International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I11P116

Volume 73 Issue 11, 208-226, November 2025

Original Article

An Integrated Approach for AI-Generated Phishing URL Detection Using Reinforcement Learning and Machine Learning

Sharvari Patil[1], Narendra M. Shekokar[2]

[1]*Information Technology, Dwarkadas J. Sanghvi College of Engineering, Maharashtra, India.*
[2]*Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Maharashtra, India.*

[1]*Corresponding Author : sharvarichorghe@gmail.com*

Received: 13 May 2025 Revised: 03 November 2025 Accepted: 10 November 2025 Published: 25 November 2025

Abstract - *Among various cyber-attacks in this era of cyber advancement, phishing remains a momentous attack despite unprecedented technological advancements during the past few years. This problem becomes more concerning in view of the exponential rise in users across social platforms, necessitating a sophisticated method to assess web vulnerabilities. The prime mode of phishing attacks is generating URLs through generative AI, which may be misinterpreted as genuine URLs. Hence, it is imperative to devise a model that can differentiate between genuine URLs and AI-generated URLs. The proposed methodology combines Machine Learning and Reinforcement Learning, ensuring continuous learning based on the experiences. The reinforcement learning agent dynamically selects the feature subset using the Q-learning algorithm, and the classification algorithm is also decided at run time. Further, in order to validate the efficiency of the proposed model, a component is developed that generates URLs using AI. During the experimental evaluation, it is observed that the proposed model yields an accuracy of 99.25% outperforming state-of-the-art models. Thus, the proposed model can be widely used to classify AI-generated URLs from genuine URLs at large.*

Keywords *- Advanced Phishing Technique, AI-generated URLs, Cyber-Attack, Internet Security, Reinforcement Learning.*

1. Introduction

Revealing sensitive data or critical online information through social engineering and technical skills to deceive internet users is defined as Phishing. The first phishing attack was launched in the mid-1990s on the American Online (AOL) users using a software named AOHell [1]. This was a Windows application that comprised a method for obtaining the passwords of America Online users. The term phishing was coined from this AOL attack. This attack marks the dawn of phishing attacks, where the automated software was used to send enticing trap messages and logging responses. In this attack, official-looking screen names to mimic AOL staff were created. With the technical advancement, these attacks are becoming intelligent with the usage of Artificial Intelligence (AI). Financial benefits are the primary source of motivation for attackers; thus, they dynamically evolve their strategies to launch an attack.

The third quarter report published by the Anti-Phishing Working Group has reported 932,923 successful phishing attempts. This indicated a growth in the count of outbreaks reported in the Q2 report, which documented 877,536 phishing incidents [2]. Figure 1 shows the rise in unique phishing websites in quarters Q2 and Q3 of 2023, as reported by the APWG in the reports. In September 2024, the highest number of 342092 phishing links was reported. The reports stated that the social media platforms were the most attacked sector, resulting in 30.5% of phishing attempts. Smishing saw a 22% rise in Q3, and Gmail accounts were involved in 83.1% of Business Email Compromise frauds [2].

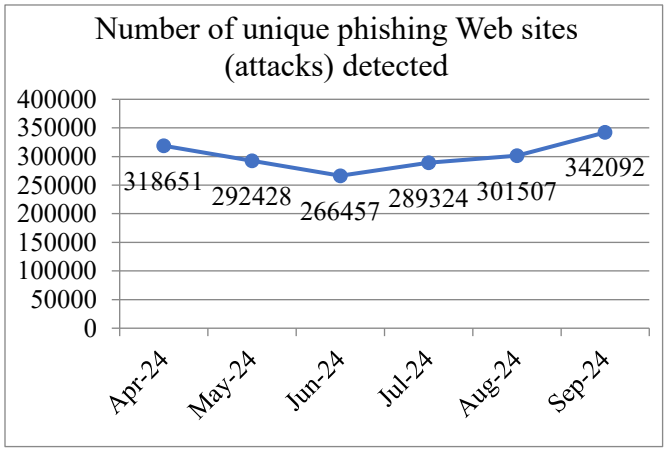


Fig. 1 Count of phishing websites detected as per the Q3 report

The Internet Crime Report [3] by the Federal Bureau of Investigation stated that Phishing has maintained its position in the top 5 cybercrime categories for the last 5 years [3]. In spite of the vast research to detect and prevent this attack, there is a continuous rise in the statistics published in the reports. The upsurge in Internet usage by a varied range of age groups of users is one of the major factors for redirecting the attackers to Phishing. Humans are the weak link in a security defence mechanism against cyberattacks.

Recently, research shows that the attackers are migrating to the use of AI for successful phishing attacks [4-6, 18]. A character-level text generation technique is employed for the generation of the AI-based URLs. Designing a phishing attack comprises two steps including the generation of a URL and the creation of a website that looks exactly the same as the genuine one. In [6], the authors have generated the URLs using the AI model. The model learns character-level and word-level structures that are common in phishing URLs. The model generates URLs by predicting one character at a time using a Recurrent Neural Network (RNN). The URL is then generated by performing filtering and then combining the domain with the path. The researchers in this paper demonstrated that the AI-powered attacks were able to bypass the ML-based detection system.

Classical techniques for phishing detection can be categorized into static approaches, heuristic-based methods, graphical similarity, machine learning, deep learning, and hybrid methods [7]. The static approach is completely based on the dataset of blacklisted URLs for the identification of fake websites. A list of blacklisted and white-listed URLs is maintained in the dataset. The URL is classified based on this static dataset as legitimate or malicious. These methods are computationally efficient but fail in the case of a zero-day phishing attack. In addition to this, for better defence, these datasets need to be updated at very short intervals of time.

A heuristic-based mechanism is a rule-based detection system. These rules are defined on the attributes that distinguish the URLs as phishing or genuine. These features are determined using the content of the website or the visual features. The features may include URL features like the domain age, structure of the URL, redirecting links, and content analysis. Visual similarity techniques use image processing for classification by comparing the target website with the genuine website image in the dataset. These techniques can detect the newly generated phishing URLs since they are not dependent on a dataset for the identification of the fake webpages. However, they have a high false positive rate and can be bypassed by the attacker by minimal modification in the URL [19, 20].

ML-based techniques train the ML model based on the features. The ML models are trained using a dataset including phishing and genuine URL samples. The trained model is further used for the classification of webpage URLs. The model training must be supported with appropriate data pre-processing and feature engineering techniques. Zero-day phishing attacks can be predicted using this technique. These techniques can be evaded by an AI-powered phishing attack.

Deep learning architecture is designed using neural networks, enabling it to uncover hidden patterns within complex data through a layer-by-layer learning process. Neural networks show great potential in phishing detection by recognizing patterns within webpages, URLs, and user behaviour. While these models offer good accuracy, they typically demand extensive computational resources and an exhaustive training dataset.

The hybrid approach integrates multiple classification techniques to enhance performance in detecting malicious websites. The techniques to be integrated are identified by evaluating the pros and cons of each methodology and understanding the nature of the dataset that will be employed to generate the model.

It is observed that the existing detection systems rely mainly on a static approach to detect fake URLs. The researchers have used machine learning to detect cyber attacks, but these systems have been proven to have failed to detect the AI-based phishing attack [6], with the rise in generative AI techniques being used to generate phishing URLs as reported by APWG in their 1st quarter report of 2025. There is a need to develop a system that adapts to these rapidly evolving attack strategies. ML and DL are the categories that previous studies have focused on to detect attacks. Very few researchers have used the adaptive nature of reinforcement learning algorithms to detect this evolving phishing attack.

The novel contribution of the proposed technique is the integration of RL combined with ML to employ the adaptive nature of the RL agent to optimize the performance of ML algorithms to detect the AI-based URLs. The proposed system uses a hybrid approach by integrating reinforcement learning and machine learning for detecting fake URLs. The innovative methodology of dynamic feature selection during runtime is implemented in the system. The proposed methodology comprises the following modules: Data Balancing, Feature Selection, Classification Model Selection, Classification Module, and Generation of fake URLs using AI to test the system using an approach recently employed by attackers.

The dataset was balanced before using it in the RL stage. During the feature selection stage, the RL agent selected random feature subsets of different sizes ranging from 10 to 15, to choose the optimal feature subset size and the best feature subset. The RL agent selects the classification model from the different classifiers, such as Logistic Regression (LR), Random Forest (RF), Gradient Boosting (GB) [11], and Decision Tree (DT) [11].

2. Related Work

Various techniques exist for the detection of phishing attacks using static lists, heuristics, and visual similarity. Researchers have proposed solutions to this problem using multiple AI-based algorithms. Anti-phishing solutions also use the program code of the target URL to extract its attributes and categorize the URL as fake or genuine. The ML-based solutions train the model using a dataset and test it based on the fixed, selected attributes. The deep learning methods use a neural network for the classification of phishing URLs. Each of these techniques majorly focuses on features selected, which include URL-based features or detecting fake webpages using image processing techniques. This section highlights some recent solutions proposed by researchers for phishing attacks.

2.1. Solutions Based on ML and DL

In [4], the researchers Sameen et al. have focused on the identification of synthetic URLs as proposed in [6], where it was stated that DeepPhish can launch phishing attacks by using AI techniques for phishing URL generation. The system is capable of detecting AI-generated and manually generated URLs with an ensemble machine learning model. They have also focused their work on detecting tiny URLs using an innovative approach named URLHit, where the tiny URL is transformed into an actual URL, which is then used for feature extraction and classification. A new lexical feature, URL HTML encoding, was introduced in this paper. Parallel execution of a machine learning ensemble model is employed for real-time classification. The system proposed in this paper could reach an accuracy of 98%.

In [8], the researchers have proposed a three-layered system including the Data Layer, the Model Layer, and the Stacking function. The Data Layer in the proposed hybrid framework is the foundational component that handles the acquisition and preprocessing of input data for phishing detection. This layer collects website features like URLs, HTML content, and DOM structures. The three models developed in the Model Layer are the URL-based model, HTML content-based model, and HTML DOM Tree-based model. These three models are combined using stacking functions like mean predictions, majority voting, most certain prediction based on confidence, DT, LR, and Neural Network.

Y. Bhanu et al. stated that the performance of the model is optimized by the use of dynamic feature selection so that it can adapt to the changing datasets. To implement the dynamic feature selection, the proposed system initially utilizes data balancing via a Conditional Wasserstein Generative Adversarial Network. The optimal features in this research are obtained by application of the Binary Grey Goose Optimization Algorithm. In the deployment phase, URL features were extracted using the Boosted ConvNeXt approach. These features were given as input to the trained classifier to classify the URL [9].

