posing additional challenges. Managing within-class imbalance is a complex and challenging task that requires addressing the uneven distribution of classes among subclasses. This type of imbalance can worsen the already existing distribution problem within and between classes, leading to two significant implications. Firstly, it increases the complexity of understanding the concepts from the dataset, and secondly, it may not always be apparent. Typically, decision trees identify the class label associated with a leaf by analyzing the training cases covered by it and selecting the most frequent class[19]. However, modifying decision trees may be necessary to overcome class imbalance issues and improve the accuracy of the classification model. To ensure the small classes are not overlooked in the learning process, it is crucial to prioritize the development of tests that can effectively distinguish between the small and large classes. This is because there are instances where the learning process may terminate prematurely, before identifying the branches that predict the small classes, or these branches may be pruned due to their vulnerability to overfitting. The reason for this is that accurately predicting a small number of samples from the small classes has a negligible impact on reducing the error rate, compared to the error rate surge caused by overfitting[20]. As a result, relying solely on pruning based on prediction error may lead to the small classes being marginalized in the learning process. Refining the model by removing a branch that predicts a minor class is akin to the delicate pruning of a tree, resulting in a thriving leaf node with a prominent label. The multi-layer perceptron (MLP) trained by the backpropagation (BP) algorithm is a formidable instrument for resolving classification issues, boasting a complex neural network composed of interlinked layers of neurons. Each neuron is a crucial component in the intricate machinery, linked to every other neuron in the preceding and subsequent layers, with the input layer serving as the driving force. Although the input layer's neurons lack activation functions, the remaining neurons are poised for action, equipped with the ability to classify and categorize with remarkable precision[21].

Let $\gamma = \{(\xi_i, \zeta_i)\}_{i=1}^N$ denote a high-dimensional clinical dataset, where each patient record $\xi_i \in \mathbb{R}^d$ encodes diagnostic measurements (lab results, vitals, imaging-derived features), and $\zeta_i \in \{0,1\}$ denotes disease status (e.g., presence of a rare pathology). Let this dataset be drawn i.i.d. from an underlying joint distribution $\mathbb{P}_{\xi, \zeta}$ over $\mathcal{X} \times \mathcal{Y}$, with significant class imbalance:

$$\gamma = \alpha \cup \beta, |\alpha| \gg |\beta|, \text{ where } \beta = \{(\xi_i, \zeta_i = 1)\}, \text{ rare disease cases}$$

We define the following operators and mappings:

Sampling Operator

The sampling transformation $\phi: \gamma \mapsto \gamma'$, consists of:

- Oversampling minority $\phi_o(\beta) \rightarrow \beta'$, such that $|\beta'| \rightarrow |\alpha|$
- Undersampling majority $\phi_u(\alpha) \rightarrow \alpha'$, such that $|\alpha'| \rightarrow |\beta|$

Balanced dataset:

$$\gamma' = \phi(\gamma) = \phi_o(\beta) \cup \phi_u(\alpha)$$

Synthetic Clinical Case Generator θ

Let $\theta: \mathcal{X}^2 \times [0,1] \rightarrow \mathcal{X}$ represent a clinical variant of SMOTE:

$$\theta(\xi_i, \xi_j; \lambda) = \xi_i + \lambda(\xi_j - \xi_i), \lambda \sim \mathcal{U}[0,1]$$

To model local manifold behavior of rare cases, define kernel-averaged neighbors:

$$\theta_k(\xi_i) = \frac{1}{k} \sum_{j=1}^k \theta(\xi_i, \xi_j; \lambda_j), \xi_j \in \mathcal{N}_k(\xi_i)$$

Augmented set becomes:

$$\gamma_{syn} = \alpha \cup \beta \cup \{(\theta_k(\xi_i), 1)\}_{\xi_i \in \beta}$$

Noise Operator $\mathcal{N}$

Real-world sensors (ECG, MRI, etc.) introduce input perturbations:

$$\mathcal{N}(\xi_i) = \xi_i + \epsilon_i, \epsilon_i \sim \mathcal{D}_{noise} \subset \mathbb{R}^d$$

We classify noise by origin:

- $\epsilon_i^{(a)}$: attribute distortion
- $\epsilon_i^{(c)}$: class labeling error
- $\epsilon_i^{(m)}$: machine drift or channel loss

Filtering threshold $\tau \in \mathbb{R}_+$ applied through:

$$\mathcal{F}_{\mathcal{N}}(\gamma) = \{(\xi_i, \zeta_i) \in \gamma \mid \|\epsilon_i\|_2 \leq \tau\}$$

Missing Feature Modeling μ

Let missingness be indicator-valued:

$$\mu: \mathcal{X} \rightarrow \{0,1\}^d, \mu_j(\xi_i) = \begin{cases} 1 & \text{if feature } j \text{ is missing in } \xi_i \\ 0 & \text{otherwise} \end{cases}$$

Imputation operator $\mathcal{I}$ defined as:

$$\mathcal{I}_j(\xi_i) = \begin{cases} \xi_{ij} & \text{if } \mu_j(\xi_i) = 0 \\ \mathbb{E}[\xi_{\cdot j}] & \text{if MCAR (missing completely at random)} \\ f(\xi_{\cdot \setminus j}) & \text{otherwise} \end{cases}$$

Normalization $\mathcal{S}$

Standard scaling across clinical features:

$$\mathcal{S}_j(\xi_i) = \frac{\xi_{ij} - \min(\xi_{\cdot j})}{\max(\xi_{\cdot j}) - \min(\xi_{\cdot j})}$$

Full preprocessing pipeline:

$$\mathcal{P} = \mathcal{S} \circ \mathcal{I} \circ \mathcal{F}_{\mathcal{N}}, \gamma_{clean} = \mathcal{P}(\gamma)$$

Functional Learning Model Ψ

Let $\Psi: \mathcal{X} \rightarrow \mathcal{Y}$ be a classifier in a RKHS space $\mathcal{H}_K$ with kernel K , trained on clean data:

$$\Psi^* = \arg \min_{\Psi \in \mathcal{H}_K} \left[\frac{1}{N'} \sum_{i=1}^{N'} \ell(\Psi(\xi_i), \zeta_i) + \lambda \|\Psi\|_{\mathcal{H}_K}^2 \right]$$

Generalization Analysis

Using empirical Rademacher complexity $\mathfrak{R}_n$, we ensure:

$$\mathbb{E}[\ell(\Psi(\xi), \zeta)] \leq \mathcal{L}_{emp}(\Psi) + 2\mathfrak{R}_n(\mathcal{H}_K) + \sqrt{\frac{\log(1/\delta)}{2n}}$$

As γ_{clean} becomes denoised and balanced, $\mathfrak{R}_n \downarrow$, reducing overfitting and increasing margin.

Model Evaluation and Confidence

Let:

$$\mathcal{F}_1 = \frac{2PR}{P+R}, AUC = \int_0^1 TPR(FPR^{-1}(x))dx$$

For probabilistic calibration:

$$\mathcal{H}(p) = - \sum_{c=1}^C p_c \log p_c, \text{Entropy of output confidence}$$

Time-Adaptive Medical System

Let $t \in \mathbb{R}_+$, for streaming data from IoMT devices:

$$\phi_t, \theta_t, \mathcal{P}_t \text{ adapted over time windows } [t, t + \Delta]$$

Real-time model:

$$\Psi_t^* = \arg \min_{\Psi} [\mathcal{L}(\Psi(\mathcal{P}_t(\phi_t(\gamma_t)))) + \lambda_t \Omega(\Psi)]$$

Under concept drift:

$$\mathbb{P}_t(\xi, \zeta) \neq \mathbb{P}_{t+1}(\xi, \zeta)$$

Dynamic retraining via exponential weighting:

$$\Psi_t = \gamma \Psi_{t-1} + (1 - \gamma) \Psi_t^{new}, \gamma \in [0, 1]$$

Let $\gamma = \{(\xi_i, \zeta_i)\}_{i=1}^N$ denote a clinical/surveillance dataset drawn from a data space $\mathcal{X} \times \mathcal{Y}$, where:

- $\xi_i \in \mathbb{R}^d$ is the feature vector,
- $\zeta_i \in \{0, 1\}$, with $\zeta_i = 1$ indicating a critical class (e.g., malignant tumor, threat event),
- Let $\beta = \{i \mid \zeta_i = 1\}$, the minority class, and $\alpha = \{i \mid \zeta_i = 0\}$, the majority class.

