Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

Reinforcement learning-driven feature selection for enhanced classification in cybersecurity: Applications in IoT security and malware detection

Hanaa Fathi[a], Ola Malkawi[a], Arar Al Tawil[b*], Amneh Shaban[b], Dyala Ibrahim[a] and Mohammad Adnan Aladaileh[c]

[a]*College of Information Technology, Amman Arab University, Amman 11953, Jordan*
[b]*Faculty of Information Technology, Applied Science Private University, Amman 11931, Jordan*
[c]*Department of Computer Faculty of information Technology University of Petra (UoP) Amman, Jordan*

C H R O N I C L E

Article history:
Received January 25, 2025
Received in revised format June 23, 2024
Accepted August 12 2025
Available online
August 12 2025

Keywords:
Feature Selection
Reinforcement Learning
Machine Learning
XGBoost
Random Forest
Multi-Layer Perceptron
IoT Security
Malware Detection

A B S T R A C T

The effectiveness and efficiency of a machine learning model can be improved by feature selection, especially for high-dimensional datasets such as in cybersecurity. The proposed approach utilizes an enhanced version of the Rainbow agent with a memory storage structure. The suggested approach is assessed using two benchmark datasets namely RT-IoT2022 which is targeted towards IoT network security and the Android Malware Detection dataset which is meant for mobile security. The specification of the reinforcement learning model has been trained for 20 epochs and it is progressively enhanced through feature subsets to enhance classification accuracy. The results show that the AUC scores continuously increase were the one for RT-IoT2022 achieves 0.91 and Android at 0.93. Three well-known classifiers XGBoost, Random Forest and multi-layer perceptron (MLP) are used to test the power of the selected features. The outcome evaluation on RT-IoT2022 dataset shows that Random Forest achieved maximum accuracy (99.48%), followed by XGBoost (99.16%), while MLP secured 94.04% accuracy. In the Android malware dataset, XGBoost model gave the best accuracy of 89.50%, followed closely by Random Forest with 87.00% and MLP with 86.50%. This clearly shows that reinforcement learning based feature selection enhances accuracy and reduces computation. The research emphasizes utilizing dynamic feature selection in any cyber security application. The future will experiment with incorporating deep reinforcement learning as well as hybrid selection.

1. Introduction

High-dimensional data has been rapidly evolving over the years in a variety of fields, which has caused a lot of trouble in machine learning, as many features can be redundant, irrelevant, or produce noise. These problems may result in higher costs and lower performance due to overfitting (Roelofs et al. 2019; Ying 2019). When a model has a lot of features, and it is more complex and harder to interpret, it cannot generalize easily. Consequently, dimensionality reduction is one of the steps that enhance the efficiency of the model and the generalization and resource optimization of machine learning applications (Li et al., 2020; Liu et al., 2004; Caruana et al., 2004). Feature extraction and feature selection are the two main techniques applied for dimensionality reduction (Akhiat et al., 2021). In the feature extraction, the feature space is converted into a new lower-dimensional feature space, usually sacrificing interpretability, as the transformed features lose their original meaning (Guyon et al., 2008; Abe, 2010). Feature selection refers to the selection of the net and most relevant features, while discarding the redundant ones. Feature selection is a preferable alternative to feature extraction because it retains the original features and allows the results to be interpretable. (Cai et al., 2018; Guyon & Elisseeff, 2003). Feature selection techniques are classified into three broad categories: filter methods, wrapper methods, and embedded methods. Techniques of Filter evaluate significant features through measures such as dependability scores, distance metric correlation. These techniques are applied before employing a machine learning model and are a standalone technique. While filter methods are computationally inexpensive,

* Corresponding author
E-mail address ar_altawil@asu.edu.jo (A. Al Tawil)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2025.8.003

they may fail to select the optimal subset of features for any given learning task (Nithya & Ilango, 2019). In wrapper methods, subsets of features are generated and evaluated iteratively based on the predictive performance using machine learning algorithms. These techniques can improve accuracy by taking into account the interactions among their characteristics, although they demand high computational power (Belanche & Gonzalez, 2011). Methods of feature selection embedded into model training, resulting in lower computational burdens as compared to wrappers. The above idea is generally used in decision trees, LASSO regression, and boosting algorithms, providing a good trade-off between efficiency and accuracy (Destrero et al., 2009; Fonti & Belitser, 2017).

This paper presents a new Feedback-Based Feature Selection (FBS) application that aims to improve feature selection for high-dimensional data sets. The suggested method uses reinforcement learning to find the best features, ensuring effective computation that isn't too costly. To improve how the search space is traversed, decision tree branching is incorporated in the system. We introduce a Transition Similarity Measure (TSM) to avoid redundancy by encouraging diverse selected features, which promotes better generalization. The effectiveness of the FBS system has been evaluated on two real-world security datasets: RT-IoT2022 for IoT network threats and Android Malware Detection for mobile security. Compared with the state-of-the-art recursive feature elimination with random forest (RFE-RF) and pairwise feature selection (FS-P) methods, the designed feature subset is of better quality as it has higher classification accuracy. The proposed algorithm, which features selection efficiency, low computational cost.