In [11], the researchers have proposed a hybrid phishing detection system that works on multiple ML algorithms. The proposed model was implemented using a combination of LR, support vector machine, and DT. Soft and hard voting technique was used for classification. The ensemble classifier combines the projected probabilities of each classifier for each class during the soft voting phase. The highest average probability determines the class label. In hard voting, the prediction is based on the majority of votes.

2.2. Solutions Based on Reinforcement Learning

In [12], the authors have addressed the unbalanced dataset for phishing classification. They have implemented a Double Deep Q-Network (DDQN) [12] classifier integrated with the Imbalanced Classification Markov Decision Process (ICMDP). The system was evaluated on the Mendeley dataset with 111 features and varying imbalance levels using the DDQN-based classifier. Additionally, the model eliminates the need for computationally expensive data-level balancing techniques, offering an efficient and robust solution for phishing detection. This research highlights the potential of cost-sensitive DRL techniques in inherently adapting to class imbalance, reducing computational overhead, and improving the reliability of cybersecurity systems.

In [13], the researchers have proposed a system called CETRA. This method dynamically selects classifiers based on prior outputs, balancing accuracy with computational efficiency. CETRA introduces an adaptive reward function that fine-tunes performance to meet predefined goals, such as TPR, FPR, and runtime, without requiring manual intervention. Additionally, CETRA enables policy transfer between different datasets. Experiments on the Bahnsen and Wang phishing datasets demonstrate that CETRA outperforms existing models, reducing processing time by up to 76% while maintaining F1-score degradation within 0.25% to 0.35%.

Gautam et al. have utilized the Q-learning algorithm to develop a system for detecting dynamically evolving phishing attacks [14]. This study uses a reinforcement learning framework with states represented by features retrieved from the URL. Predictions given by the agent are considered as actions. The agent is rewarded based on the accuracy of its predictions. The model follows a state-action-reward paradigm, extracting 111 URL-based features to make classification decisions and update its Q-table using the Bellman equation. Trained on a large dataset of 58,645 samples, it outperforms traditional machine learning models, achieving lower mean squared error and higher cumulative rewards. The model continuously learns and adapts to new phishing techniques, offering improved accuracy, lower false positives, and scalability for real-time applications.

Haidar Jabbar et al. have implemented phishing detection using a deep Q-Network along with Reinforcement Learning.

The research achieved improved detection accuracy, a reduction in false negatives, and improved classifier performance. The methodology has primarily worked on the reduction of false positives by penalizing the RL agent for generating false positives. In this work, an environment is a dataset of emails, and the agents classify them based on the features [20]. Ariyadasa, S. et al. [21] have proposed a solution to phishing attacks named SmartiPhish that combines DL and RL. The model analyzes a webpage's URL and HTML content to estimate how likely it is to be a phishing site. This probability is then sent to the RL system, which makes the final decision by taking into account how popular the webpage is and what it has already learned about similar sites. The model achieved a detection accuracy of 96.40%.

2.3. AI-Based Phishing Attacks

In [6], the researchers present DeepPhish, an advanced AI-driven phishing URL generation framework designed to simulate how malicious actors could weaponize AI to bypass detection systems. The study examines over one million phishing URLs to identify the behavioral patterns and strategies employed by real-world threat actors. The authors cluster phishing attacks by similarity and select two threat actors with notable effectiveness. Using Long Short-Term Memory [6] networks, they develop a character-level sequence model that learns the structure of successful phishing URLs and generates synthetic URLs that mimic these patterns. The DeepPhish algorithm receives sequences of effective URLs, encodes them using one-hot encoding, and trains an LSTM model to predict character sequences. The model then generates synthetic URLs with altered degeneration parameters to create variations.

In [10], the researchers have worked on a dataset for the detection of phishing attacks that were deployed using Phishing Kits. The research proposed a methodology for collecting a dataset for the detection of Phishing kits deployed on the website. PhiKit is a dataset that includes 510 phishing kit examples, 859 phishing website attacks, 1141 legitimate URLs, and traces of a phishing campaign. The research has performed three experiments, including Familiarity Analysis, Detection and Classification of Phishing Kits in multiple classes. Familiarity Analysis was performed to identify relationships among phishing kits and understand patterns of kit usage and evolution. The experiment for the detection of phishing websites was conducted using a graph representation algorithm.

In [16], the researchers propose URLGEN, a novel framework that automatically generates malicious URLs using Generative Adversarial Networks, with the aim of simulating realistic phishing URLs to evaluate and improve detection systems. The approach involves training a GAN model where the generator learns to create phishing-like URLs from a latent space, while the discriminator differentiates between real and synthetic URLs.

To summarize the literature survey, the work done for phishing detection ranges from hybrid ML models to RL and DL. The researchers in [4] have worked on the detection of AI-generated URLs using multithreading for real-time classification. This approach is best suited for real-time classification. In [8], a hybrid detection model was proposed that combined multiple ML techniques to enhance robustness. It focused on six key factors for real-world applicability, like effectiveness, speed, scalability, adaptation, flexibility, and robustness. The PDSMV3-DCRNN framework proposed in [9] further improves detection accuracy to 99.21% by incorporating advanced feature selection, data balancing, and deep learning ensemble models. However, although these hybrid models enhance accuracy, they often fail to address resource efficiency and practical deployment in dynamic environments. RL has emerged as a promising direction for phishing detection. Maci et al. and Kamal et al. introduce RL-based methods that use reward-based learning to adapt dynamically. Lavie et al. extend this approach by implementing automatic hyperparameter tuning, reducing the computational cost of training RL-based phishing detectors. The major research gap that the proposed system has attempted to solve is phishing detection using the models trained and deployed on a predefined feature subset. This lack of adaptability of the current tools fails to deal with the new techniques of phishing attacks launched using AI-generated URLs or other strategies. The proposed RL-based system overcomes this limitation by dynamically selecting the relevant features, allowing it to adapt to new threats without requiring dataset modifications or retraining. Hybrid and ensemble-based phishing detection models often rely on a predetermined combination of classifiers without dynamically evaluating their performance. The proposed system intelligently selects the optimal classifier during each detection phase, thereby improving overall accuracy through real-time reinforcement learning optimization. Table 1 summarizes the survey by comparing recent phishing detection approaches studied in this research.

Table 1. Comparative analysis of recent phishing detection approaches

Reference	Key Contribution	Methodology	Strengths	Research Gaps
[4]	Testing on AI-generated URLs	Utilizes ensemble ML models, multithreading, lexical analysis, URL encoding, and a voting-based classifier	Real-time detection, efficient multithreading, capable of detecting AI-generated phishing attacks (DeepPhish)	Lacks a dataset balancing strategy, does not support dynamic model selection
[8]	Proposed a hybrid	Employs a stacking	High accuracy (97.44%),	No real-time

	phishing detection framework integrating multiple phishing detection techniques	ensemble combining URL, HTML, and DOM-based models, and evaluates adversarial robustness	resistant to bypass attempts, validated for real-world applicability.	performance evaluation, lacks dynamic classifier selection.
[9]	Introduced PDSMV3-DCRNN, an ensemble deep learning model optimized with feature selection and dataset balancing	Implements CWGAN[9] for data balancing, BGGOA[9] for feature selection, and integrates PyDS-MV3[9] and DCRNN.	Highest reported accuracy (99.21%), fast training time (0.11s), and effectively optimizes feature selection.	Lacks scalability evaluation, relies on fixed classifiers instead of adaptive model selection
[10]	Developed PhiKitA, a dataset for phishing kit attack analysis	Aggregates real-world phishing kits, providing metadata on attacker techniques and automation tools	Enhances dataset diversity, improves phishing kit-based detection	Lacks adversarial attack integration, not widely integrated into ML/DL models
[11]	Designed a hybrid ML-based phishing detection system focused on URL-based analysis	Combines lexical, host-based, and third-party features, integrates RF, SVM, and deep learning classifiers.	Better generalization than standalone models, effective URL-based classification	Does not adapt to evolving phishing techniques and lacks feature selection optimization.
[12]	DRL model to address class imbalance in phishing detection	Implements reward-based learning, dynamically adjusts class weights	Improves resilience against minority phishing classes, enables adaptive model updates	Does not address zero-day attacks, lacks real-time deployment evaluation
[13]	Developed an automated hyperparameter tuning framework for RL-based phishing detection	Utilizes Bayesian optimization and meta-learning for cost-efficient RL adaptation	Reduces computational cost, enhances model adaptability across datasets	Lacks real-world deployment evaluation, requires scalability testing
[14]	Designed a Reinforcement Learning (RL)-based phishing detection model	Develops an agent-based learning system, assigns rewards/penalties based on classification accuracy	Self-learning phishing detection, adaptive to evolving threats	Limited evaluation on large-scale datasets, lacks adversarial robustness testing

3. Proposed System

The proposed methodology includes reinforcement learning for automated feature selection, classification model selection, and learning a policy using a free Q-learning algorithm. The proposed system includes a module for generating AI-based phishing URLs. These URLs are used for testing the proposed system for phishing identification. The agent is trained to identify the best feature group and the classification model. This trained agent is used for real-time url detection of the URL. A learning component is added to the system to give feedback based on the current prediction and thus learn from these experiences. Figure 2 depicts the proposed detection framework using reinforcement learning.

3.1. Data Balancing Module

Balancing of the dataset is a crucial step for the phishing detection problem, since the phishing data is always less than the genuine data. The data balancing module is responsible for balancing the dataset that was used for the RL agent training. Mendeley Dataset [15] is used in this research. The data consists of features taken from a set of websites. There are 111 features, 96 of which are URL features, and 15 were extracted using Python code [15]. The dataset has two variations: dataset_full and dataset_small. The smaller version has a total number of 58,645 instances. The full dataset includes 88,647 data points with 30,647 samples categorized as fake and 58,000 samples categorized as authentic. We have used the full dataset in this study since it has more samples.

The dataset can be balanced using the standard oversampling and undersampling techniques. Oversampling generates synthetic minority class samples, whereas in the sampling approach, the majority class samples are removed from the dataset to get a balanced dataset. Hybrid data balancing techniques, such as ADASYN-ENN, were applied to the dataset. The results from this experimentation showed that ADASYN-ENN gave a balanced dataset of 47592 legitimate samples and 47647 phishing samples. This technique also reduced the total count of false negatives from 614 to 220 for the KNN algorithm and from 426 to 166 for

Random Forest before and after data balancing. This balanced dataset was further used for training the agent. For each sample in the minority class, ADASYN calculates how difficult it is to classify based on the number of majority class neighbours using the KNN algorithm. If a minority class sample has more majority class neighbours, it is considered harder to classify and thus requires more synthetic data. The final dataset has a better class balance, and the newly generated samples are concentrated in the regions where the decision boundary is most unclear. ADASYN can introduce noise if too many synthetic samples are generated. To address this issue, Edited Nearest Neighbour, an under-sampling technique, is used for cleaning noisy data and balancing imbalanced datasets. It removes samples from the majority class that are likely to be misclassified, leading to a cleaner decision boundary.