Class Distribution Imbalance

$$|\beta| \ll |\alpha| \Rightarrow \gamma = \alpha \cup \beta$$

Cost-sensitive Risk Formulation

Define a model $\Psi: \mathcal{X} \rightarrow \mathcal{Y}$ with surrogate loss ℓ , then the cost-sensitive empirical risk:

$$\mathcal{L}_{cost}(\Psi) = \frac{1}{|\gamma|} \sum_{i=1}^{|\gamma|} w_i \cdot \ell(\Psi(\xi_i), \zeta_i)$$

Where:

$$w_i = \begin{cases} \omega_\beta, & \text{if } \zeta_i = 1 \text{ (minority)} \\ \omega_\alpha, & \text{if } \zeta_i = 0 \text{ (majority)} \end{cases} \text{ with } \omega_\beta \gg \omega_\alpha$$

In cancer detection:

$$\omega_\beta = C_{FN} \gg \omega_\alpha = C_{FP}, C_{FN}, C_{FP} \in \mathbb{R}_+$$

Sampling-based Balancing

Define resampling operators ϕ_{up}, ϕ_{down} , where:

- $\phi_{up}: \beta \rightarrow \beta', |\beta'| \approx |\alpha|$,
- $\phi_{down}: \alpha \rightarrow \alpha', |\alpha'| \approx |\beta|$,
- Balanced training set $\gamma' = \phi_{up}(\beta) \cup \phi_{down}(\alpha)$

This alters the empirical distribution $\hat{\mathbb{P}}_{\gamma'}$, but incurs:

Time complexity: $\mathcal{O}(|\gamma|^2)$ in high dimensions for oversampling

Active Learning with Entropy Acquisition

Let unlabeled pool $\mathcal{U} \subset \mathcal{X}$, define entropy function:

$$\mathcal{H}(\Psi(\xi)) = - \sum_{c \in \mathcal{Y}} p_c \log p_c, p_c = \mathbb{P}(\Psi(\xi) = c)$$

Acquisition function selects:

$$\xi^* = \arg \max_{\xi \in \mathcal{U}} \mathcal{H}(\Psi(\xi))$$

Active learning rounds:

$$\gamma_t = \gamma_{t-1} \cup \{(\xi^*, \zeta^*)\}, \zeta^* \text{ queried via oracle}$$

This prioritizes instances that induce decision boundary uncertainty, particularly around minority class regions.

Penalty-weighted Optimization Objective

Regularized empirical loss becomes:

$$\Psi^* = \arg \min_{\Psi} [\mathcal{L}_{cost}(\Psi) + \lambda \Omega(\Psi)]$$

where Ω is a penalty term (e.g., L2 norm), and $\lambda \in \mathbb{R}_+$ controls model complexity.

Class-weight Dynamics in Real-Time Risk Scenarios

Let class-weight vector evolve over time $t \in \mathbb{R}_+$, adapting to risk:

$$\omega_\beta(t) = f(\text{clinical severity, false negative rate}_t), \frac{d\omega_\beta}{dt} \geq 0$$

E.g., in cancer detection:

$$f = \delta \cdot \text{Sensitivity}_t^{-1}, \delta \in \mathbb{R}_+$$

Generalization Bound with Cost-weighted Margin

Margin $\gamma_i = y_i f(x_i)$ extended to weighted margin:

$$\gamma'_i = w_i \cdot y_i f(x_i)$$

Define margin loss over margin distribution:

$$\mathcal{L}_{margin} = \mathbb{E}_i[\mathbb{I}(\gamma'_i \leq \theta)]$$

Generalization bound (probabilistic form):

$$\mathbb{P}[\mathcal{L}_{gen}(\Psi) - \mathcal{L}_{emp}(\Psi) > \epsilon] \leq \exp(-2n\epsilon^2/(w_{max}^2 R^2))$$

where R is the radius of feature space and $w_{max} = \max\{\omega_\alpha, \omega_\beta\}$.

Final Hybrid Framework Objective

Define unified training procedure:

$$\Psi_t^* = \arg \min_{\Psi} \left[\sum_{i=1}^{|Y_t|} w_i^{(t)} \cdot \ell(\Psi(\xi_i), \zeta_i) + \lambda_t \|\Psi\|^2 \right], w_i^{(t)} \text{ dynamically updated}$$

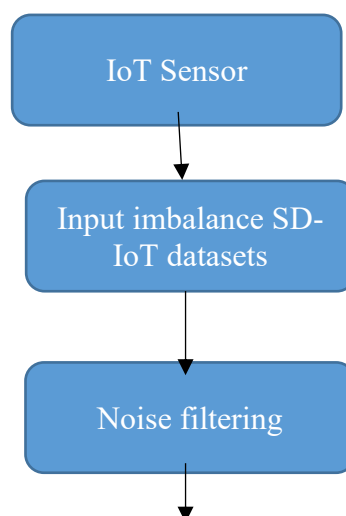
Subject to:

- Data sampling adjustments: ϕ_t
- Active acquisition rules: $\xi^* \sim \arg \max \mathcal{H}(\Psi_t(\xi))$
- Real-world misclassification costs

To overcome the above-mentioned issues, suggested a unique strategy to preserve high positive outcomes on disturbed data for further feature analysis, enabling machine-learning models on sensitive data without sacrificing privacy in a distributed machine-learning environment.

3. Proposed Methodology

This section explains the proposed methodology with data filtering, feature ranking based unsupervised learning, and a classification framework for real-time patient's SD-IoT severity prediction. The framework involves three phases: real-time data capturing and filtering, unsupervised learning, and classification prediction. Fig 1 depicts the schematic diagram for the proposed model, which includes the following phases: 1) Real-time patient data capturing and monitoring, 2) data filtering, 3) Cluster-based ensemble classification. Initially, data is acquired from real-time sensors and transmitted to AWS cloud storage for data analysis. The data is then preprocessed using the filtering approach and outlier detection approaches. In the second phase, feature ranking and density-based clustering approaches are proposed to group contextual similar diseases patients for class balancing. Finally, a hybrid ensemble learning classifier is proposed to predict the SD-IoT patient's disease severity. Cluster-based class membership plays a key role in identifying patterns in different imbalanced class distributions. Clustering can assist in identifying trends within the minority class that may not be visible when considering the entire dataset. One approach to using clustering for imbalanced datasets is to use clustering algorithms to identify clusters of data points within the minority class. These clusters can be used to generate data points to balance the class distribution and assign class membership to the data points. This involves determining the cluster to which each data point belongs and assigning it to the minority class if it falls within a cluster that contains a higher percentage of minority class data points. Probabilistic models, such as Gaussian distributions, can be used to assign class membership by modeling the distribution of each class. This approach not only predicts the class label of an observation but also provides a probability score that represents the confidence of the prediction. Figure 1 illustrates the proposed model's overall framework. Initially, the model filters imbalanced datasets using a noise filtering approach and a feature ranking measure. In the next step, it employs the K-density probabilistic clustering approach to determine the class membership of the filtered data. Finally, the model implements an ensemble learning framework on the clustered data to improve the statistical classification metrics.