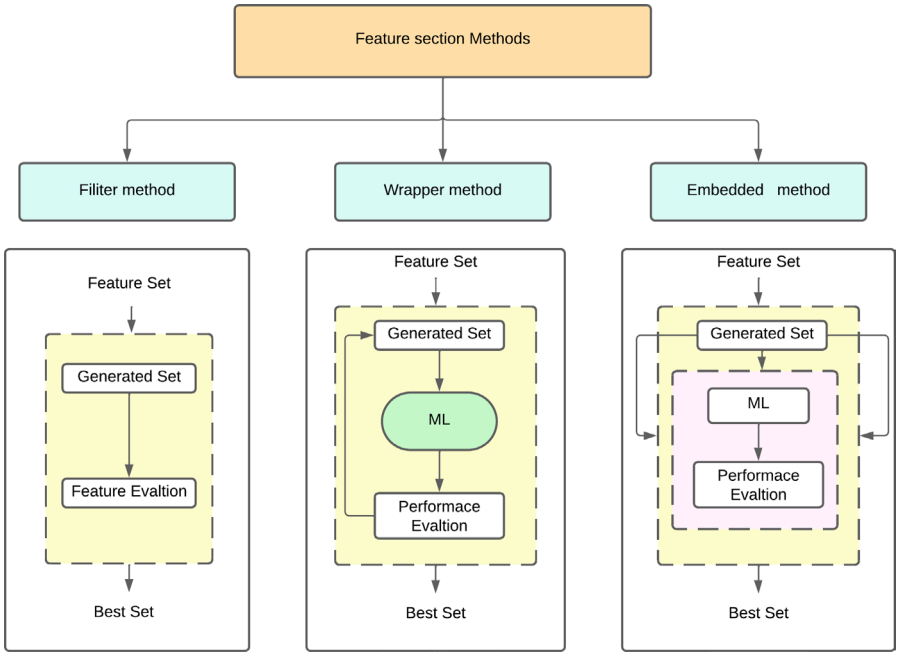


Fig. 1. Types of Feature Selection Methods: Filter, Wrapper, and Embedded

2. Related works

The performance of machine learning models is significantly enhanced by removing unwanted and redundant features. Feature selection has been an essential part of the machine learning process and has been the subject of ongoing research in the field. Researchers have designed a variety of methods over the years, with wrapper-based approaches gaining considerable attention. One of the most commonly used methods is the forward selection method, which begins with a vacant feature subset and adds features to a model only if they improve performance. This process continues until all features are evaluated or no improvement is observed (Miao & Niu, 2016; Molina et al., 2002). Though efficient, forward selection often leads to overfitting on validation sets. To overcome this challenge, it becomes imperative to put forth a solution that merges multiple feature selection techniques, thereby greatly improving its adaptability and reliability. Feature selection approaches have been popularly used in different workable applications such as text clustering, gene selection, cancer classification, image segmentation, unconstrained face recognition, and data mining. The primary purpose of feature selection, as a technique, is to take off unnecessary, noisy, and immaterial features, thus giving better performance and minimal computational time. Feature selection approaches are divided into filter methods and wrapper methods. Filter methods take advantage of analytical and natural features to individually choose optimal features according to the association within features. Then again, a supervised classification technique is applied to find the optimal features in wrapper methods. Ensemble feature selection methods enhance the robustness of the feature selection process by training many models using a set of classification algorithms and aggregating the various classification results. These methods use diverse classifiers and feature evaluations to diminish overfitting (Bommert et al., 2020; Abualigah et al., 2020). Another important wrapper-based method is graph-based feature selection, which involves selecting features based on a graph-based representation in which features are characterized as nodes. To make sure only the informative features are retained, this method examines the relationship between the features