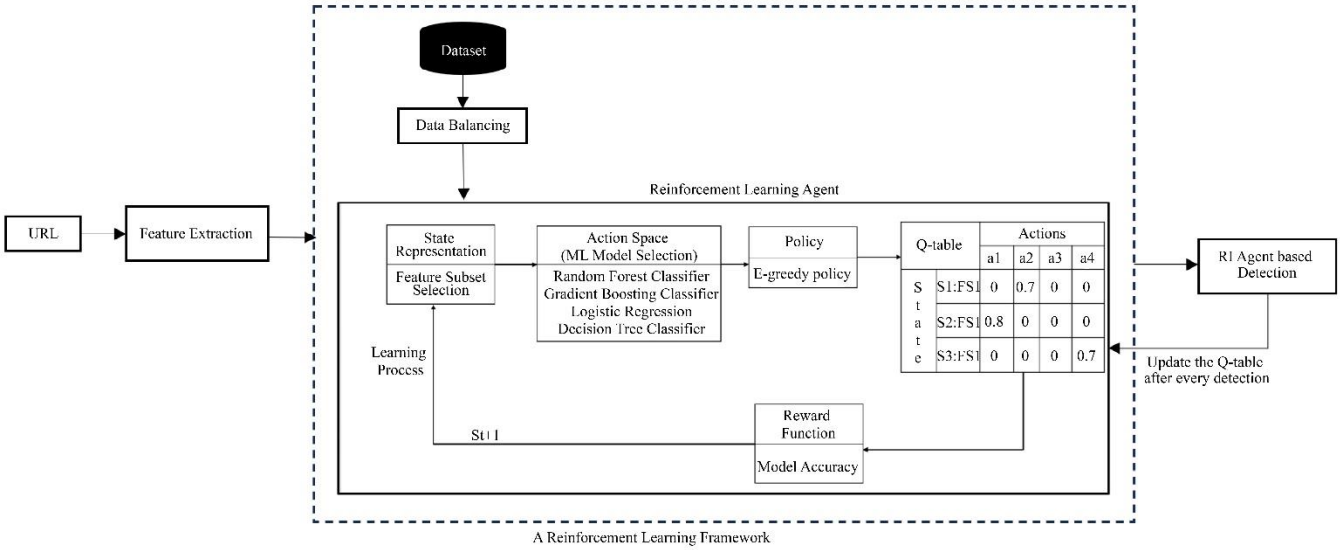


Fig. 2 Proposed RL-based system architecture

3.2. Overview of RL Framework

The component that has the prime responsibility of classification of the URL is the Reinforcement Learning Framework. The Reinforcement Learning approach enables the dynamic learning component based on experience. This improves the performance of the system. In the RL framework, the agent learns from a trial-and-error methodology. Agent, Environment, Policy, Reward, and Value Function are the important elements of the RL framework [17]. Figure 3 describes a generic representation of the RL framework.

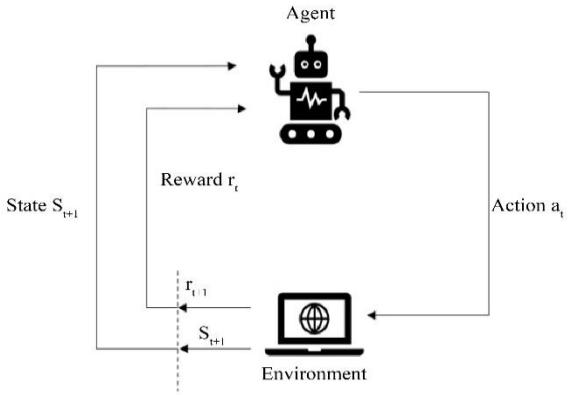


Fig. 3 Generic RL Framework

The agent and environment are the essential modules of RL [17]. The agent interacts with the environment by taking actions to obtain maximum reward [17]. The agent also explores the environment to learn better actions for future choices [17].

3.2.1. Agent

The reinforcement learning framework consists of a Q-learning Agent, and it calculates the Q-values to select the best action by selecting feature subsets. The RL agent in the proposed system is responsible for the selection of the best feature subset for phishing detection. The agent chooses the classification model dynamically at runtime, thus creating a black-box detection system. The goal of the RL agent is to maximize the accuracy over the training episodes by selecting different combinations of feature subsets.

3.2.2 Implementation of Q-Learning Algorithm for the Detection of AI-Generated URLs

This algorithm is a model-free RL algorithm. It is based on Q-values that are calculated using the action-value function. This function calculates the expected total reward. Q-value is the reward that the agent gets for taking action in a state. The goal of the algorithm is to choose the best action in each state to maximize total reward. The agent interacts with the PhishingEnvironment to learn an optimal policy. In the proposed system, the agent is responsible for simultaneously selecting the best feature subsets to maximize the

classification accuracy and choosing the best classification model dynamically to improve the performance. The optimal policy in the proposed system is subdivided into two aspects. One is a strategy that consistently picks the most effective feature subset to maximize phishing detection accuracy, and the other aspect is a classifier selection strategy that assigns the right model for the best feature subset. The optimal policy is represented using the formula (1), where s is the current state of the agent. The state of the Phishing Environment is the group of features that are selected. A is an action of selecting a new feature. The Q-learning values are represented by Q(s, a). These values indicate the expected future accuracy. The policy is to select the feature with the highest Q-value [22].

$$\pi^*(s) = arg\overset{max}{a} Q(s,a) \quad (1)$$

During training, the agent explores different feature subsets and gradually learns which subsets maximize the accuracy. Initially, the selection is at random with high exploration at ε=1.0. But over time, it exploits learned knowledge gained and a low exploration rate at ε=0.01. The Q-table stores the best feature subset selections and classifiers. Eventually, the optimal policy is obtained that includes the feature with maximum accuracy and the corresponding classifier. Equation (2) is used to calculate the future accuracy using the Q-value update Equation (2):

$$Q(s,a) = Q(s,a) + \propto [R + \gamma\, max_{a'}\, Q(s',a') - Q(s,a)] \quad (2)$$

Here, Q(s,a) is equal to the current Q-value for selecting feature a in state s. α is the Learning rate, which is set to 0.1. R is the immediate reward for classification accuracy. γ is the discount factor that decides how much future rewards matter and is set to 0.99.$max_{a'}\, Q(s',a')$ is the highest Q-value in the next state s′, which is the best expected future accuracy.

To summarize, the agent's role is to select the feature subsets. Based on the observation, take actions of selecting the features, learn from the rewards, and storing the optimal feature subsets in the Q-table. The agent also ensures that it balances exploration-exploitation using an ε-greedy strategy and improves its performance over episodes using Q-learning updates.

3.2.3. Environment

The environment represents the phishing detection problem, where the agent must learn to select features and a classification model to maximize accuracy. The main components of the environment are state, action, reward, transition function, Feature selection enforcement, classification model selection, and the reset function. The state space is defined in the environment as a binary feature representation. The actions in the environment are the selection of a subset of features. The environment transitions after every step. Certain actions are performed by the agent, including the selection of the feature subset based on which it receives the reward. Based on the reward, the agent updates its Q-values and refines its policy to maximize accuracy.

The environment is a component that provides a feature selection task, evaluates accuracy, and assigns the rewards. This setup enables the agent to autonomously learn an optimal feature selection strategy. Feature Selection ensures that all the features are used, and eventually, all 111 features are selected at least once. If there is any feature that has never been used, then the function replaces a used feature with an unused feature. The environment randomly selects a classification model from RF, DT, GB, and LR. The environment teaches the agent to find the best feature subset that gives maximum classification accuracy.

3.2.4. Reward and Learning Component

The reward function in the RL framework is defined as the accuracy of the selected feature subset with the chosen classification model. The agent selects 10 features from the 111 features in the dataset. The environment selects specific features from the dataset and divides them into training and testing sets. The classification model is randomly selected, and the accuracy score is used as the reward. The agent, therefore, will prefer feature subsets that gives higher accuracy in the future.

The learning component uses Q-learning to update Q-values based on rewards. Each episode consists of multiple steps where the agent selects a feature subset, trains the classifier, and gets accuracy as a reward. The updates in the Q-table are made using the Bellman equation [22] given in (2). The Q-table is updated as follows:

Step 1 : The Q-table is initialized with zeros. It stores Q-values for (state, action) pairs.
Step 2 : Choosing a feature subset. The agent selects 10 features from the dataset of 111 features.
Step 3 : Calculate the maximum future Q-value for the next state.
Step 4 : The Q-value for the selected action is updated. If the reward is high, the Q-value increases, making the action more likely to be chosen in future episodes.

The working of the learning component can be explained with an example. Initially, the learning rate α is 0.1, the discount factor γ is 0.99, and the exploration rate ε is 1, which decreases with each episode to ensure a balance between exploration and exploitation. In the first episode, the state is a random binary vector of length 111. It is stored in the Q-table with all Q-values set to 0. Since at this stage the agent has no prior knowledge, it must learn through exploration.