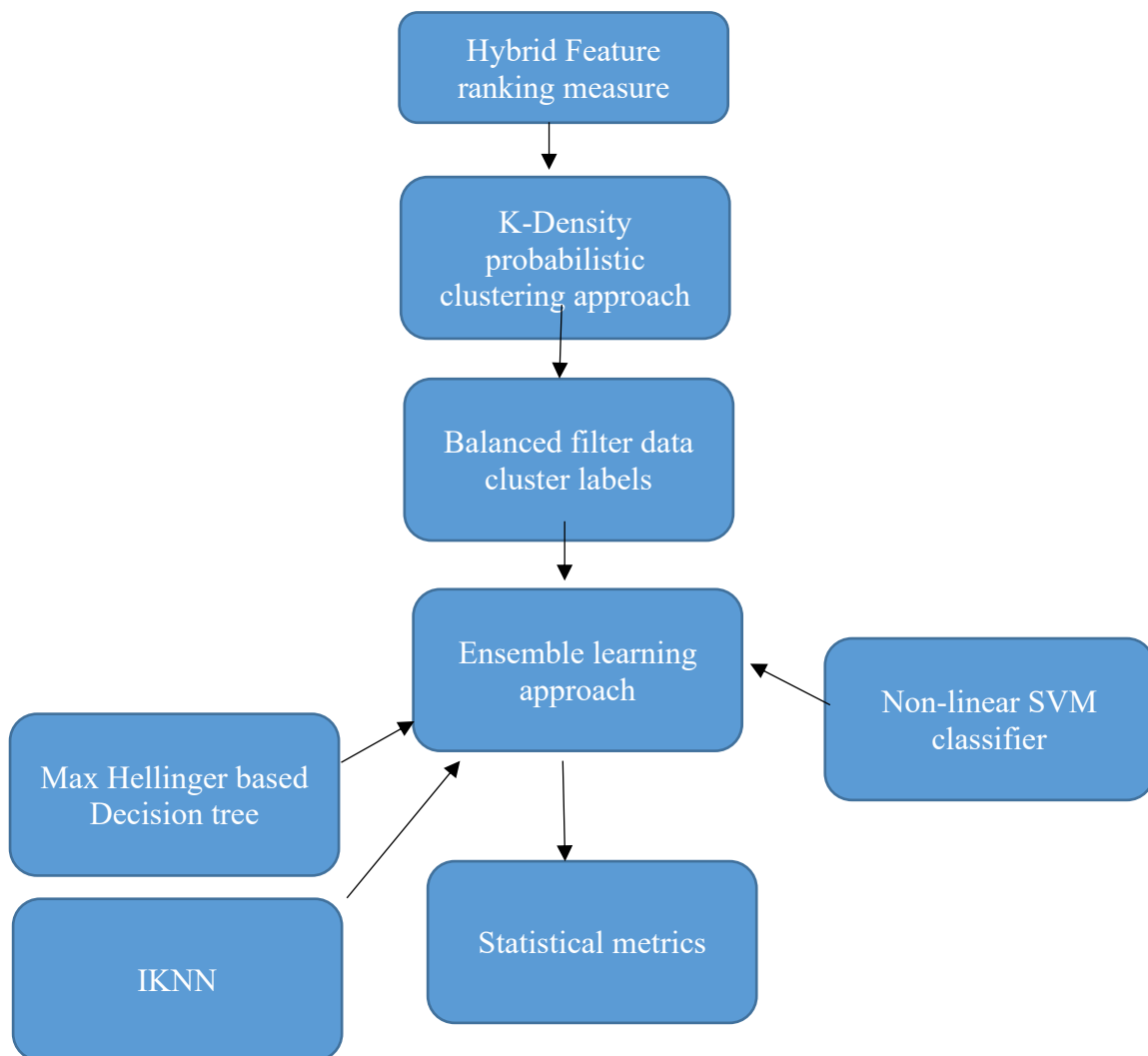


Figure 1: Proposed Framework

1. Proposed Noise Filtering Approach using gamma quartile variation

Sparse filtering is a method that can be utilized to address imbalanced data for feature ranking, remove noise, and filter out missing values. In the context of missing values, sparse filtering can estimate missing values by using a non-linear technique based on the non-linear Gaussian model. The non-linear Gaussian estimation (NGE) method is a statistical approach used to estimate missing values or feature transformations based on this non-linear Gaussian model. The non-linear Gaussian model is more adaptable than conventional linear techniques in representing underlying relationships and structures of data. NGE and sparse filtering can transform numerical values using this non-linear Gaussian model, which can improve data quality for feature ranking and other tasks. The Non-linear Gaussian estimation equation can be utilized to identify sparse features in imbalanced data with a high degree of missing values. Let the original data matrix be denoted by $X \in \mathbb{R}^{n \times d}$, where n is the number of instances and d the number of features. We represent the feature vectors with Greek symbols as $\xi_i \in \mathbb{R}^d$, denoting the i -th observation.

In sparse filtering, each feature vector ξ_i is transformed into a sparse activation $\varphi_i \in \mathbb{R}^k$, where $k \leq d$, using a sparse projection matrix $\Theta \in \mathbb{R}^{k \times d}$, such that:

$$\varphi_i = \sigma(\Theta \xi_i)$$

Here, $\sigma(\cdot)$ is a non-linear activation function such as soft absolute value or rectifier (ReLU). To handle noise and imbalanced data Ψ_{imb} , we introduce normalization constraints, where the row-wise (feature-wise) and column-wise (sample-wise) normalizations are applied:

$$\varphi_{ij} \leftarrow \frac{\varphi_{ij}}{\sqrt{\sum_j \varphi_{ij}^2}}, \varphi_{ij} \leftarrow \frac{\varphi_{ij}}{\sqrt{\sum_i \varphi_{ij}^2}}$$

Sparse filtering minimizes the overall L1 norm of normalized activations:

$$\mathcal{L}_{\text{sparse}} = \sum_{i=1}^n \sum_{j=1}^k |\varphi_{ij}|$$

For missing value estimation and filtering noisy components, we integrate the non-linear Gaussian estimation model. Assume the missing feature values are modeled by a multivariate Gaussian distribution $\mathcal{N}(\mathbf{v}, \mathbf{\Sigma})$, where $\mathbf{v} \in \mathbb{R}^d$ is the mean vector and $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is the covariance matrix.

2. Feature ranking measures

Feature ranking is especially important for imbalanced datasets, which are characterized by a significant imbalance between the majority and minority classes, with the minority class typically having a much smaller representation. Feature ranking is an essential step in developing effective machine learning models for imbalanced datasets. To enhance model performance and avoid overfitting, selecting the most relevant features for imbalanced datasets is crucial, especially when the majority and minority classes are strongly imbalanced.

The non-linear Gaussian estimate Γ_i of the feature vector ξ_i is computed as:

$$\Gamma_i = \exp\left(-\frac{1}{2}(\xi_i - \mathbf{v})^T \mathbf{\Sigma}^{-1}(\xi_i - \mathbf{v})\right)$$

This formulation ensures that data points distant from the Gaussian center receive low confidence scores, effectively weighting less informative or corrupted entries.

The combined model aims to minimize a total loss function that balances sparse filtering and Gaussian regularization, given by:

$$\mathcal{L}_{\text{total}} = \lambda_1 \sum_{i=1}^n \sum_{j=1}^k |\varphi_{ij}| + \lambda_2 \sum_{i=1}^n \|\xi_i - \mathbf{v}\|_{\mathbf{\Sigma}^{-1}}^2$$

The Mahalanobis distance component in the second term measures the deviation of each sample from the Gaussian prototype $(\mathbf{v}, \mathbf{\Sigma})$, preserving underlying data structure.

Assuming sparsity acts as a prior, we treat this as a MAP estimation where the posterior of $\boldsymbol{\theta}$ is:

$$P(\boldsymbol{\theta}|\mathbf{X}) \propto \exp(-\mathcal{L}_{\text{total}})$$

To solve this model, we perform gradient-based optimization over $\boldsymbol{\theta}$, $\mathbf{v}$, and $\mathbf{\Sigma}$, treating missing features via imputation as:

$$\hat{\xi}_{ij} = \mathbb{E}[\xi_{ij}|\xi_{-j}, \mathbf{v}, \mathbf{\Sigma}]$$

Finally, the filtered and enhanced feature representation is the projection:

$$\mathcal{T}(\xi_i) = \varphi_i \cdot \Gamma_i$$

which combines sparse activation and non-linear Gaussian confidence, ensuring robust feature extraction even under high dimensional imbalance and missingness.

Let the dielectric feature vector be denoted:

$$\boldsymbol{\delta} = \{\delta_1, \delta_2, \dots, \delta_n\}, \delta_i \in \mathbb{R}$$

Define the absolute value vector:

$$\boldsymbol{\alpha} = \{|\delta_1|, |\delta_2|, \dots, |\delta_n|\}$$

Define the first quartile range (R1):

$$\rho_1 = Q_{25\%}(\boldsymbol{\alpha})$$

Define the second quartile range (R2) as average of the 25th and 75th percentiles:

$$\rho_2 = \frac{1}{2}(Q_{25\%}(\boldsymbol{\alpha}) + Q_{75\%}(\boldsymbol{\alpha}))$$

Define the third quartile range (R3) as the average of the 75th percentile and the 25th percentile from the end (which is the 75th percentile of the reversed vector):

$$\rho_3 = \frac{1}{2}(Q_{75\%}(\boldsymbol{\alpha}) + Q_{25\%}(\boldsymbol{\alpha}^{\leftarrow}))$$

Compute the upper outlier threshold Υ_U :

$$\Upsilon_U = \rho_3 + \eta \cdot \log \Gamma(\rho_3 - \rho_1)$$

Compute the lower outlier threshold Υ_L :

$$\Upsilon_L = \rho_1 - \eta \cdot \log \Gamma(\rho_3 - \rho_1)$$

Define the upper outlier limit $\mathcal{O}_U$:

$$\mathcal{O}_U = \rho_3 + \eta \cdot \min\{\chi_9^2(\rho_3 + \rho_1), \exp(\Gamma(\rho_3 - \rho_1))\}$$

Define the lower outlier limit $\mathcal{O}_L$:

$$\mathcal{O}_L = \rho_3 - \eta \cdot \min\{\chi_9^2(\rho_3 + \rho_1), \exp(\Gamma(\rho_3 - \rho_1))\}$$

Feature selection

Given: $\Delta = \{(\mathcal{X}_{1i}, \mathcal{X}_{2i}, \dots, \mathcal{X}_{mi}, \kappa_i)\}_{i=1}^n$

Let:

- $\mathcal{F}_\chi$: Set of features selected via Chi-Squared test.
- $\mathcal{F}_\mu$: Set of features selected via Mutual Information.
- $\mathcal{F}_\gamma$: Set of features selected via Gain Ratio.
- $\mathcal{F}_\rho$: Set of redundant features via correlation.
- $\mathcal{F}_{\text{ensemble}} = \mathcal{F}_\chi \cap \mathcal{F}_\mu \cap \mathcal{F}_\gamma$: Intersection of important features.