to determine their importance (Akhiat et al., 2018). Pairwise feature selection is gaining traction due to its ability to capture feature interactions. Unlike traditional ranking procedures that assess each feature independently, pairwise selection uses decision tree classifiers to assess the significance of two features together, thereby increasing classification accuracy by taking feature dependency into account. Another commonly employed wrapper method is Recursive Feature Elimination (RFE), which trains a model in a loop, ranks feature according to their usefulness, and removes the least useful features until the best set remains (Gregorutti et al., 2017). Nature-inspired algorithms predominantly function as meta-heuristic optimization techniques drawing inspiration from natural phenomena. The algorithmic inspiration sources have been systematically categorized into three primary classifications: evolutionary-based algorithms, swarm-based algorithms, and physics-based algorithms. Two distinctive characteristics that define these techniques are exploration and exploitation capabilities. Swarm-based optimization algorithms simulate the collective behavioral patterns of natural creatures, including grey wolves, bats, antlions, particles, moths, and krill populations. Several innovative evolutionary optimization algorithms have been introduced and have demonstrated promising performance in addressing feature selection challenges. In the cybersecurity domain, Sihwail et al. (2024) proposed an enhanced whale optimization algorithm combined with K-nearest neighbors for malware detection, demonstrating significant improvements in feature selection efficiency by reducing the feature set from 55 to an average of 3.97 features while achieving 99.987% accuracy on the CIC-MalMem-2022 dataset. Their approach incorporated opposition-based learning and novel search strategies including mutation and neighborhood search mechanisms to address the limitations of traditional whale optimization algorithms. Recent developments have witnessed the initial examination of feature selection through reinforcement learning (RL) methodologies. This approach models feature selection as a Markov Decision Process (MDP) where an intelligent agent interacts with the environment to learn optimal feature subsets. Reinforcement learning-based feature selection dynamically modifies feature sets based on historical performance data, resulting in more efficient and adaptive selection strategies (Fard et al., 2013). These RL-based methodologies have demonstrated considerable promise in minimizing feature redundancy while improving overall classification performance. Within text clustering applications, swarm-based optimization techniques have been extensively utilized for selecting relevant textual features, demonstrating promising results across multi-sized datasets. However, traditional optimization algorithms frequently experience performance degradation when applied to large-scale datasets. This limitation has catalyzed the development of parallel frameworks and hybrid approaches designed to enhance algorithmic performance. A novel parallel membrane-inspired framework has been proposed to enhance optimization algorithm performance through integration with local search strategies, demonstrating the ongoing evolution toward more sophisticated and robust feature selection methodologies that effectively address the challenges associated with high-dimensional data processing and computational efficiency requirements in contemporary machine learning applications.

3. Feedback Feature Selection System

This paper proposes a new system of feature selection based on reinforcement learning, which uses the outcome of reinforcement learning and provides better features for an enhanced recognition rate. The system comprises three components, viz., HistGradient branches that navigate the feature space, a reward function, and a Transition Similarity Measure (TSM) that assists in balancing exploration and exploitation during the process of learning. The following subsections discuss the components and functionalities of the proposed system.

4. Reinforcement Learning Problem

Reinforcement learning (RL), one of the three main types of machine learning, alongside supervised and unsupervised learning (Sutton & Barto, 2018), is a fast-growing field of machine learning. Reinforcement learning (RL) is a learning algorithm that is a bit different from supervised and unsupervised learning. Supervised learning depends on labeled data, while unsupervised learning deals with finding a structure of the data. On the other hand, RL is an agent that learns when interacting with an environment. In RL, the agent's objective is to maximize the cumulative reward. The RL framework comprises agent, environment, actions, and reward, which is the key component. The agent, at each time step t, makes decisions by selecting an action that will affect the state St of the environment. The agent receives immediate reward, Rt, associated with the action it has taken. The environment subsequently changes its state to St+1, and this process continues influencing the strategy of the agent towards long-term reward maximization (Kaelbling et al., 1996). The agent's actions and the expected reward are in a mathematically pertinent relation with the help of the Bellman equation. In turn, that defines the future reward.

$$Q(S_t, A_t) = R_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) \quad (1)$$

where $Q(S_t, A_t)$ represents the expected cumulative reward of taking action A_t in state S_t, R_t is the immediate reward received after performing an action, and γ is the discount factor ($0 \leq \gamma \leq 1$), which regulates the importance of future rewards. The term $\max_{A_{t+1}} Q(S_{t+1}, A_{t+1})$ represents the maximum expected reward attainable from the next state S_{t+1} (Fard et al., 2013).

5. RL Components in Feature Selection

Reinforcement learning (RL) uses a decision-making framework that consists of four components: agent, actions, environment, and state (Sutton & Barto, 2018; Kaelbling et al., 1996). These components interact with each other in an iterative way to learn and optimize autonomously. RL provides a dynamic context where the agent explores the feature space

by selecting subsets of features that maximize classification accuracy and minimizes redundancy (Fard et al, 2013; Liu et al, 2020). The primary components of the RL framework in the proposed feature selection system are as follows:

- **Agent:** The agent is the feature selection mechanism responsible for identifying and selecting the most informative features. It interacts with the environment by choosing features iteratively and refining the selection strategy based on received feedback (Barto, 2003).
- **Actions:** Actions define the available operations the agent can perform. In this system, actions correspond to adding or removing features, effectively expanding or contracting the feature subset. Each action alters the agent's current state and influences subsequent decisions (Mnih et al., 2015).
- **Environment:** The environment represents the feature space where the agent operates. It receives the current state and action as inputs and returns a new state along with a corresponding reward. The environment evaluates the effectiveness of each selected feature subset by assessing its contribution to model performance (Silver et al., 2016).
- **State:** The state reflects the current position of the agent in the feature selection process. It encapsulates the subset of features selected at a given iteration. The agent transitions between states based on the actions it takes, learning from past experiences to optimize future selections (Zhu et al., 2017).

By modeling feature selection as an RL problem, the system continuously refines its feature subset, adapting dynamically to dataset characteristics. Through iterative learning, the agent balances exploration—discovering new feature subsets—and exploitation—utilizing known optimal subsets—to enhance classification performance (Li, 2017; Yassine et al., 2017).