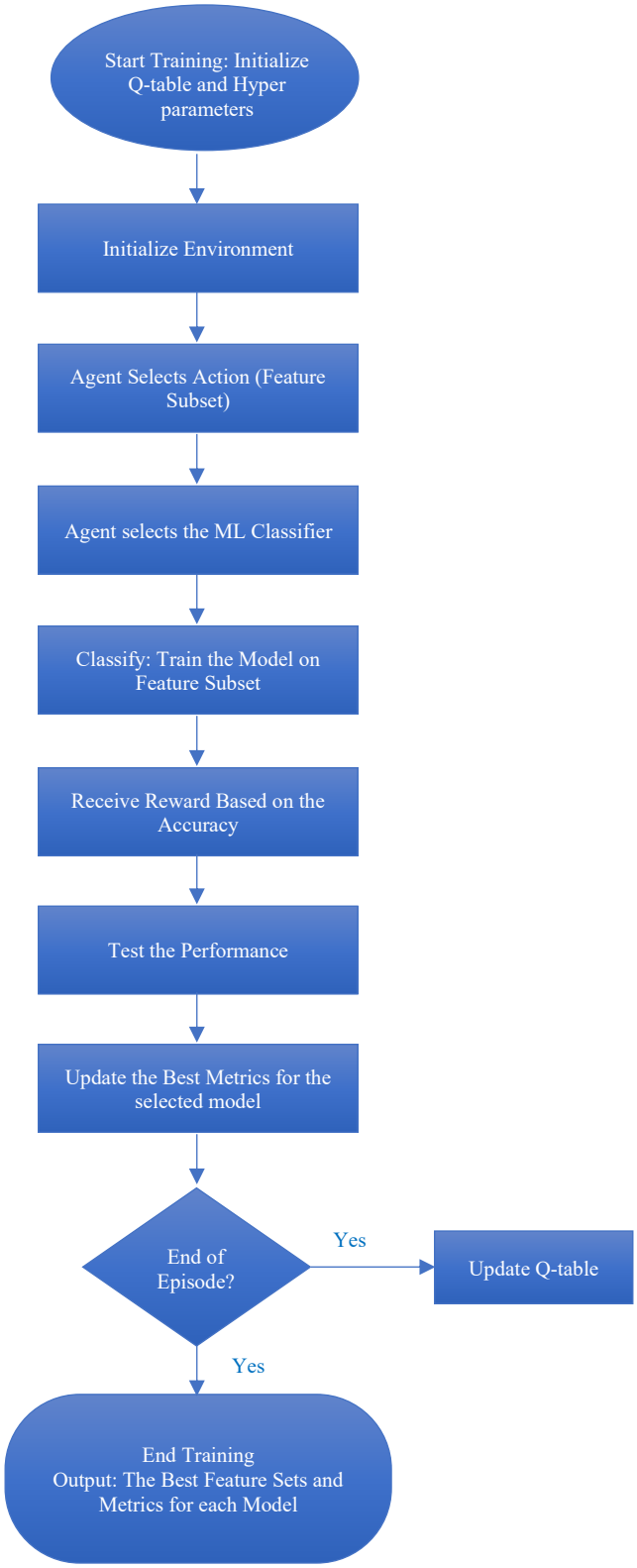


Fig. 4 Training model workflow

Example: Initial State for Episode 1

State = (1, 0, 1, 0, 1, 1, 0, 0, ..., 1, 0, 1) where 1 is a representation for a selected feature and 0 is for an inactive feature. The agent has no control at this initial stage, and it is randomly generated. In this step, the agent checks the Q-table for this state. Since it is episode 1, the Q-table is empty, and the new state is added to the Q-table with all the values initialized to 0. In the first episode, the ε value is 1; therefore, the agent explores by randomly selecting the feature subset and the classifier model. We assume that the accuracy that is equivalent to reward is 0.76. In this step, the Q-value is updated using equation (2). In the first episode, Q(s, a)=0, and max Q(S', a') is also zero since the Q-table is empty initially. Learning rate α is 0.1 and discount factor γ is 0.99.

$$New\ Q(s1, a1) = 0 + 0.1[0.76 + 0.99(0 - 0)]$$

$$New\ Q(s, a) = 0.076$$

After the episode is completed, the epsilon value decreases to 0.995.

Episode 2

A new state that is equivalent to a new feature subset is generated. This state is different from the state in episode 1. This state is added to the Q-table with all values initialized to zero. Since the epsilon value is still high, the state is selected randomly, and the agent is in an exploring state. The agent selects the classification model randomly. Based on the new state, the model accuracy is calculated, and the Q-value is updated.

$$New\ Q(s2, a2) = 0 + 0.1[0.82 + 0.99(0.076 - 0)]$$

$$New\ Q(s2, a2) = 0.1(0.82 + 0.07524)$$

$$New\ Q(s2, a2) = 0.0895$$

The epsilon value further decreases to 0.990.

Episode 3

Let us assume that at this stage, the agent starts exploiting the information stored in the Q-table. The new state is generated, and a feature subset is selected by the agent. At ε = 0.990, the agent mixes between exploration and exploitation. It chooses mostly from past Q-values but still explores. Let us assume that the agent selects the classification model and receives an accuracy of 0.78 as a reward. Max Q(S', a') is calculated from previous episodes. We will assume that Max Q(S', a') =0.0895, which is the best value from episode 2. Based on the new state, the model accuracy is calculated, and the Q-value is updated.

$$New\ Q(s3, a3) = 0 + 0.1[0.78 + 0.99(0.0895 - 0)]$$

$$New\ Q(s3, a3) = 0.1(0.78 + 0.0886)$$

$$New\ Q(s3, a3) = 0.0869$$

Table 2 below shows the values in the Q-table for 3 episodes.

Table 2. Q-table for phishing detection episodes

Episode	State	Action	Model	Accuracy (Reward)	Updated Q-value
1	State_1	[3, 7, 15, 24, ...]	Decision Tree	0.76	0.0760
2	State_2	[2, 5, 8, 19, ...]	Logistic Regression	0.82	0.0895
3	State_3	[1, 6, 10, 25, ...]	Random Forest	0.78	0.0869

3.3. AI-Based URL Generator Module

In this module, synthetic URLs are generated for testing the proposed methodology on AI-generated URLs. Random selection of SLD, GPT-2 text generation, and predefined patterns are the stages that contribute to the generation of URLs. The URLs used for this purpose are the phishing URLs from the PhishTank dataset. These URLs were segmented into their components, listed in Figure 5. Http or https defines the communication protocol used in the URLs. A subdivision of a main domain, used to organize different sections of a website, is called a Subdomain. SLD is the main part of a domain name, usually representing the organization or website. The last part of a domain name, often indicating the domain's purpose or country, is the TLD. Port is a number specifying the gateway for network communication. Figure 4 gives an overview of the structure of the URL.

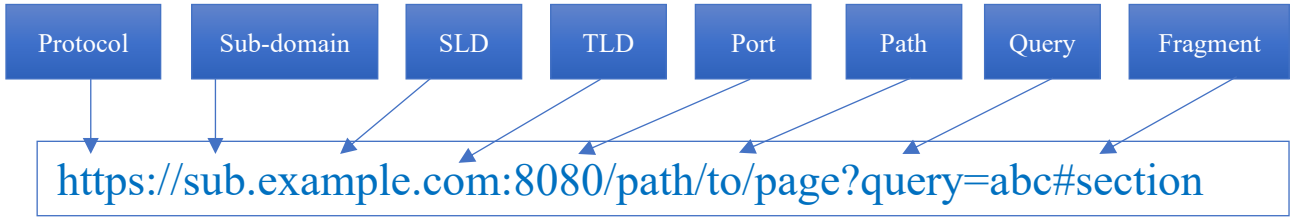


Fig. 5 General structure of the URL used for segmentation of the dataset

For the generation of the synthetic URLs, the system combines real-world data with AI-generated domains. This ensures that the synthetic URLs resemble actual web addresses. The proposed hybrid approach maintains realism while introducing enough variety for advanced testing scenarios. The URL generation starts with the dataset, including fake URLs. This dataset is further segmented to extract the components of the URLs to get a pre-processed dataset. This step ensures the generated URLs have a realistic domain structure. The domains in the generated URLs are either selected from the pre-processed dataset or they are generated using the GPT model. This is to ensure that there is a balance between AI-generated data and real data. Figure 6 is an overview of the AI-based URL generator module.

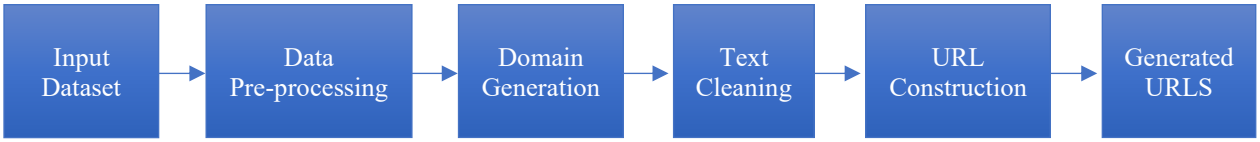


Fig. 6 AI-based URL generator module

GPT-2, a generative language model developed by OpenAI, is based on the Transformer architecture. It takes a real domain as a prompt and generates a new domain name. It is based on the transformer model, which includes the Self-attention mechanism, feed-forward Neural network, and Positional encoding [24]. Self-Attention Mechanism helps the model to understand the relationships between words in a sentence. Feed-Forward Neural Networks (FFN) are for processing the input through multiple layers for complex pattern recognition. Positional Encoding is required since transformers do not have recurrence. Therefore, positional encoding helps track word order. GPT-2 is made up of multiple transformer decoder blocks stacked on top of each other. Before passing text into GPT-2, it must be converted into a numerical format that the model understands. This is done using tokenization and positional encoding. The process of the GPT model is discussed as follows:

Tokenization and Positional Encoding

The input text, e.g., "www.example", is broken down into sub-word tokens using Byte Pair Encoding (BPE).

Input: "www.tech"

BPE Tokenized: ["www", ".", "tech"]

Token IDs: [1452, 25, 2765] (Unique numbers from GPT-2's vocabulary). These tokens are converted into unique IDs from a predefined vocabulary.GPT-2 uses positional encodings to indicate the order of tokens in a sentence. These position embeddings are added to the token embeddings before passing them to the model.

Token	“www”	“.”	“tech”
Position	1	2	3

Transformer Decoder Blocks

GPT-2 repeats multiple transformer decoder layers to process the tokens and generate new ones. Each decoder layer contains three key components: Masked Multi-Head Self-Attention, FFN, Layer Normalization, and Residual Connections [23].

Masked Multi-Head Self-Attention [23]

This is the important component of the model. It allows the model to focus on important words while generating text. The term “Masked" means it cannot look at future words. GPT-2 can only predict one token at a time based on previous ones.

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \quad (3)$$

Q (Query), K (Key), and V (Value) are the transformed input embeddings [23]. The SoftMax function gives higher scores to important words.

Input: ["www", ".", "tech"]

Attention: [0.2, 0.1, 0.7]

The word tech gets the highest importance.

Feed Forward Neural Network

Each FFN block consists of a First Linear Layer for expansion. This layer expands the input representation to a higher-dimensional space. In GPT-2’s Feed-Forward Neural Network, each token is represented as a vector of 768 dimensions. When this token representation enters the FFN, it goes through the first Linear Layer, which expands the vector from 768D to 3072D. The next layer is Non-Linear Activation. This layer is responsible for the addition of non-linearity. This allows the model to learn complex relationships. The final component in the FFN structure is the Second Linear Layer. This layer maps the transformed representation back to its original size. The expanded 3072D representation is compressed back to 768D.

After self-attention, the output passes through a fully connected neural network. FFN takes the output of the self-attention layer, processes it through two linear transformations with an activation function in between, and then passes the result to the next layer. The input to this network is a vector representation of a word or token.

This helps in understanding complex relationships between words. The following steps demonstrate the working of FFN:

Step 1 : First Linear Layer

"tech" is converted into a high-dimensional vector (e.g., 768D → 3072D).