Chi-Squared Test:

For each feature $\mathcal{X}_j$, compute:

$$\chi^2(\mathcal{X}_j, \kappa) = \sum_{u \in \mathcal{U}, v \in \mathcal{V}} \frac{(\mathcal{O}_{uv} - E_{uv})^2}{E_{uv}}$$

Where:

- $\mathcal{U}$: Unique values in feature $\mathcal{X}_j$
- $\mathcal{V}$: Unique class labels in κ
- $\mathcal{O}_{uv}$: Observed frequency
- E_{uv} : Expected frequency under independence

Select $\mathcal{X}_j \in \mathcal{F}_\chi$ if $\chi^2 > \theta_\chi$

Mutual Information (MI):

$$I(\mathcal{X}_j; \kappa) = \sum_{x \in \mathcal{X}_j} \sum_{y \in \mathcal{Y}} P(x, y) \log \left(\frac{P(x, y)}{P(x)P(y)} \right)$$

Select $\mathcal{X}_j \in \mathcal{F}_\mu$ if $I(\mathcal{X}_j; \kappa) > \theta_\mu$

Gain Ratio:

Let:

- $H(\kappa)$: Entropy of class label.
- $H(\kappa|\mathcal{X}_j)$: Conditional entropy.

$$\text{Gain}(\mathcal{X}_j) = H(\kappa) - H(\kappa|\mathcal{X}_j)$$

$$\text{SplitInfo}(\mathcal{X}_j) = - \sum_{v \in \mathcal{X}_j} \frac{|v|}{n} \log \left(\frac{|v|}{n} \right)$$

$$\text{GainRatio}(\mathcal{X}_j) = \frac{\text{Gain}(\mathcal{X}_j)}{\text{SplitInfo}(\mathcal{X}_j)}$$

Select $\mathcal{X}_j \in \mathcal{F}_\gamma$ if $\text{GainRatio} > \theta_\gamma$

Correlation Filtering:

Compute correlation matrix $\rho \in \mathbb{R}^{m \times m}$:

$$\rho_{jk} = \frac{\text{Cov}(\mathcal{X}_j, \mathcal{X}_k)}{\sigma_{\mathcal{X}_j} \sigma_{\mathcal{X}_k}}$$

Form:

$$\mathcal{F}_\rho = \{(\mathcal{X}_j, \mathcal{X}_k) \mid \rho_{jk} > \tau, j \neq k\}$$

Remove one feature from each pair $\mathcal{F}_\rho$

Final Feature Set:

$$\Phi_{\text{imp}} = \mathcal{F}_{\text{ensemble}} \setminus \mathcal{F}_\rho$$

Feature ranking algorithm

Definitions and Inputs

Let:

Dataset: $\Delta = \{\mathcal{X}_1, \dots, \mathcal{X}_m\} \in \mathbb{R}^{n \times m}$

Target variable: $\kappa \in \mathbb{R}^n$

Correlation threshold: $\tau \in [0,1]$

Classes: $\mathcal{C} = \{\zeta_1, \zeta_2, \dots, \zeta_K\}$

Final feature ranking score for $\mathcal{X}_j$: $\mathcal{H}_j$

1. Feature-Wise Statistical Metrics

For each feature $\mathcal{X}_j \in \Delta$:

Chi-Squared Statistic:

$$\chi_j^2 = \sum_{u,v} \frac{(O_{uv} - E_{uv})^2}{E_{uv}} \quad (\text{Observed } O_{uv}, \text{Expected } E_{uv})$$

Mutual Information:

$$I_j = \sum_{x \in \mathcal{X}_j} \sum_{y \in \mathcal{Y}} P(x, y) \log \left(\frac{P(x, y)}{P(x)P(y)} \right)$$

Gain Ratio:

$$\text{GR}_j = \frac{H(\kappa) - H(\kappa | \mathcal{X}_j)}{-\sum_v \frac{|v|}{n} \log \left(\frac{|v|}{n} \right)}$$

2. Feature Voting via Ensemble Intersection

Define selected sets:

$$\mathcal{F}_\chi = \{\mathcal{X}_j \mid \chi_j^2 > \theta_\chi\}$$

$$\mathcal{F}_\mu = \{\mathcal{X}_j \mid I_j > \theta_\mu\}$$

$$\mathcal{F}_\gamma = \{\mathcal{X}_j \mid \text{GR}_j > \theta_\gamma\}$$

Ensemble-selected features:

$$\mathcal{F}_{\text{ensemble}} = \mathcal{F}_\chi \cap \mathcal{F}_\mu \cap \mathcal{F}_\gamma$$

3. Redundancy Pruning via Correlation Matrix

Correlation matrix:

$$\rho_{jk} = \frac{\text{Cov}(\mathcal{X}_j, \mathcal{X}_k)}{\sigma_j \sigma_k}$$

Identify redundant pairs:

$$\mathcal{F}_\rho = \{(\mathcal{X}_j, \mathcal{X}_k) \mid \rho_{jk} > \tau\}$$

Remove one feature per redundant pair from $\mathcal{F}_{\text{ensemble}}$

4. Hybrid Feature Ranking Measure (HFRM) Using Entropy Fusion

Let:

$H(\mathcal{X}_j \mid \zeta_i)$: Entropy of feature j conditioned on class ζ_i

Fusion entropy:

$$\mathcal{H}_j = \sum_{i=1}^K \sum_{k=1}^K \omega_{ik} \cdot [H(\mathcal{X}_j \mid \zeta_i) - H(\mathcal{X}_j \mid \zeta_k)]^2$$

Where:

ω_{ik} : Class-level weighting between class ζ_i and ζ_k

Normalization over class pairs ensures robustness in imbalanced settings.

5. Final Feature Set Selection

Apply a global threshold or top-k selection:

$$\Phi_{\text{final}} = \{\mathcal{X}_j \mid \mathcal{H}_j > \theta_H\}$$

Return φ_{final} : Feature subset with high statistical, information-theoretic, and correlation-aware relevance.

Ensemble feature ranking with different measures are useful for identifying the most important features in imbalanced datasets. By combining multiple feature ranking techniques, it can help to reduce the risk of overfitting and improve the overall performance of machine learning models.

The approach aims to eliminate redundant features and retain a subset λ_1

containing important features, where $\lambda_1 \in \text{Features (F)}$.

The mathematical explanation is as follows:

- λ_1 is defined as the set difference between two sets: $\mathcal{E}_w$ and $\mathcal{E}_{\text{correlation}}$.
- $\mathcal{E}_w$ is the set of features that are important according to the Chi-Squared Test (χ^2).
- $\mathcal{E}_w = \{x | x \in \text{Chi}^2 \cap MI\}$
- $\mathcal{E}_{\text{correlation}}$ is the set of features with high correlation.
- $\mathcal{E}_{\text{correlation}} = \{x | x \in \text{High correlation in features}\}$
- Chi^2 represents the set of important features found from the Chi-Squared Test.
- MI represents the set of important features found from Mutual Information.

In the next phase of the approach:

- The top 80% of features are selected for further analysis, stored in two different sets: Chi^2 and MI .
- Highly correlated features with over 85% correlation are identified and stored in $\mathcal{E}_{\text{correlation}}$.

Algorithm : Binary PSO Algorithm

Input:

- Constants c_1 and c_2
- Weight factors w_{max} and w_{min}
- Maximum velocity V_{max}
- Maximum iteration max_iter

Initialization:

- Initialize the population randomly as q for each solution, where each solution is a D -dimensional vector.
- Set $t = 0$.