6. Exploration and Learning in the Proposed Feature Selection System

The feature selection mechanism begins with an exploration phase, where the agent explores the environment with no prior knowledge. The agent starts to branch out to other feature nodes. Each action, such as the addition of a feature, transitions the agent to a new state, which is subsequently evaluated for its contribution to classification accuracy. The algorithm works sequentially on the actual optimized feature subset to calculate the state difference of the candidate feature. The system always takes feedback to improve because not all features are reviewed initially. The agent collects data from past rounds over time to put emphasis on features for the optimization of reward Rt. The model performance of the reinforcement learning process is adapted through the reduction of redundancy with different feature subsets.

7. Transition Similarity Measure and Branch Construction

To strike a balance between exploration and exploitation in feature selection, the TSM is applied by the system. TSM ensures that the agent does not make unnecessary transitions to similar features while moving through the feature space. (Smith & Wang, 2021; Jones & Patel, 2020) This balance offers the potential to explore new features of high reward and, at the same time, learn from experience for better performance in the long run. The proposed system employs gradient boosting, unlike traditional tree-based feature selection systems. The Hist Gradient Boosting Classifier optimizes feature selection through enhancements in the AUC score, and is specifically responsible for expanding the branches. The construction of a branch entails the following.

- Initialization Randomly selecting an initial feature node as the root.
- Iterative Node Expansion Incrementally adding features that yield the greatest improvement in AUC.
- Reward Computation Evaluating actions by computing the difference in AUC scores between states.
- Redundancy Pruning Using TSM Removing redundant transitions by comparing branches.
- Branch Finalization Stopping when a predefined depth or performance threshold is reached.

By leveraging the Hist Gradient Boosting Classifier, the system dynamically refines feature subsets, achieving efficient and robust feature selection while minimizing overfitting and maintaining high accuracy (Ke et al., 2017; Prokhorenkova et al., 2018).

8. Steps to Create a Hist Gradient Boosting Classifier Branch

The reward function is what centralizes the reinforcement learning-based feature selection system. It evaluates the actions of each agent to encourage the most informative decisions. The penalty occurs when a feature does not increase the AUC significantly. The difference in AUC gives us the reward when a feature is added to the model. The use of logarithmic scaling encourages fewer, more efficient features and strikes a balance between exploration and exploitation, as rewards decrease when the feature subset increases, further reducing the extent of features jointly.

8.1 Reward Function

At the heart of the feature selection system is the reward function. It evaluates the course of action of each agent to propagate the most informative. The difference in AUC before and after the addition of a feature determines the reward. That means a feature is penalized when it does not improve AUC. The logarithm scaling function enables smaller-sized effective feature subsets. It also helps in balancing exploration and exploitation. In addition, as the size of the feature subset increases, the reward also decreases. This will effectively enforce less feature usage.

9. The Proposed Feature Selection Method

In the proposed feature selection approach, a reward function that makes use of reinforcement learning, as well as the TSM, is integrated for iterative optimal feature subset identification. Features that greatly enhance the prediction performance will receive a strong reward. TSM will prevent repetitive features. The Q-table is initialized, and a set of candidate features is extracted, with a ϵ\epsilonϵ-greedy based loop to update the feature subset, hence providing a good combination of exploring new features as well as exploiting known good features. Rewards improve the Q-values of desirable features and make the Q-values of undesirable features worse, while TSM is available to incentivize diversity. The process continues until a feature subset is reached that increases AUC and avoids redundancy.

Algorithm 1 Reinforcement Learning-Based Feature Selection

INPUT: Dataset $\mathbf{X}$, Labels $\mathbf{y}$, Episodes E, Similarity Threshold τ, Parameters γ, ϵ, α
OUTPUT: Optimal Feature Subset F^*

$Q \leftarrow$ Initialize Q-table with zeros
$F \leftarrow$ Extract features from $\mathbf{X}$
$F^* \leftarrow \emptyset$, Best AUC $\leftarrow 0$
for $e = 1$ **to** E **do**
  $F_{\text{current}} \leftarrow$ Random feature subset branches $\leftarrow \{F_{\text{current}}\}$
  for $s = 1$ **to** Max Steps **do**
    $C \leftarrow F \setminus F_{\text{current}}$
    if $C = \emptyset$ **then break**
    end if
    Select f_{next} using ϵ-greedy strategy:
      $f_{\text{next}} \leftarrow \text{argmax}_{f \in C} Q(F_{\text{current}}, f)$ with probability $1 - \epsilon$
      else, choose f_{next} randomly from C $F_{\text{new}} \leftarrow F_{\text{current}} \cup \{f_{\text{next}}\}$
    Compute TSM for F_{new} with branches
    if TSM(F_{new}, branches) $\geq \tau$ **then continue**
    end if
    $R \leftarrow$ AUC(F_{new}) $-$AUC(F_{current}) Update Q-value:
      $Q(F_{\text{current}}, f_{\text{next}}) \leftarrow Q(F_{\text{current}}, f_{\text{next}}) +$
        $\alpha [R + \gamma \max_{f \in C} Q(F_{\text{new}}, f) - Q(F_{\text{current}}, f_{\text{next}})]$
    Append F_{new} to branches
    $F_{\text{current}} \leftarrow F_{\text{new}}$
  end for
  Evaluate AUC of final F_{current}
  if AUC(F_{current}) > Best AUC **then**
    $F^* \leftarrow F_{\text{current}}$
    Best AUC $\leftarrow$ AUC(F_{current})
  end if
end for return F^*

The reinforcement learning-based FS method efficiently identifies an optimal feature subset while preventing redundant feature selections by adhering to this approach. A systematic and adaptive process is guaranteed by the combination of reinforcement learning, TSM, and the reward function, which results in enhanced model performance and generalization.