The model learns more features about "tech".

Step 2 : Activation Function

The ReLU activation function adds non-linearity.

Example Effect: "tech" might become associated with "startup", "news", "hub",”ify”.

Step 3 : Second Linear Layer

The expanded 3072D representation is compressed back to 768D.

"tech" is now refined and ready for prediction.

Step 4 : Output: Next Token Prediction

Based on the processed "tech", GPT-2 predicts possible extensions:

“www.tech" → www.techify

"www.tech" → www.techhub

"www.tech" → www.technology

The hidden representation of the token "tech" when processed through FFN, predicts the word "ify".

Layer Normalization and Residual Connections

Layer normalization stabilizes activations. Prevents unstable activations and ensures a consistent feature distribution for each token. Residual Connection ensures that the original meaning is preserved. It helps to generate meaningful URLs instead of gibberish characters. Once the decoder has processed all the input tokens, the final layer predicts the next token using a SoftMax layer.

Step 1 : Input Token Embeddings

Assume that we have input as "www.tech”. GPT-2 needs to predict the next token.

Token	Initial Embedding
www	[0.1, 0.5, 0.3, ...]
tech	[0.7, 0.2, 0.4, ...]

Step 2 : Self-Attention

Self-attention computes how much each token influences others in the sequence. Here are the attention scores GPT-2 assigns for the next possible token:

Token	Attention Score
hub	0.3
ify	0.5
solutions	0.2

Interpretation from this step is that "hub" has the highest importance, meaning it is most contextually relevant.

Step 3 : Feed-Forward Network

The feed-forward network transforms token embeddings by expanding each token representation from 768D to 3072D to increase expressiveness.

This helps to model complex relationships between words. The ReLU activation function is applied to introduce non-linearity. The tokens are compressed back to 768D

Token	Transformed Representation (3072D → 768D)
hub	[0.5, 0.6, 0.4, ...]
ify	[0.8, 0.3, 0.5, ...]
solutions	[0.2, 0.7, 0.3, ...]

Step 4 : Residual Connection (Adds Original Input Back)

After FFN transformation, the model adds back the original "tech" embedding to keep contextual meaning:

Final Output=FFN Output+ Original Embedding

Token	Residual Output
hub	[1.2, 0.8, 0.7, ...]
ify	[1.5, 0.5, 0.8, ...]
solutions	[0.9, 1.0, 0.6, ...]

Step 5 : Layer Normalization

Layer normalization scales the outputs to keep training stable:

Token	Normalized Score
hub	0.25
ify	0.65
solutions	0.10

Step 6 : Final Token Selection

Now, GPT-2 chooses the next token using Softmax:

Token	Probability (%)
hub	25%
ify	65%
solutions	10%

Step 7 : Generate Full URL

Now that we have "techify", we combine it with:

A random subdomain (e.g., "www")

A TLD (e.g., "com")

A path (e.g., "/login")

Final URL:

This is the final URL that gets generated by the GPT model: https://www.techify.com/login.

The text generated by the model is cleaned such that it must match a domain-like pattern that includes letters and hyphens and 3-15 characters.

The final step is URL construction that combines domain components into a complete URL. This step creates synthetic URLs that resemble real-world website URLs.

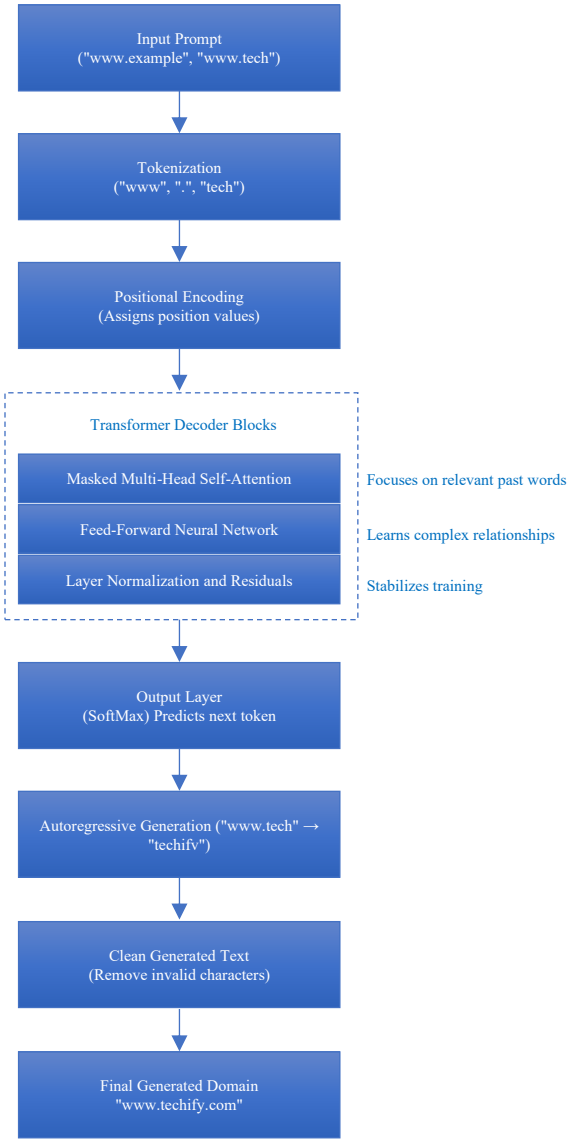


Fig. 7 Process flow diagram of GPT model for domain generation

3.4. Phishing Detection Processing Sequence

In this section, we will discuss the processing sequence of the phishing detection system. The Reinforcement Learning Agent is trained over 100 episodes to dynamically select the feature subset and the classification model. The Q-table from this training is used for real-time phishing detection. The features are extracted from the target URL and are converted into a state representation for the agent to take actions. This action includes the selection of a feature subset and the classification model. The classifier is trained on the selected subset, and the prediction result is given as output. Feedback is taken from the user regarding the correctness of the classification, and the agent is rewarded based on the user feedback. The positive or negative reward is further used to update the Q-table. This real-time feedback will help the agent to keep learning based on the experiences. The Q-learning-based phishing detection algorithm is given below.

Algorithm 1. Q-learning-based Phishing Detection

Input: URL dataset, Feature Extraction Module, Machine Learning Classifiers, Initialized Q-table, Exploration rate (ε),
Learning rate (α), Discount factor (γ)
Output: Phishing classification result and updated Q-table

Begin

Initialize Q-table Q(s,a)

For each training episode do begin
  Select a URL u from the dataset
  Extract features from Furl using the Feature Extraction Module
  Convert URL into state representation s
  Choose an action using ε-Greedy policy:

If random (0,1) <ϵ, select a random action
Else, select action a=arg max Q(s,a)
  Select feature subset Fs and classifier Ca based on action a
  Train classifier Ca on Fs.
  Predict phishing or legitimate using trained Ca.
  Get feedback:

If the user confirms correct classification → Assign positive reward r=1
Else → Assign negative reward r=−1
  Compute next state s′ (simple transition s′=(s+1) mod |Q|)
  Update Q-table using:
$Q(s,a) = Q(s,a) + \propto [R + \gamma \, max_{a'} \, Q(s',a') - Q(s,a)]$.. [23]
  Store updated Q-table
End for
Return phishing classification result
End

4. Experimental Results

The experiments on the proposed methodology are done in two stages. Initially, the RL Agent was trained and evaluated. Then, the testing of the system on the AI-generated URLs.

4.1. Experiments on Q-Learning-Based RL Agent

In this study, a Q-learning-based RL agent was trained for intelligent feature selection for the detection of phishing URLs. The custom environment was designed to allow the agent to select features randomly and receive a reward proportional to the classification accuracy obtained using a randomly selected classifier. The reward value is used to calculate the Q-values that are required for decision-making. The training was conducted over 100 episodes with 111 steps in each episode to ensure the selection of all the features. The epsilon decay value ensured that exploration gradually decreased and exploitation increased through epsilon decay. After each episode, the updated Q-table provided feedback to the agent's action-value estimates evolved over time. A sample of selected feature Q-values was also tracked across episodes.

The Q-table values indicated the agent's growing understanding of the environment. Initially, the Q-values were uniformly low due to random exploration. As episodes progressed, features contributing to higher classification accuracy received increasingly higher Q-values, while less informative features maintained lower values. This behaviour shows that the agent effectively learned to favour better feature selections. Figure 8 gives an overview of the Q-values over the episodes, which shows that there was a steady rise in the values starting low from the first episode to reaching 80 by the 100th episode. This shows that the agent was learning effectively over time, and the Q-table was being updated in a way that high-value actions are increasingly being reinforced. The graph indicates convergence and improved policy stability.

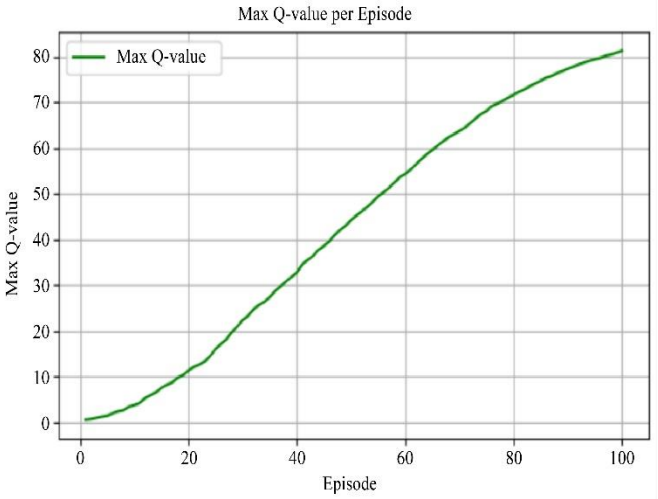
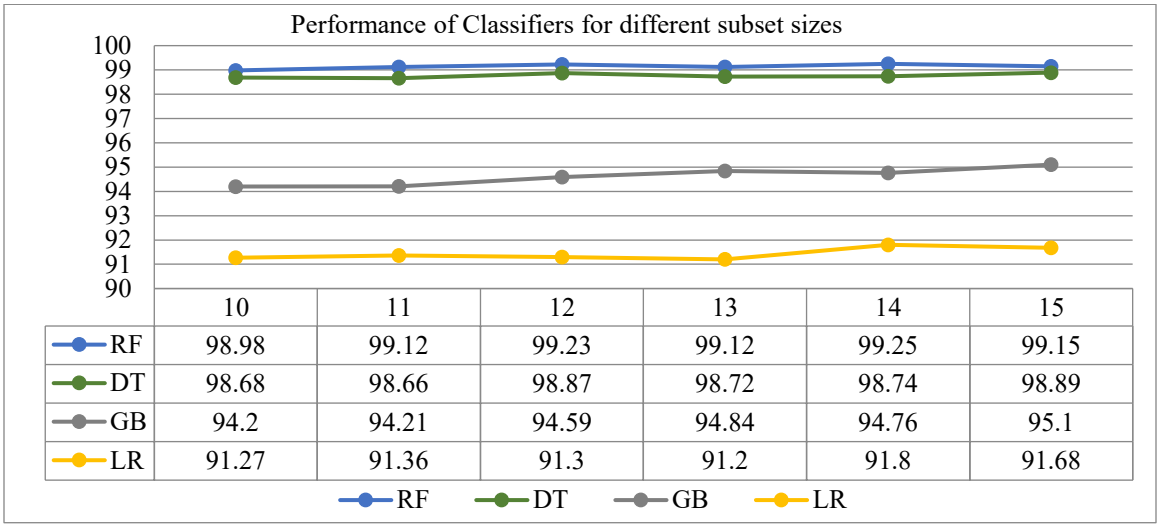


Fig. 8 Max Q-values over training episodes

Fig. 9 Best accuracy attained during the training episodes

Figure 9 shows the best accuracy achieved by the agent in each episode. It was observed that accuracy per episode fluctuated between 96.5% and 99% but mostly clustering around 98–98.5%. There was no strong upward trend, but it was generally stable at a high level. High accuracy indicated that the selected features were indeed effective for classification. The variability in the graph is due to the randomness in the feature selection strategy and classifier selection. However, because accuracy stays high, it asserts that good subsets are being consistently chosen, and the strategy is working correctly.