- Calculate the initial value of w as $w = w_{\text{max}} - \text{iteration} \cdot \left(\frac{w_{\text{max}} - w_{\text{min}}}{m_{\text{max}} - m_{\text{min}}} \right)$.
- Evaluate the fitness value for each solution as $f(q_{i,j}^t)$.
- Initialize Pbest and Gbest values.

3. K-Density probabilistic clustering approach

The K-Density probabilistic clustering approach is an extension of the EM (Expectation-Maximization) model implemented for imbalanced data.

Main Loop:

- For $k = 1$ to SS (where SS is the swarm size):
- Update the velocity for each particle as:

$$V_i^{t+1} = w \cdot V_i^t + c_1 \cdot m_{i1} \cdot (Pbest_i^t - q_i^t) + c_2 \cdot m_{i2} \cdot (Gbest^t - q_i^t)$$
- For $k = 1$ to SS :
- For $j = 1$ to D (where D is the dimensionality):
- If $v(i, j)$ is greater than V_{max} set $v(i, j)$ equal to V_{max} .
- If $v(i, j)$ is less than $-V_{max}$, set $v(i, j)$ equal to $-V_{max}$.
- Calculate s as $s = \frac{1}{1 + e^{-v(i, j)}}$.
- If $rand < s$, set $q_{i,j}^{t+1}$ equal to 1; otherwise, set $q_{i,j}^{t+1}$ equal to 0.
- Increment t by 1 .

Output:

- Produce the best solution based on the optimization process.

It can handle datasets with a significant degree of class imbalance where the minority class has a very low sample size, withstand data noise and outliers, and locate complex and non-linear clusters, which is crucial for numerous real-world uses. It is utilized to identify clusters in datasets with imbalances by utilizing a probabilistic framework. The approach represents the probability in the data clusters by a mixture of probability distributions, and it employs the EM algorithm to estimate the parameters of these distributions. The method iteratively updates the estimations of the mixture components until convergence is obtained. The K-Density probabilistic clustering approach estimates the probability density function of each cluster using a kernel density estimator. The ability of the K-Density probabilistic clustering approach to be applied to both binary and multi-class classification issues is one of its main benefits. It can also be utilized for unsupervised learning when the number of classes is unknown in advance. Another benefit of the K-Density probabilistic clustering approach is its ability to be used with other methods to enhance the performance of machine learning models.

4. Ensemble classification framework

c) Optimal KNN

Algorithm ENN (Edited Nearest Neighbors)

Input:

- Dataset S
- The majority samples S_{maj}
- Number of nearest neighbors k
- Initial under-sampling rate M

Output:

- First balanced dataset S

Definitions:

Definition : The k -nearest neighbor of the sample S_i is the sum of samples in dataset S the distance between which and S_i is less than the distance

between the k -th nearest neighbor sample of S_i and S_i . It is written as:

$$KNN(S_i, k) = \{y_i \in S \mid \text{dist}(S_j, S_i) \leq \text{dist}(S'_i, S_i)\}$$

where S_i is the k -th nearest neighbor sample of S_i in dataset S , and dist is the

Optimized KNN steps:

Algorithm Steps:

1. Initialize the under-sampling multiple M and the value of $M_{del} (M \times |S_{min}|)$ for counting the total number of deleted samples, set to 0 .
2. Traverse each of the major class samples S_{maj} :
3. Set $C_{det} = 0$. Find k nearest neighbor samples of $S_{i_{maj}}$ and store their indexes in K_{ind} .
4. According to Equation , select the three samples closest to $S_{i_{maj}}$ whose indexes are in K_{maj} ? as their neighbor samples.
5. Compare the label of $S_{i_{maj}}$ with its neighbor sample classes. If the label of $S_{i_{maj}}$ is different from that of more than two neighbor samples, delete it.
6. $C_{del} = C_{del} + 1$. If $C_{del} < M_{del}$, return to Step 3. Otherwise, the operation for $S_{i_{maj}}$ is ended.
7. Remove the deleted samples from the dataset S to obtain the first balanced dataset S .

SVM Models with Feature Selection

In a training set Ω , the elements (usually called vectors) are partitioned into two classes and represented by a pair $(x_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$,

where n is the number of features observed for each vector.

x_i contains the feature values for vector i , and y_i indicates to which of the two classes of Ω vector i belongs.

The soft margin Support Vector Machine (SVM) problem consists of determining

the hyperplane $f(x) = w^T \cdot x + b$ that separates the vectors in the training set

in such a way that the distance between two parallel hyperplanes

supporting some vector of the two classes is maximized, and the

sum of classification errors is minimized.

The soft margin SVM model minimizes a compromise between these two objectives,

known respectively as the structural risk and the empirical risk.

SVM Model Formulation:

$$\begin{aligned}
 \min \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\
 \text{s.t.} \quad & y_i (w^T x_i + b) \geq 1 - \xi_i \quad i = 1, \dots, m, \\
 & \xi_i \geq 0 \quad i = 1, \dots, m, \\
 & w_j \in \mathbb{R} \quad j = 1, \dots, n, \\
 & b \in \mathbb{R}
 \end{aligned}$$

- Variables w_j and b represent the coefficients of the two parallel hyperplanes $w^T x + b = 1$ and $w^T x + b = -1$.
 - The first term in the objective function $\frac{1}{2} \|w\|^2$ is the structural risk, as $\|w\|$ is twice the inverse of the distance between these two hyperplanes.
 - The second term in the objective function $\sum_{i=1}^m \xi_i$ is the empirical risk, i.e., the sum of the deviation of misclassified objects.
 - Decision makers use the parameter C to regulate the trade-off between the two objectives.
 - Constraints in the model ensure that either vectors i in the class represented by $y_i = 1$ satisfy $(w^T x_i + b) \geq 1$ and vectors in class $y_i = -1$ satisfy $(w^T x_i + b) \leq -1$, or variable ξ takes a positive value.
- This formulation describes the soft margin SVM model with feature selection.
- It aims to find an optimal hyperplane that maximizes the margin between classes while considering a trade-off parameter C to handle misclassification errors.

Stage 1: Feature Preprocessing and Outlier Pruning

For each feature χ_j :

- Compute absolute quartile boundaries:

$$\rho_1 = Q_{25\%}(|\chi_j|), \rho_3 = \frac{1}{2}(Q_{75\%} + Q_{25\%}^{\text{rev}})$$

- Compute outlier thresholds:

$$\begin{aligned} \mathcal{O}_U &= \rho_3 + \eta \cdot \min(\chi_j^2(\rho_3 + \rho_1), \exp(\Gamma(\rho_3 - \rho_1))) \\ \mathcal{O}_L &= \rho_3 - \eta \cdot \min(\chi_j^2(\rho_3 + \rho_1), \exp(\Gamma(\rho_3 - \rho_1))) \end{aligned}$$

- Filter:

$$\chi_j \leftarrow \chi_j[\chi_{j,i} \in [\mathcal{O}_L, \mathcal{O}_U]]$$

Stage 2: Independent Feature Importance Measures

For each $\chi_j \in \Delta$:

1. **Chi-Squared Importance:**

$$\chi_j^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \rightarrow \mathcal{F}_\chi = \{\chi_j \mid \chi_j^2 > \theta_\chi\}$$

2. **Mutual Information:**

$$I_j = \sum_{x,y} P(x,y) \log \left(\frac{P(x,y)}{P(x)P(y)} \right) \rightarrow \mathcal{F}_\mu = \{\chi_j \mid I_j > \theta_\mu\}$$

3. **Gain Ratio:**

$$\text{GainRatio}_j = \frac{H(\kappa) - H(\kappa|\chi_j)}{\text{SplitInfo}(\chi_j)} \rightarrow \mathcal{F}_\gamma = \{\chi_j \mid \text{GainRatio}_j > \theta_\gamma\}$$

Stage 3: Ensemble Voting Mechanism

Voting function:

$$\mathcal{V}(\chi_j) = \sum_{f \in \{\mathcal{F}_\chi, \mathcal{F}_\mu, \mathcal{F}_\gamma\}} \mathbb{1}_{\chi_j \in f}$$