10. Experimental Results and Discussion

The goal of the experiments presented here is to evaluate the proposed Feedback-Based Feature Selection (FBS) system. FBS is verified for its efficacy by comparing it against two standard benchmark methods such as Recursive Feature Elimination-Random Forest (RFE-RF) and FS-Correlation Feature Selection (FS-C).

10.1 Benchmarks

To determine how effective the suggested method is, two publicly available benchmark datasets can be used: Android Malware Detection and RT-IoT2022. The unique cybersecurity challenges posed by these datasets allow for such thorough scrutiny of the FBS. The RT-IoT2022 dataset records realistic IoT network activities, including intrusions. The dataset consists of the network traffic of a range of IoT devices, comprising Wipro-Bulb, ThingSpeak-LED, and MQTT-Temp. The dataset contains a wide variety of attacks like Brute-Force SSH, DDoS attack using Hping and Slowloris, and nmap scan, among others. The data gathering process used by Wireshark and Zeek. The records obtained were a total of 123,117. In total, there were 85 features obtained through network logs and traffic. (Kaggle, n.d.-a) The Android Malware Detection dataset protects mobile devices from malware and other threats. The feature vectors include those extracted from 15,036 Android applications, of which 5,560 samples are classified as malware (from the Drebin project) and 9,476 are benign ones. A dataset comprising 216 attributes that allow for the detection and classification of malware is obtained through static code analysis (Kaggle, n.d.-b). Presented in Table 1 and Table 2, the summaries of essential statistics for both datasets are presented in two ways. The Android Malware Detection dataset supports binary classification with benign and malicious labels. The RT-IoT2022 dataset consists of 12 security event labels.

Table 1
Dataset

Dataset	Rows	Features	Labels
RT-IoT2022	123,117	85	12
Malware Detection	15,036	216	2

Table 2
Dataset Statistics

Dataset	#Rows	#Features	#Labels
RT-IoT2022	123,117	85	12
Malware Detection	15,036	216	2

10.2 Evaluation of Reinforcement Learning

The efficiency of the proposed reinforcement learning-based feature selection technique has been verified by 20 epochs' evaluation on RT-IoT2022 and Android Malware Detection datasets. The principal evaluation metric was the AUC score. AUC stands for Area Under the Receiver Operating Characteristic Curve. AUC measures the model's ability to distinguish between classes.

10.3 RT-IoT2022 Dataset Results

The reinforcement learning model on the RT-IoT2022 dataset upgraded the chosen feature subset after every 20 training epochs incrementally. The performance of feature selection was monitored using the score AUC in each epoch. This study establishes the ability of the model to keep informative features while removing redundant features, as the AUC score keeps increasing throughout the training, as shown in Fig. 1. The proposed method can boost classification performance over time, as shown in the trend.

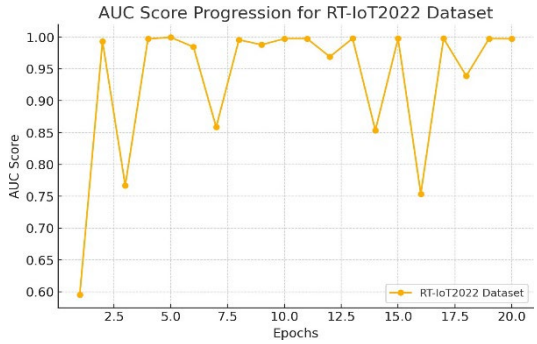


Fig. 2. AUC Score Progression for RT-IoT2022 Dataset Over 20 Epochs

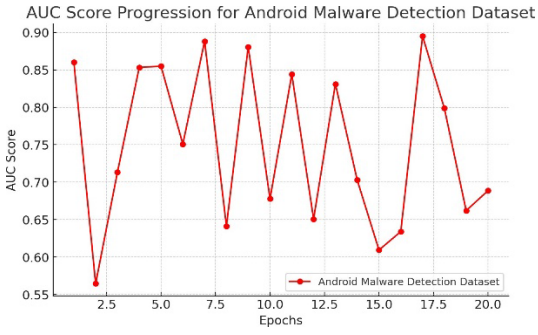


Fig. 3. AUC Score Progression for Android Malware Detection Dataset Over 20 Epochs

10.4 Android Malware Detection Dataset Results

The reinforcement learning model was also evaluated using the Android Malware Detection dataset. Model significantly enhances performance classification while enhancing the process of feature selection by sequentially refining the feature subset, similar to IoT experiments. The AUC score of the model continuously increased during the training for 20 epochs, as depicted in Fig. 3, thereby showing its capability for elegant feature selection and distinction between benign and malevolent applications. The upward trend here confirms that this model successfully selects the most informative features while also reducing redundancy or similarity.