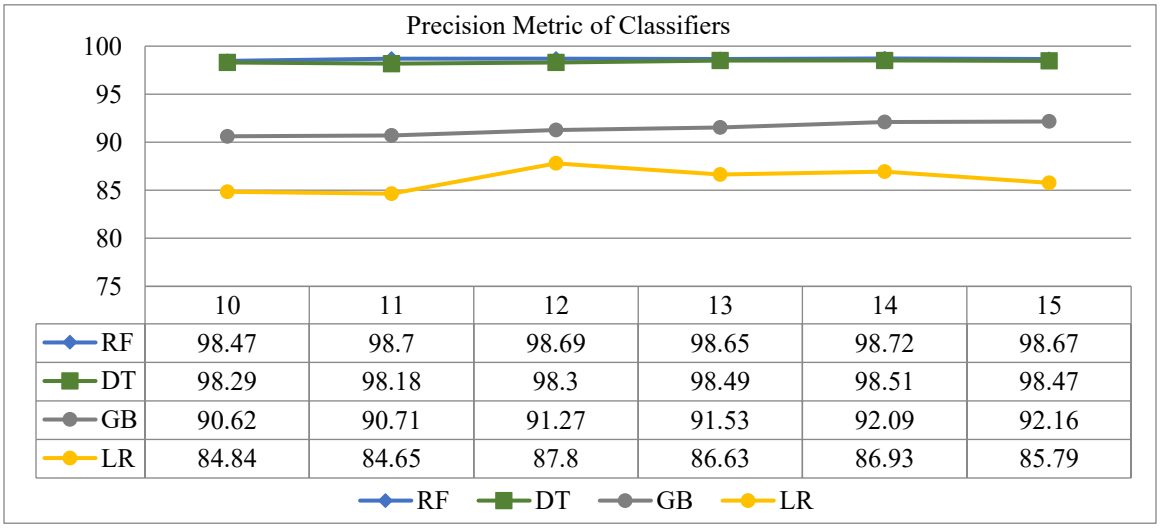
The Q-learning agent was trained on different feature subset sizes, and at the end of the training, the accuracies of the classification models for different subset sizes were obtained. This experiment was conducted to analyze how the feature subset size impacts the agent's performance. The performances of the different models for a subset size ranging from 10 to 15 are given in Figure 10. Accuracy, Precision, Recall, and F1-Score are calculated for the evaluation of the proposed system. The results are analysed to understand the impact of the reinforcement learning-based feature selection approach on the overall system performance.



	10	11	12	13	14	15
RF	98.98	99.12	99.23	99.12	99.25	99.15
DT	98.68	98.66	98.87	98.72	98.74	98.89
GB	94.2	94.21	94.59	94.84	94.76	95.1
LR	91.27	91.36	91.3	91.2	91.8	91.68

Fig. 10 Accuracies of the different models for a subset size ranging from 10 to 15

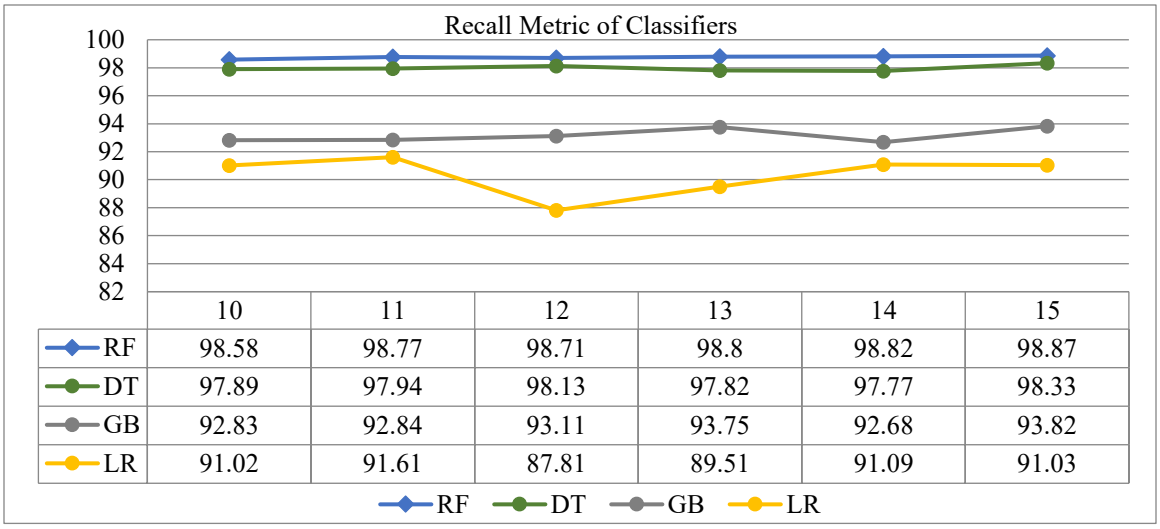
The model achieves high accuracy, indicating the effectiveness of the reinforcement learning approach in selecting the optimal subset of features. The consistent accuracy across multiple trials highlights the model's stability. The high precision score indicates that the model effectively minimizes false positives, classifying most phishing URLs correctly. This is crucial for phishing detection systems, as false positives can lead to unnecessary blocking of legitimate websites. Compared to baseline algorithms, the Q-learning model consistently achieves higher precision, confirming its capability in discriminating between phishing and legitimate URLs.



	10	11	12	13	14	15
RF	98.47	98.7	98.69	98.65	98.72	98.67
DT	98.29	98.18	98.3	98.49	98.51	98.47
GB	90.62	90.71	91.27	91.53	92.09	92.16
LR	84.84	84.65	87.8	86.63	86.93	85.79

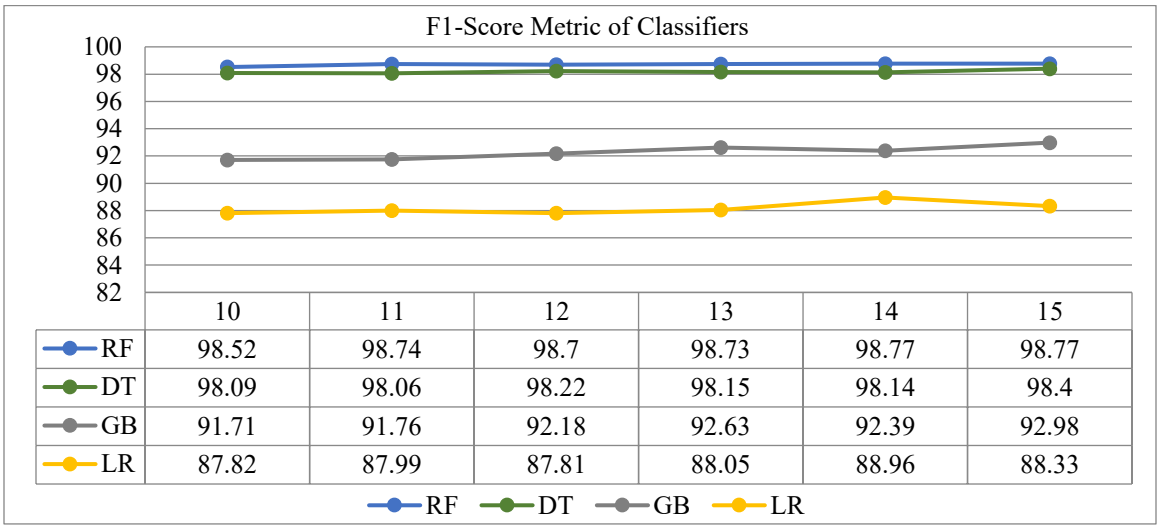
Fig. 11 Precision of the different models for a subset size ranging from 10 to 15

The recall score is depicted in Figure 12. The model demonstrates a high recall, suggesting its capability to identify the majority of phishing URLs. Despite this trade-off, Figure 13 shows the F1-score, which balances precision and recall. Higher F1-score values obtained during the experiment contribute to minimizing false positives and false negatives.



	10	11	12	13	14	15
RF	98.58	98.77	98.71	98.8	98.82	98.87
DT	97.89	97.94	98.13	97.82	97.77	98.33
GB	92.83	92.84	93.11	93.75	92.68	93.82
LR	91.02	91.61	87.81	89.51	91.09	91.03

Fig. 12 Recall of the different models for a subset size ranging from 10 to 15



	10	11	12	13	14	15
RF	98.52	98.74	98.7	98.73	98.77	98.77
DT	98.09	98.06	98.22	98.15	98.14	98.4
GB	91.71	91.76	92.18	92.63	92.39	92.98
LR	87.82	87.99	87.81	88.05	88.96	88.33

Fig. 13 F1-Score of the different models for a subset size ranging from 10 to 15

4.2. Experiments using AI-Generated URLs on the Agent

The agent trained in phase 1 was then used for testing the AI-generated URLs. Over 200 URLs were generated by the AI-based URL Generator Module. The features are extracted from the target URL, and a feature vector is generated that represents the URL. The Q-learning agent selects the features and classification model by hashing the URL to generate a state index. Hashing converts a URL into a numeric state index to fit into a fixed-sized Q-table. The agent predicts using the feature subset and the classification model. Based on the feedback, whether the prediction was correct or incorrect, the Q-table is updated with a positive or negative reward, respectively. The agent was given 200 AI-generated synthetic URLs, out of which it could correctly predict 190 URLs as phishing, giving an accuracy of 95%. Figure 14 shows the results obtained in this experiment.