Define:

$$\mathcal{F}_{\text{voted}} = \{\chi_j \mid \mathcal{V}(\chi_j) \geq 2\}$$

Stage 4: Redundancy Pruning via Correlation

Compute correlation matrix:

$$\rho_{jk} = \frac{\text{Cov}(\chi_j, \chi_k)}{\sigma_j \sigma_k}$$

Redundant feature pairs:

$$\mathcal{F}_\rho = \{(\chi_j, \chi_k) \mid \rho_{jk} > \tau, j \neq k\}$$

Remove one feature per pair from $\mathcal{F}_{\text{voted}}$

Stage 5: Final Selection

Final important features:

$$\phi_{\text{final}} = \mathcal{F}_{\text{voted}} \setminus \mathcal{F}_p$$

4.Experimental Analysis

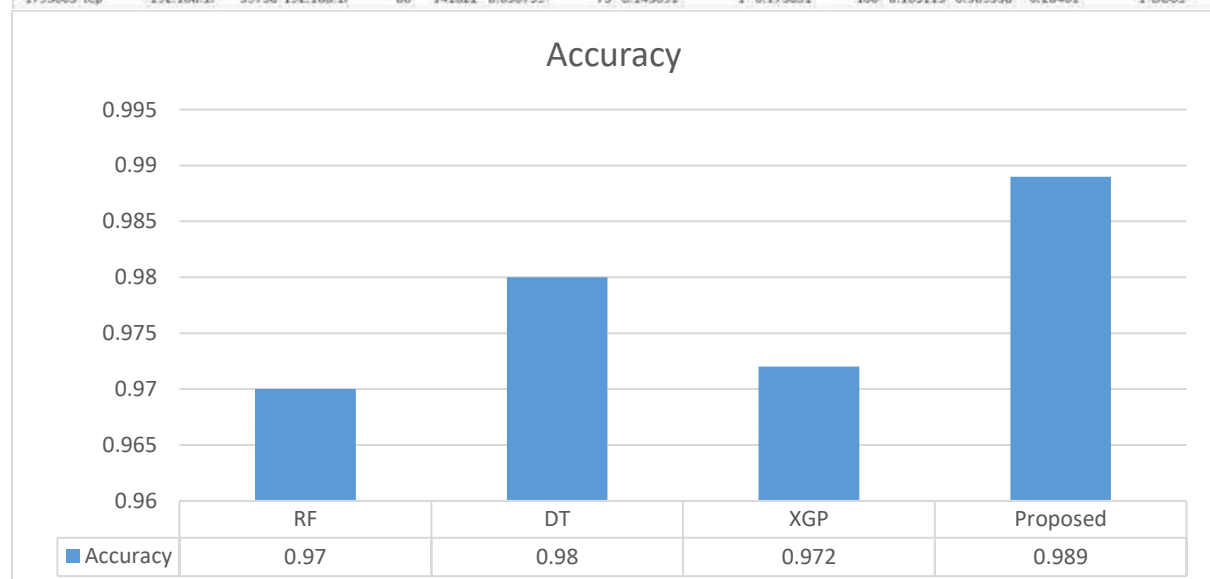
This section presents experimental results to evaluate our proposed framework in terms of accuracy, precision, recall, and F-measure, specifically on classification tasks on different imbalanced SD-IoT datasets. The experiment involved conducting 10 iterations of 10-fold cross-validation on a selection of imbalanced datasets from distributed SD-IoT hospitals. The datasets were sourced from various hospitals to capture different features using real-time sensors. Here, captured data from each hospital is stored in AWS S3. The selection process aimed to ensure that the datasets represented a range of properties, including varying imbalance ratios, numbers of classes, total examples, and attributes.

Table 1: Sample BoT-IoT dataset

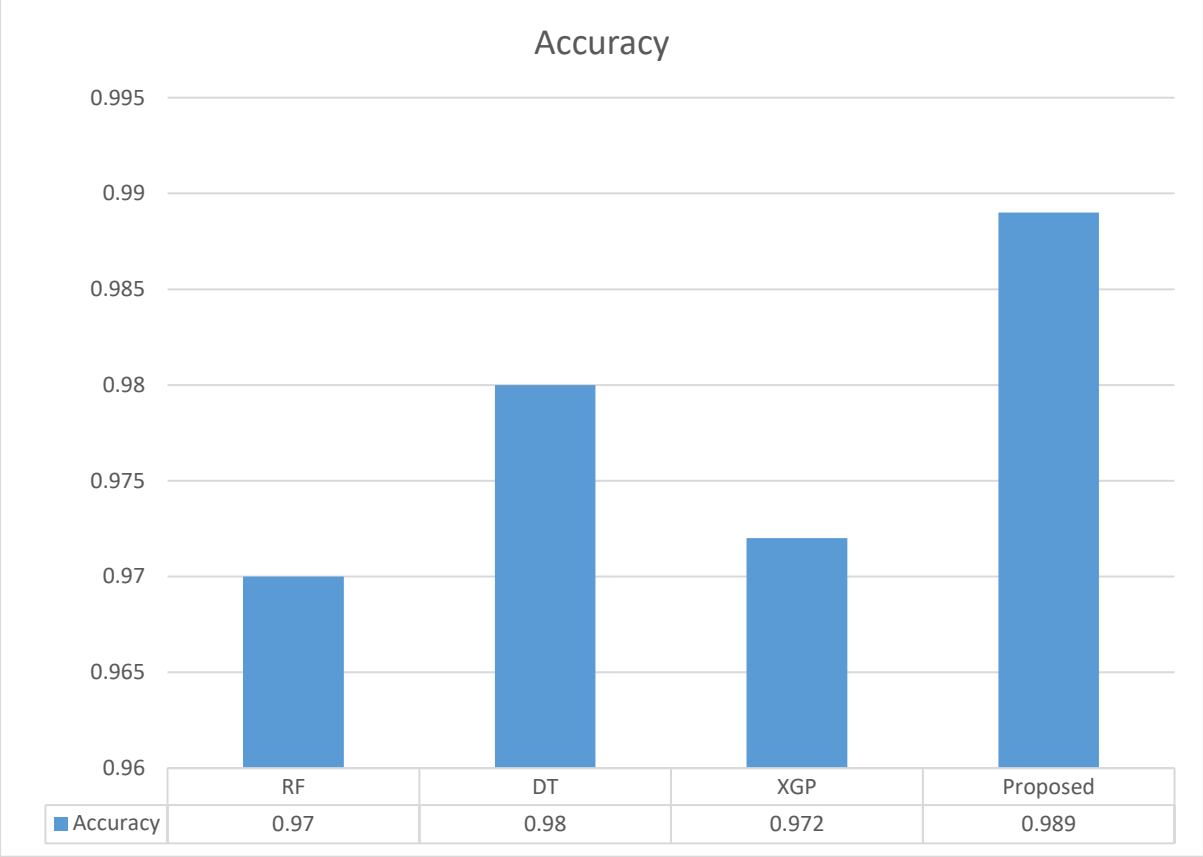
ML algorithms	Results	Our proposed framework
Random forest	The authors achieved different results by varying features of RF. Accuracy has been the major factor in evaluation. While authors also optimized run-time overheads by reducing RF forest size and tree path.	In comparison to this framework, we used three different algorithms and achieved an accuracy rate of 97.4%, 96.1%, and 98.1%, for Naïve Bayes, SVM, and DT respectively. From the results, it is noticed that the forest size affects the accuracy.
Decision tree	F1-score is reported to be approximately 97% showing that the system detects DDoS attacks with high accuracy	In comparison to the presented study, we used three different algorithms and the DT achieved an accuracy ratio of 98.1%.
Random Forest, Logistic Regression, and Extreme Gradient Boost (XGP)	The authors achieved results at a sampling rate of 20% of network traffic, showing high precision of approximately 93%, and a low false alarm rate of 96%.	Our proposed framework, on average, achieved an accuracy of 97.2%

The BoT-IoT dataset was designed and created in the Cyber Range Lab of UNSW Canberra Cyber. It contains a combination of normal and botnet traffic, recorded in pcap files that are 69.3 GB with over 72 million records. The data is separated into categories and subcategories based on types of attacks, including DDoS, DoS, OS and Service Scan, Keylogging, and Data exfiltration. To make it easier to work with, 5% of the data, about 1.07 GB and 3 million records.