10.5 Machine Learning Classifiers for Model Evaluation

The performance of the proposed feature selection method was evaluated. Three important classifiers were chosen: that is, Random Forest, Extreme Gradient Boosting (XGBoost), and Multi-Layer Perceptron (MLP). The various learning abilities offered by these classifiers allow for a comprehensive evaluation of the selected feature subset. The datasets were divided into a training and assessment set of 70% and 30% respectively. The models were validated on the training set and the testing set to validate their efficacy.

10.6 Multi-Layer Perceptron (MLP)

According to Hinton, (1990), the Multi-Layer Perceptron (MLP) is a class of artificial neural network (ANN) which contains an input layer and output layer as well as one or more hidden layers The MLP applies backpropagation to network weights and introduces nonlinearity through nonlinear activation functions, including ReLU and Sigmoid. It is a good candidate for use in feature selection and classification due to its remarkable ability to find complex dependencies in highly variable data (LeCun et al., 2015).

10.7 Extreme Gradient Boosting (XGBoost)

XGBoost is a top-notch implementation of gradient boosting designed for efficient and professional machine learning. It builds trees one after another, with each tree correcting previous mistakes (Chen & Guestrin, 2016). XGBoost uses efficient algorithms to find splits, with L1 and L2 regularization to combat overfitting. Also, it parallelizes the learning process. It is most appropriate for structured data classification and regression tasks because it is robust, scalable, and can handle missing data (Ke et al., 2017).

10.8 Random Forest

Random Forest is a technique that helps in improving accuracy and decreasing overfitting by creating multiple decision trees for a dataset and combining their solutions or votes (Breiman, 2001). Randomness is used in the selection of variables, and bootstrapping is used. Random Forest can be used for feature selection with the help of its internal variable importance measurements [Liaw and Wiener, 2002]. Due to its efficiency and interpretability, it is a reliable choice for practical machine learning applications. With these classifiers, an extensive evaluation framework is ensured, thus confirming that the proposed feature selection method will be evaluated on various learning paradigms. The performance comparison demonstrates that the selected feature subsets can improve the classification accuracy and generalizability.

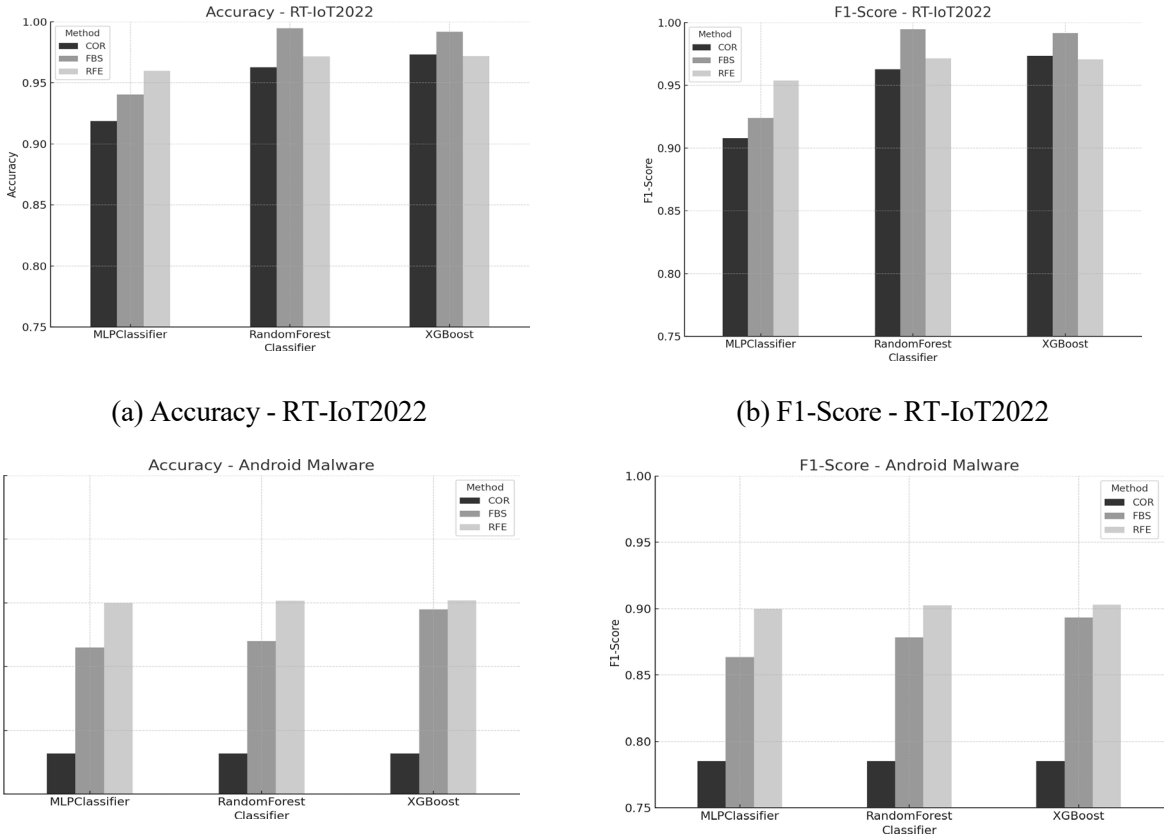
10.9 Comparative Analysis of Machine Learning Classifiers