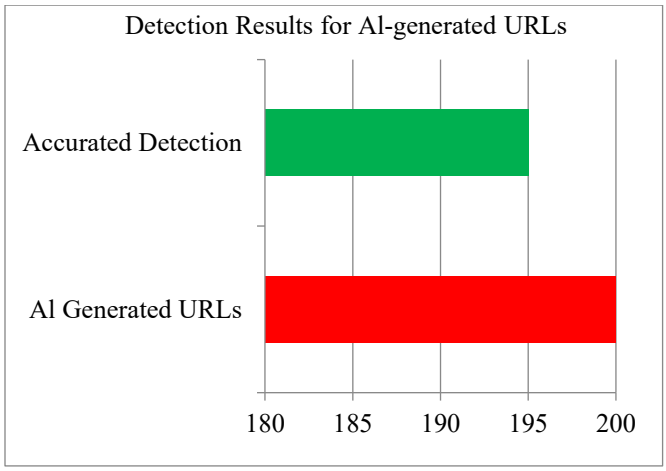


Fig. 14 Detection results for AI-generated URLs

This dataset of AI URLs was pre-processed to extract 111 features, the same as the features in the dataset used for training the RL agent. The dataset, including the AI-generated URLs, was used to retrain the agent. This experimentation gave the accuracy as shown in Figure 15. The results show that random forest gave the best accuracy of 98.16%. High accuracy for Random Forest indicates that it is effective at correctly classifying URLs as phishing or genuine. The Decision Tree is slightly lower but still performing well. The lower performance of Logistic Regression suggests that the selected features are not linearly separable, or the feature selection agent may be choosing feature subsets that are better suited for more complex models like Decision Tree, Random Forest, or Gradient Boosting.

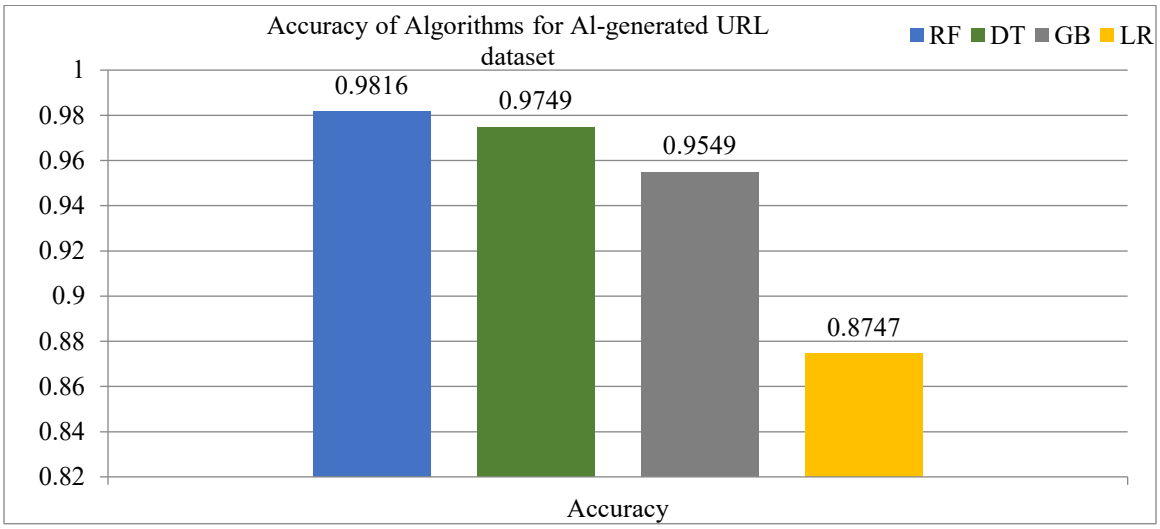


Fig. 15 Accuracy of the classification algorithms for AI-generated URL dataset

Figure 16 summarizes the results of the classification algorithm for the metric precision. Random forest and Decision tree algorithms gave comparable precision values of 98.27% and 98.25 % respectively. High precision indicates that when the model predicts a phishing URL, it is usually correct. However, overly high precision compared to recall suggests that the model may be conservative in predicting phishing URLs.

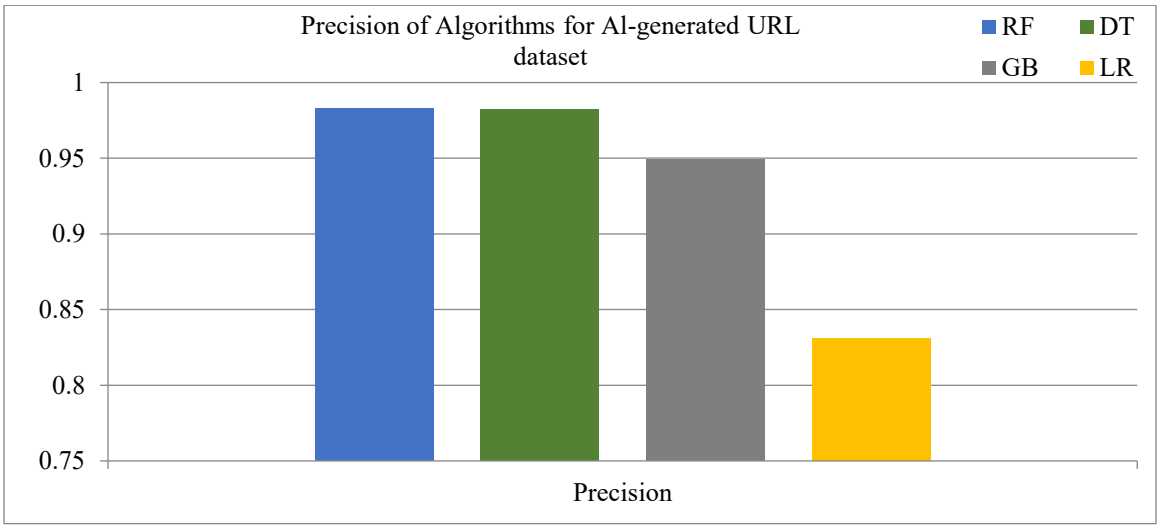


Fig. 16 Precision of the classification algorithms for AI-generated URL dataset

Experiments show that random forest and logistic regression gave the best recall values, indicating improvement in the reduction of false positives. Recall measures how many actual phishing URLs are correctly identified. A high recall indicates low false negatives, which is essential for detecting as many phishing URLs as possible. Random forest algorithm gave the best result for F1-score with 98.41% and comparable value of 97.82% values. High F1 scores imply good performance even when dealing with an imbalanced dataset. Since the environment penalizes episodes where accuracy drops, this metric indicates whether the agent is also indirectly learning to optimize for F1 Score.

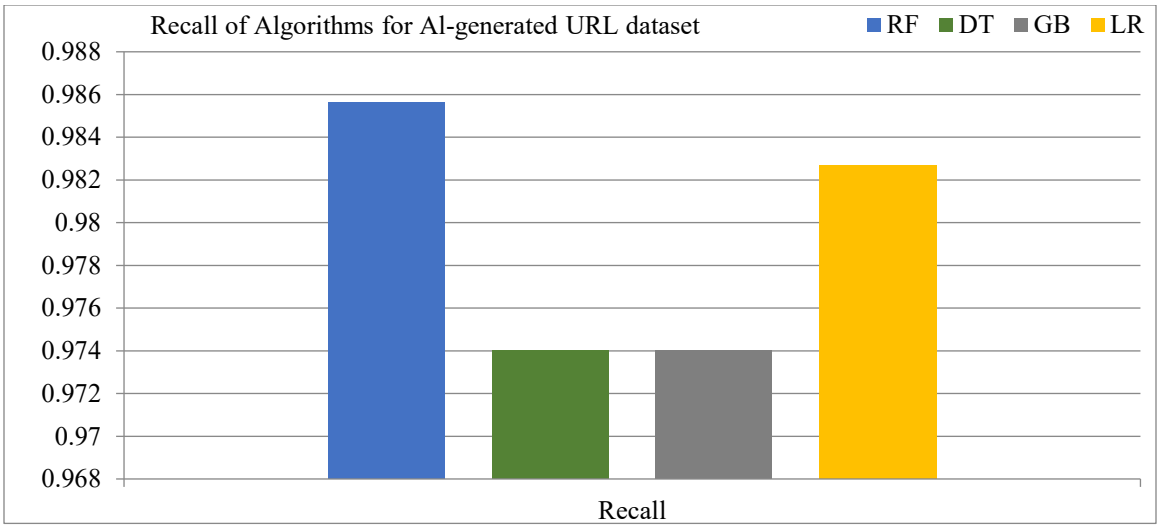


Fig. 17 Recall of the classification algorithms for AI-generated URL dataset

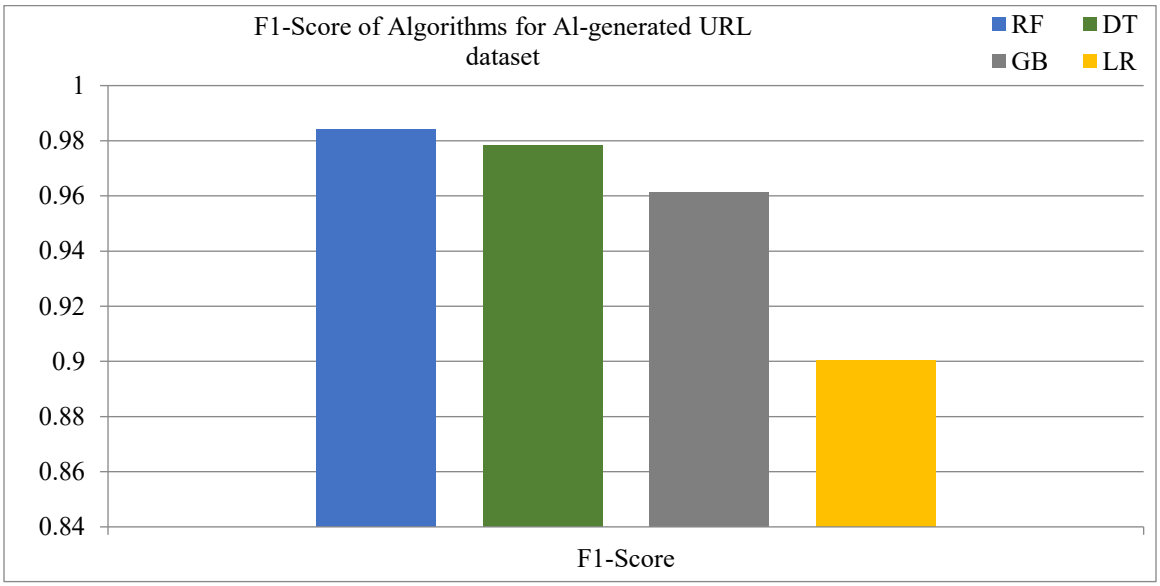


Fig. 18 F1-Score of the classification algorithms for AI-generated URL dataset

Table 3. Comparative results of the RL agent

Methodology	Precision	Recall	Accuracy	F1-score
[4] Machine learning ensemble model	98.00	97.95	98.00	97.96
[8] Ensemble Model	96.81	96.32	97.44	96.56
[9] Deep learning model	99.02	98.99	99.05	99
[11] Hybrid ML Model	95.15	96.38	95.23	95.77
[12] DDQN+RL	87.5	95.1	88.4	91.1
[13] RL-Based Framework	97.98	96.74	98.75	97.66
[20] Deep Q-network + _RL	96	94	95	-
[21] Deep learning and reinforcement learning.	95.71	97.15	96.40	96.42
Proposed System	**98.72**	**98.82**	**99.25**	**98.77**