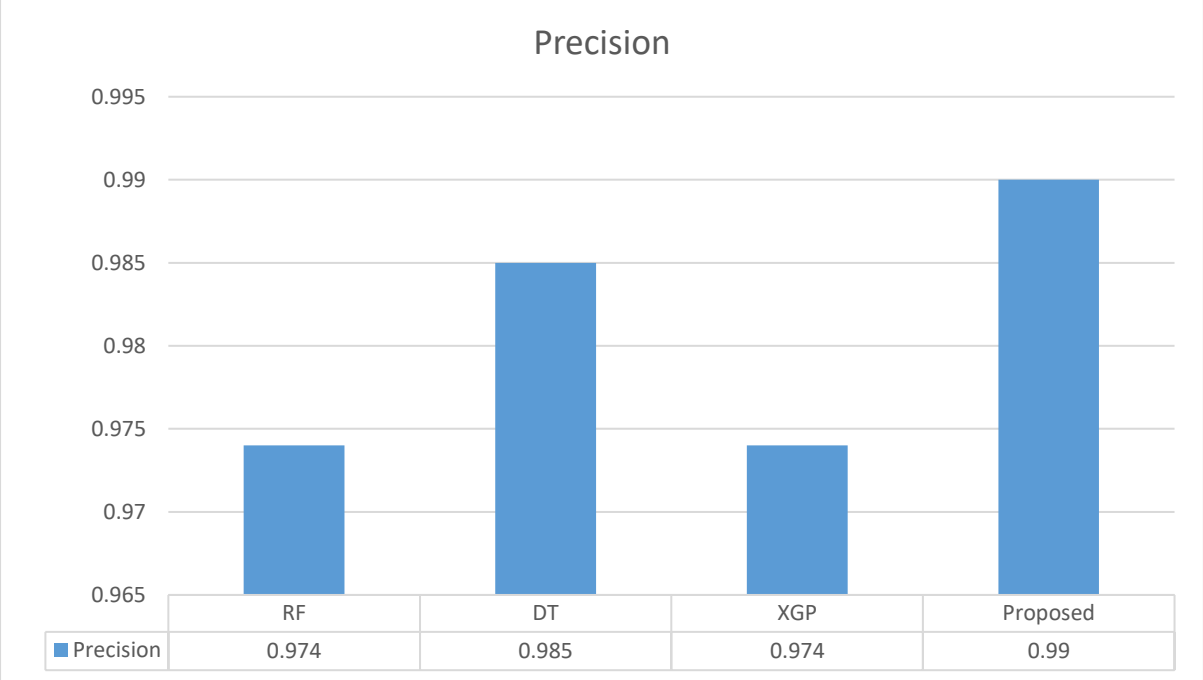
pkSeqID	proto	saddr	sport	daddr	dport	seq	stddev	N_IN_Con	min	state_num	mean	N_IN_Con	rate	max	attack	category
792371	udp	192.168.1.1	48516	192.168.1.1	80	175094	0.226784	100	4.100436	4	4.457383	100	0	0.404711	4.719438	1 DoS
2056438	tcp	192.168.1.1	22267	192.168.1.1	80	143024	0.451998	100	3.439257	1	3.806172	100	0.225077	0.401397	4.44293	1 DDoS
2795650	udp	192.168.1.1	28629	192.168.1.1	80	167033	1.931553	73	0	4	2.731204	100	0	0.407287	4.138455	1 DDoS
2118009	tcp	192.168.1.1	42142	192.168.1.1	80	204615	0.428798	56	3.271411	1	3.626428	100	0	0.343654	4.2297	1 DDoS
303688	tcp	192.168.1.1	1645	192.168.1.1	80	40058	2.058381	100	0	3	1.188407	100	0	0.135842	4.753628	1 DoS
420025	tcp	192.168.1.1	39733	192.168.1.1	80	156396	2.177835	36	0	3	1.539962	36	0	0.127912	4.619887	1 DoS
3008812	udp	192.168.1.1	10800	192.168.1.1	80	118034	1.368196	100	1.97518	4	3.910081	100	0	1.02512	4.885159	1 DDoS
1064106	udp	192.168.1.1	19625	192.168.1.1	80	184672	1.788452	100	0	4	3.576574	100	0	0.446612	4.49208	1 DoS
3258414	udp	192.168.1.1	22692	192.168.1.1	80	105486	0.822443	100	2.98003	4	3.982845	100	0	1.003092	4.994536	1 DDoS
1793063	tcp	192.168.1.1	39738	192.168.1.1	80	141822	0.030759	73	0.143091	1	0.173851	100	0.103113	0.309338	0.20461	1 DDoS



The above figure shows the results of the classification accuracy achieved using this proposed technique on Bot-IoT dataset. The X-axis of the graph would represent the different techniques or methods used, while the Y-axis would display the classification accuracy on BoT-IOT stat data.



The above figure shows the results of the classification precision achieved using this proposed technique on Bot-IoT dataset. The X-axis of the graph would represent the different techniques or methods used, while the Y-axis would display the classification precision on BoT-IOT stat data.



The above figure shows the results of the classification recall achieved using this proposed technique on Bot-IoT dataset. The X-axis of the graph would represent the different techniques or methods used, while the Y-axis would display the classification recall on BoT-IOT stat data.

```

Correctly Classified Instances      674          96.4235 %
Incorrectly Classified Instances    25           3.5765 %
Kappa statistic                    0.9211
Mean absolute error                0.0591
Root mean squared error            0.1663
Relative absolute error            13.0727 %
Root relative squared error        34.9901 %
Total Number of Instances         699

```

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.969	0.046	0.976	0.969	0.973	0.921	0.992	0.996	1
	0.954	0.031	0.943	0.954	0.948	0.921	0.992	0.982	2
Weighted Avg.	0.964	0.040	0.964	0.964	0.964	0.921	0.992	0.991	

```

Correctly Classified Instances      687          98.2833 %
Incorrectly Classified Instances    12           1.7167 %
Kappa statistic                    0.9642
Mean absolute error                0.0301
Root mean squared error            0.1115
Relative absolute error            6.276 %
Root relative squared error        22.7536 %
Total Number of Instances         699

```

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.971	0.010	0.986	0.971	0.978	0.964	0.999	0.998	cluster1
	0.990	0.029	0.981	0.990	0.986	0.964	0.999	0.999	cluster2
Weighted Avg.	0.983	0.021	0.983	0.983	0.983	0.964	0.999	0.999	

5. Conclusion

In this paper, we propose an ensemble cluster-based classification model to address issues related to imbalanced datasets, such as the high ratio of majority to minority classes and substantial noise present in the SD-IoT data. In this work, we explored various algorithms and techniques for solving complex problems in machine learning and data analysis. We discussed a range of algorithms, including Non-Linear Differential Evolution with Mutation Enhancement (NLDE-ME), Modified Naive Bayes for Unlabeled Records, Support Vector Machine (SVM) for Classification, and the Bonferroni Mean Fuzzy k-Nearest Neighbors (BM-FKNN) algorithm. These algorithms cover a diverse set of applications, from feature selection and optimization to classification and data preprocessing.

IoT devices and networks are crucial components of the modern Internet; however, they often possess security weaknesses and vulnerabilities. The lack of security design in many IoT devices makes them susceptible to attacks that exploit these vulnerabilities, leading to severe consequences. In this work, we proposed a filter-based classification approach for detecting IoT bot cyber attacks. Our method demonstrated promising results in terms of accuracy, precision, recall, and F1-score compared to traditional methods. These findings indicate that the cluster-based classification approach is a viable solution for real-time detection of IoT bot cyber attacks. Future work in this area can include several improvements. Firstly, exploring more advanced machine learning algorithms could enhance the accuracy of the proposed method. Secondly, evaluating the method on larger and more diverse datasets would help verify its robustness and scalability. Finally, integrating the proposed method with other intrusion detection techniques could further improve overall performance and accuracy, leading to more comprehensive protection for IoT devices and networks.