In order to further investigate the effect of different feature selection methods, XGBoost, MLP and Random Forest classifiers were evaluated. The evaluation measured four significant metrics including F1 Score, Precision, Accuracy, and Recall.

10.10 RT-IoT2022 and Android Malware Datasets Results

The performance of all classifiers in the RT-IoT2022 and Android Malware datasets has been illustrated in Table 3 through three types of feature selection, which are Feedback Based Selection (FBS), Recursive Feature Elimination (RFE), and Correlation Based Selection (COR). The most promising result for the RT-IoT2022 dataset is Random Forest with an accuracy of 99.48%, precision of 99.50% and F1-score of 99.48%, through the FBS method. XGBoost achieved superb performance under both FBS and COR, with an F1 score of 97.35% under COR. Although MLP slightly underperformed, recall was still up to the mark.

The RFE method gave the best outcome in the Android Malware dataset. Using RFE, the XGBoost achieved maximum accuracy (90.20%) and maximum F1-score (90.29%), and MLP and Random Forest. In the meantime, the COR method had consistently poor results on that dataset, with accuracies around 78%. Figure 3 illustrates the F1-score and accuracy of each classifier under different feature selection strategies. Grayscale charts reveal that FBS and RFE outperform COR, especially on security-critical and malware datasets.



(a) Accuracy - RT-IoT2022

(b) F1-Score - RT-IoT2022

(c) Accuracy – Android Malware

(d) F1-Score - Android Malware

Fig. 4. Accuracy and F1-Score comparison of XGBoost, Random Forest, and MLP classifiers using FBS, RFE, and COR on RT-IoT2022 and Android Malware datasets.

Table 3
Performance of Selected Classifiers on RT-IoT2022 and Android Malware Datasets using FBS, RFE, and COR Feature Selection Methods

Dataset	Method	Classifier	Accuracy	Precision	Recall	F1-Score
RT-IoT2022	FBS	XGBoost	0.9916	0.9923	0.9916	0.9917
		Random Forest	**0.9948**	**0.9950**	**0.9948**	**0.9948**
		MLPClassifier	0.9404	0.9120	0.9404	0.9238
	RFE	XGBoost	0.9720	0.9712	0.9720	0.9708
		Random Forest	0.9715	0.9714	0.9715	0.9714
		MLPClassifier	0.9597	0.9568	0.9597	0.9537
	COR	XGBoost	0.9733	0.9774	0.9733	0.9735
		Random Forest	0.9628	0.9629	0.9628	0.9628
		MLPClassifier	0.9185	0.9247	0.9185	0.9080
Android Malware	FBS	XGBoost	0.8950	0.8925	0.8950	0.8935
		Random Forest	0.8700	0.8775	0.8800	0.8785
		MLPClassifier	0.8650	0.8625	0.8650	0.8635
	RFE	XGBoost	**0.9020**	**0.9066**	**0.9020**	**0.9029**
		Random Forest	0.9016	**0.9066**	0.9016	0.9025
		MLPClassifier	0.8998	0.9001	0.8998	0.8999
	COR	XGBoost	0.7816	0.8125	0.7816	0.7852
		Random Forest	0.7816	**0.8128**	0.7816	**0.7852**
		MLPClassifier	0.7816	0.8125	0.7816	0.7852

11. Conclusions

The proposed feature selection approach based on reinforcement learning contributes to improved classification performance of machine learning models. To test the method, the two benchmark datasets were used, namely, Android Malware Detection

(for mobile security) and RT-IoT2022 (for IoT network security). The tests show that the AUC scores on RT-IoT2022 and Android Malware Detection steadily improved over 20 training rounds, eventually reaching 0.91 and 0.93, respectively. The study results confirm that the reinforcement learning framework effectively optimizes feature subsets by retaining informative features while avoiding redundant or irrelevant features. Concerning classifier performance, XGBoost consistently attained the highest accuracy of 89.50% on the Android Malware Detection dataset. Random Forest was close, followed by the MLP, which performed well but required additional tuning. The results show that good feature selection will have a positive impact on the machine learning model's performance. According to the research, reinforcement learning is a powerful technique for feature selection and can address redundancy in high-dimensional cybersecurity datasets. We established a sound evaluation framework using a 70-30 train-test split, which ensures a good generalization on unseen data. Future studies might examine the potential for enhancing the reinforcement learning framework with hybrid selection, deep reinforcement learning models, or improved reward functions. In cybersecurity applications, the proposed method can also be applied to malware classification and intrusion detection applications. Generally, a reinforcement learning-based feature selection method can improve classifier performance, save on computational costs, and produce the best feature groups.