References

- [1] U. K. Lilhore, S. Dalal, and S. Simaiya, "A cognitive security framework for detecting intrusions in IoT and 5G utilizing deep learning," *Computers & Security*, vol. 136, p. 103560, Jan. 2024, doi: 10.1016/j.cose.2023.103560.
- [2] C. Alex, G. Creado, W. Almobaideen, O. A. Alghanam, and M. Saadeh, "A Comprehensive Survey for IoT Security Datasets Taxonomy, Classification and Machine Learning Mechanisms," *Computers & Security*, vol. 132, p. 103283, Sep. 2023, doi: 10.1016/j.cose.2023.103283.
- [3] C. Wei, G. Xie, and Z. Diao, "A lightweight deep learning framework for botnet detecting at the IoT edge," *Computers & Security*, vol. 129, p. 103195, Jun. 2023, doi: 10.1016/j.cose.2023.103195.
- [4] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowledge-Based Systems*, vol. 279, p. 110941, Nov. 2023, doi: 10.1016/j.knosys.2023.110941.
- [5] G. A. Mukhaini, M. Anbar, S. Manickam, T. A. Al-Amiedy, and A. A. Momani, "A systematic literature review of recent lightweight detection approaches leveraging machine and deep learning mechanisms in Internet of Things networks," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 1, p. 101866, Jan. 2024, doi: 10.1016/j.jksuci.2023.101866.
- [6] A. Nazir et al., "Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 10, p. 101820, Dec. 2023, doi: 10.1016/j.jksuci.2023.101820.
- [7] Md. M. Rashid et al., "Adversarial training for deep learning-based cyberattack detection in IoT-based smart city applications," *Computers & Security*, vol. 120, p. 102783, Sep. 2022, doi: 10.1016/j.cose.2022.102783.
- [8] D. Tiwari, B. S. Bhati, B. Nagpal, S. Sankhwar, and F. Al-Turjman, "An enhanced intelligent model: To protect marine IoT sensor environment using ensemble machine learning approach," *Ocean Engineering*, vol. 242, p. 110180, Dec. 2021, doi: 10.1016/j.oceaneng.2021.110180.
- [9] O. Abu Alghanam, W. Almobaideen, M. Saadeh, and O. Adwan, "An improved PIO feature selection algorithm for IoT network intrusion detection system based on ensemble learning," *Expert Systems with Applications*, vol. 213, p. 118745, Mar. 2023, doi: 10.1016/j.eswa.2022.118745.
- [10] S. Hamdan, S. Almajali, M. Ayyash, H. Bany Salameh, and Y. Jararweh, "An intelligent edge-enabled distributed multi-task learning architecture for large-scale IoT-based cyber-physical systems," *Simulation Modelling Practice and Theory*, vol. 122, p. 102685, Jan. 2023, doi: 10.1016/j.simpat.2022.102685.
- [11] Y. Cao, Z. Wang, H. Ding, J. Zhang, and B. Li, "An intrusion detection system based on stacked ensemble learning for IoT network," *Computers and Electrical Engineering*, vol. 110, p. 108836, Sep. 2023, doi: 10.1016/j.compeleceng.2023.108836.
- [12] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Anomaly based network intrusion detection for IoT attacks using deep learning technique," *Computers and Electrical Engineering*, vol. 107, p. 108626, Apr. 2023, doi: 10.1016/j.compeleceng.2023.108626.
- [13] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, "Anomaly-based intrusion detection system for IoT networks through deep learning model," *Computers and Electrical Engineering*, vol. 99, p. 107810, Apr. 2022, doi: 10.1016/j.compeleceng.2022.107810.
- [14] A. M. Araujo, A. Bergamini de Neira, and M. Nogueira, "Autonomous machine learning for early bot detection in the internet of things," *Digital Communications and Networks*, vol. 9, no. 6, pp. 1301–1309, Dec. 2023, doi: 10.1016/j.dcan.2022.05.011.
- [15] S. Ali, Q. Li, and A. Yousafzai, "Blockchain and federated learning-based intrusion detection approaches for edge-enabled industrial IoT networks: a survey," *Ad Hoc Networks*, vol. 152, p. 103320, Jan. 2024, doi: 10.1016/j.adhoc.2023.103320.
- [16] O. Kornyo et al., "Botnet attacks classification in AMI networks with recursive feature elimination (RFE) and machine learning algorithms," *Computers & Security*, vol. 135, p. 103456, Dec. 2023, doi: 10.1016/j.cose.2023.103456.
- [17] M. M. Alani, "BotStop : Packet-based efficient and explainable IoT botnet detection using machine learning," *Computer Communications*, vol. 193, pp. 53–62, Sep. 2022, doi: 10.1016/j.comcom.2022.06.039.
- [18] X. Sáez-de-Cámara, J. L. Flores, C. Arellano, A. Urbieto, and U. Zurutuza, "Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks," *Computers & Security*, vol. 131, p. 103299, Aug. 2023, doi: 10.1016/j.cose.2023.103299.
- [19] P. L. S. Jayalaxmi, G. Kumar, R. Saha, M. Conti, T. Kim, and R. Thomas, "DeBot: A deep learning-based model for bot detection in industrial internet-of-things," *Computers and Electrical Engineering*, vol. 102, p. 108214, Sep. 2022, doi: 10.1016/j.compeleceng.2022.108214.
- [20] A. Aldhaheri, F. Alwahedi, M. A. Ferrag, and A. Battah, "Deep learning for cyber threat detection in IoT networks: A review," *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 110–128, Jan. 2024, doi: 10.1016/j.iotcps.2023.09.003.

- [21] M. Alipio and M. Bures, "Deep Reinforcement Learning Perspectives on Improving Reliable Transmissions in IoT Networks: Problem Formulation, Parameter Choices, Challenges, and Future Directions," *Internet of Things*, vol. 23, p. 100846, Oct. 2023, doi: 10.1016/j.iot.2023.100846.
- [22] R. K. Muna, M. I. Hossain, Md. G. R. Alam, M. M. Hassan, M. Ianni, and G. Fortino, "Demystifying machine learning models of massive IoT attack detection with Explainable AI for sustainable and secure future smart cities," *Internet of Things*, vol. 24, p. 100919, Dec. 2023, doi: 10.1016/j.iot.2023.100919.
- [23] H. Alshahrani et al., "Energy aware routing with optimal deep learning based anomaly detection in 6G-IoT networks," *Sustainable Energy Technologies and Assessments*, vol. 60, p. 103494, Dec. 2023, doi: 10.1016/j.seta.2023.103494.
- [24] N. Prazeres, R. L. de C. Costa, L. Santos, and C. Rabadão, "Engineering the application of machine learning in an IDS based on IoT traffic flow," *Intelligent Systems with Applications*, vol. 17, p. 200189, Feb. 2023, doi: 10.1016/j.iswa.2023.200189.
- [25] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing IoT network security through deep learning-powered Intrusion Detection System," *Internet of Things*, vol. 24, p. 100936, Dec. 2023, doi: 10.1016/j.iot.2023.100936.
- [26] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for machine learning-based intrusion detection in IoT networks," *Digital Communications and Networks*, Sep. 2022, doi: 10.1016/j.dcan.2022.08.012.
- [27] N. F. Syed, M. Ge, and Z. Baig, "Fog-cloud based intrusion detection system using Recurrent Neural Networks and feature selection for IoT networks," *Computer Networks*, vol. 225, p. 109662, Apr. 2023, doi: 10.1016/j.comnet.2023.109662.
- [28] M. Sarhan, W. W. Lo, S. Layeghy, and M. Portmann, "HBFL: A hierarchical blockchain-based federated learning framework for collaborative IoT intrusion detection," *Computers and Electrical Engineering*, vol. 103, p. 108379, Oct. 2022, doi: 10.1016/j.compeleceng.2022.108379.
- [29] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad, "HDA-IDS: A Hybrid DoS Attacks Intrusion Detection System for IoT by using semi-supervised CL-GAN," *Expert Systems with Applications*, vol. 238, p. 122198, Mar. 2024, doi: 10.1016/j.eswa.2023.122198.
- [30] E. Ntizikira, L. Wang, J. Chen, and K. Saleem, "Honey-block: Edge assisted ensemble learning model for intrusion detection and prevention using defense mechanism in IoT," *Computer Communications*, vol. 214, pp. 1–17, Jan. 2024, doi: 10.1016/j.comcom.2023.11.023.
- [31] C. Malathi and I. N. Padmaja, "Identification of cyber attacks using machine learning in smart IoT networks," *Materials Today: Proceedings*, vol. 80, pp. 2518–2523, Jan. 2023, doi: 10.1016/j.matpr.2021.06.400.
- [32] O. Habibi, M. Chemmakha, and M. Lazaar, "Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105669, Feb. 2023, doi: 10.1016/j.engappai.2022.105669.
- [33] S. A. Khanday, H. Fatima, and N. Rakesh, "Implementation of intrusion detection model for DDoS attacks in Lightweight IoT Networks," *Expert Systems with Applications*, vol. 215, p. 119330, Apr. 2023, doi: 10.1016/j.eswa.2022.119330.
- [34] T. Gaber, A. El-Ghamry, and A. E. Hassanien, "Injection attack detection using machine learning for smart IoT applications," *Physical Communication*, vol. 52, p. 101685, Jun. 2022, doi: 10.1016/j.phycom.2022.101685.
- [35] M. Abd Elaziz, M. A. A. Al-qaness, A. Dahou, R. A. Ibrahim, and A. A. A. El-Latif, "Intrusion detection approach for cloud and IoT environments using deep learning and Capuchin Search Algorithm," *Advances in Engineering Software*, vol. 176, p. 103402, Feb. 2023, doi: 10.1016/j.advengsoft.2022.103402.