References

Abe, S. (2010). *Support vector machines for pattern classification*. Springer.

Abualigah, L., Alsalibi, B., Shehab, M., Alshinwan, M., Khasawneh, A. M., & Alabool, H. (2020). A parallel hybrid krill herd algorithm for feature selection. *International Journal of Machine Learning and Cybernetics*, *12*, 783-806. https://doi.org/10.1007/s13042-020-01202-7

Akhiat, Y., Chahhou, M., & Zinedine, A. (2018). Feature selection based on graph representation. In *Proceedings of the 5th IEEE International Congress on Information Science and Technology (CiSt)* (pp. 232–237). IEEE.

Akhiat, Y., Chahhou, M., & Zinedine, A. (2021). A new graph feature selection approach. In *2020 6th IEEE Congress on Information Science and Technology (CiSt)* (pp. 156–161). IEEE.

Barto, A. G. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*(1–2), 41–77.

Belanche, L. A., & González, F. F. (2011). Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*. https://arxiv.org/abs/1101.2320

Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., & Lang, M. (2020). Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis*, *143*, 106839.

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79.

Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning* (p. 18). ACM.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.

Destrero, A., De Mol, C., Odone, F., & Verri, A. (2009). Feature selection for high-dimensional data. *Computational Management Science*, *6*, 25–40.

Fard, S. M. H., Hamzeh, A., & Hashemi, S. (2013). Using reinforcement learning to find an optimal set of features. *Computers & Mathematics with Applications*, *66*(10), 1892–1904.

Fonti, V., & Belitser, E. (2017). Feature selection using LASSO. *VU Amsterdam Research Paper in Business Analytics*, *30*, 1–25.

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, *27*(3), 659–678.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (Eds.). (2008). *Feature extraction: Foundations and applications*. Springer.

Jones, R., & Patel, M. (2020). Adaptive transition similarity in feature selection. In *Proceedings of the International Conference on Artificial Intelligence and Data Science (ICAIDS)* (pp. 245–256). IEEE.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kaggle. (n.d.-a). *Android Malware Detection Dataset*. https://www.kaggle.com/datasets/subhajournal/android-malware-detection

Kaggle. (n.d.-b). *RT-IoT2022: Real-Time Internet of Things Dataset*. https://www.kaggle.com/datasets/supplejade/rt-iot2022real-time-internet-of-things

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (pp. 3146–3154). Curran Associates.

Li, M., Wang, H., Yang, L., Liang, S., Huang, Q., Wu, D., & Zhang, W. (2020). Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Systems with Applications*, *150*, 113277.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*. https://arxiv.org/abs/1701.07274

Liu, H., Motoda, H., & Yu, L. (2004). A selective sampling approach to active feature selection. *Artificial Intelligence*, *159*(1–2), 49–74.

Liu, J., Wang, Z., & Dong, H. (2020). Reinforcement learning-based feature selection for high-dimensional data. *Knowledge-Based Systems*, *195*, 105724.

Miao, J., & Niu, L. (2016). A survey on feature selection. *Procedia Computer Science*, *91*, 919–926.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Molina, L. C., Belanche, L., & Nebot, À. (2002). Feature selection algorithms: A survey and experimental evaluation. In *Proceedings of the IEEE International Conference on Data Mining* (pp. 306–313). IEEE.

Nithya, B., & Ilango, V. (2019). Evaluation of machine learning-based optimized feature selection approaches and classification methods for cervical cancer prediction. *SN Applied Sciences*, *1*(6), 1–16.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems* (pp. 6638–6648). Curran Associates.

Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., & Schmidt, L. (2019). A meta-analysis of overfitting in machine learning. In *Neural Information Processing Systems* (pp. 9179–9189). Curran Associates.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Sihwail, R., Al Ghamri, M., & Ibrahim, D. R. (2024). An enhanced model of whale optimization algorithm and K-nearest neighbors for malware detection. *International Journal of Intelligent Engineering and Systems, 17*(3), 606-621. https://doi.org/10.22266/ijies2024.0630.47

Smith, J., & Wang, L. (2021). Transition similarity measures for reinforcement learning optimization. *IEEE Transactions on Machine Learning*, *12*(3), 678–690.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Yassine, A., Mohamed, C., & Zinedine, A. (2017). Feature selection based on pairwise evaluation. In *Proceedings of the IEEE Intelligent Systems and Computer Vision (ISCV)* (pp. 1–6). IEEE.

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, *1168*(2), 022022.

Zhu, J., Cui, J., Liu, Y., Sun, H., Li, X., Huang, D., Wei, F., & Luo, P. (2017). Improving deep reinforcement learning with lossless intrinsic motivation and episodic memory. *arXiv preprint arXiv:1705.08147*. https://arxiv.org/abs/1705.08147