4.3. Comparative Results with the Existing Methodologies

This section presents a comparative study of the proposed methodology with existing systems. Most of the existing methodologies have evaluated the performances based on precision, recall, F1-score, and AUC values. Table 3 gives a comparative study of the performances of existing methodologies with the proposed system. The comparative results of the performances of existing systems and the

proposed methodology show that the proposed system has performed better in terms of accuracy, precision, recall, and F1-score. The adaptive and continuous learning nature of the RL agent has contributed to the better performance of the proposed methodology compared to existing systems. The proposed system also ensures dynamic feature subset selection along with the classification model for better learning of the agent to ensure achieving good accuracy.

4.4. Analysis based on the Confusion Matrix for AI-Generated URLs

The confusion matrix heatmap shown in Figure 19 highlights the number of false positives detected by the system. It shows that 17 legitimate URLs were incorrectly classified as phishing URLs. The results show that on test data, the system has misclassified a few URLs. However, considering the adaptive nature of the RL agent, the agent will improve its performance over a period of time. However, high precision and recall indicate that the system prioritizes user safety by minimizing undetected phishing threats.

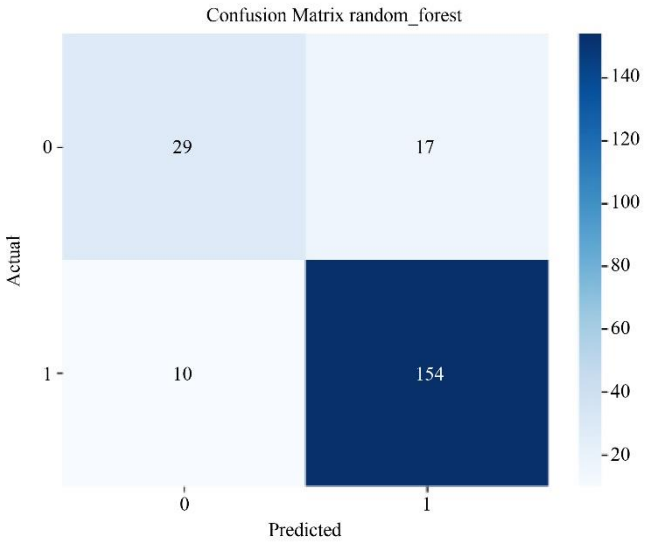


Fig. 19 Confusion matrix of results of AI-generated URLs

5. Discussions

The evolving nature of phishing strategies is driven by the increasing adoption of technology. The attackers have started using AI for launching attacks and generating phishing URLs that can be bypassed by the current detection systems. The proposed methodology is thus tested on the AI-generated URLs. The proposed methodology has succeeded in detecting up to 95% of these phishing URLs. Figure 9 illustrates that the proposed methodology of dynamic feature and classification model selection can detect the AI-generated URLs effectively. The system also ensures continuous learning by rewarding the agent for correct predictions. Furthermore, as shown in Figure 10, the proposed methodology has achieved the highest accuracy of 99.25 percent for the Random Forest classifier for feature subset size=14

Based on the experiments conducted in the study, it can be stated that the reinforcement learning-based feature selection approach increases the effectiveness of the model. By selecting the most relevant features, the model reduces computational complexity while improving detection performance. The high precision and recall scores suggest that the model is highly effective for real-time phishing detection applications. The dynamic nature of feature selection in the Q-learning model enables the agent to adapt to evolving phishing patterns, representing a significant improvement over the static feature selection technique. However, further improvements can be made by optimizing the reward function and exploring hybrid models that combine reinforcement learning with deep neural networks. Feature selection has thus been proven to be a dominant factor in enhancing the performance of the system. These results open avenues for future research in adaptive phishing detection frameworks, leveraging reinforcement learning to improve detection efficiency in dynamic environments.

Furthermore, experiments were performed on the agent using AI-generated URLs. The model demonstrated its capability to handle adversarially generated phishing URLs, which are typically more challenging to detect than conventional phishing URLs. This experiment highlighted the strength of the reinforcement learning-based feature selection in capturing subtle patterns in malicious URLs. The consistent performance across different types of phishing URLs, including AI-generated ones, confirms the model's generalization ability. This is crucial in real-time environments where attackers constantly modify their strategies. The results highlight the potential of RL-based recognition systems in combating evolving phishing threats, especially AI-generated phishing attacks, which are expected to become more prevalent.

The proposed reinforcement learning framework has been developed with considerations of ethical and responsible use of AI in cybersecurity. The dataset used is publicly available. The inclusion of AI-generated URLs is only for defensive research, enabling the model to recognize emerging phishing patterns in a secure, controlled environment without any risk of misuse. The adaptive learning behavior of RL agents can be both powerful and potentially risky. While the agents autonomously optimize feature subsets and classifier selection to enhance phishing detection accuracy, such adaptability could be misused to optimize phishing attacks if replicated irresponsibly. To mitigate this concern, the reinforcement learning environment and AI-generated data are used strictly for research and testing purposes**.** No generated URLs are publicly released, deployed, or used for real-world testing beyond the academic research context.

6. Conclusion and Future Scope

Multiple solutions to detect phishing attacks have been implemented. However, the evolving nature of the phishing

attacks makes this eradication difficult. The researchers have shown how the AI-generated URLs could easily bypass the current solutions. The proposed methodology has addressed both the challenges of the dynamic nature of phishing attacks and the AI-generated URLs. The proposed methodology has used a hybrid approach by using RL methodology and ML models for implementing the phishing detection system.

The adaptive nature of RL and the performance of the ML models together have demonstrated an accuracy of 99.25%. During the experiment on AI-generated URLs, it achieved an accuracy of 95% which can be improved by additional training episodes.The future work could explore the use of multiple agents for feature selection and model selection. The system could provide a probability score instead of classification into phishing or legitimate URLs. Techniques can be employed to implement autonomy in feature selection, rather than restricting it to the features included in the dataset. RL agents can be trained using Deep-Q networks instead of the Q-learning algorithm.

The proposed work uses accuracy as a reward function. Additional work can be done on optimization of the reward function. Such experiments may achieve better results in real-world applications. The work can also be extended by reducing the number of false positives that result in the unnecessary blocking of genuine websites. Manual feedback can be included for such cases.

References

[1] Koceilah Rekouche, "Early Phishing," *arXiv Preprint*, pp. 1-9, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[2] "*Phishing Activity Trends Report*," Summary - 1st Quarter 2025, Anti-Phishing Working Group, 2025. [Publisher Link]

[3] Darren E. Tromblay, *Federal Bureau of Investigation*, The Handbook of Homeland Security, 1st ed., CRC Press, 2023. [Google Scholar] [Publisher Link]

[4] Maria Sameen, Kyunghyun Han, and Seong Oun Hwang, "Phishhaven-An Efficient Real-Time AI Phishing URLs Detection System," *IEEE Access*, vol. 8, pp. 83425-83443, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Beauden John, "Adapting to Advanced Threats: Celery Trap's Approach to Combating AI-Generated Phishing Campaigns," pp. 1-9, 2025. [Google Scholar]

[6] Alejandro Correa Bahnsen et al., "DeepPhish: Simulating Malicious AI," *2018 APWG Symposium on Electronic Crime Research*, pp. 1-8, 2018. [Google Scholar]

[7] Nguyet Quang Do et al., "Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions," *IEEE Access*, vol. 10, pp. 36429-36463, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] R.J. Van Geest et al., "The Applicability of a Hybrid Framework for Automated Phishing Detection," *Computers and Security*, vol. 139, pp. 1-17, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Y. Bhanu Prasad, and Venkatesulu Dondeti, "PDSMV3-DCRNN: A Novel Ensemble Deep Learning Framework for Enhancing Phishing Detection and URL Extraction," *Computers and Security*, vol. 148, pp. 1-16, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Felipe Castaño et al., "PhiKitA: Phishing Kit Attacks Dataset for Phishing Websites Identification," *IEEE Access*, vol. 11, pp. 40779-40789, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Abdul Karim et al., "Phishing Detection System through Hybrid Machine Learning Based on URL," *IEEE Access*, vol. 11, pp. 36805-36822, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Antonio Maci et al., "Unbalanced Web Phishing Classification through Deep Reinforcement Learning," *Computers*, vol. 12, no. 6, pp. 1-30, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Orel Lavie, Asaf Shabtai, and Gilad Katz, "A Transferable and Automatic Tuning of Deep Reinforcement Learning for Cost Effective Phishing Detection," *arXiv Preprint*, pp. 1-43, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Hasan Kamal et al., *Reinforcement Learning Model for Detecting Phishing Websites*, Cybersecurity and Artificial Intelligence, Springer, Cham, pp. 309-326, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Grega Vrbančič, Iztok Fister, and Vili Podgorelec, "Datasets for Phishing Websites Detection" *Data in Brief*, vol. 33, pp. 1-7, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] Rodolfo Vieira Valentim et al., "URLGEN-Toward Automatic URL Generation Using GANs," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3734-3746, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Richard S. Sutton, and Andrew G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Adaptive Computation and Machine Learning Series, The MIT Press, 2018. [Google Scholar] [Publisher Link]

[18] Abdul Basit et al., "A Comprehensive Survey of AI-Enabled Phishing Attacks Detection Techniques," *Telecommunication Systems*, vol. 76, no. 1, pp. 139-154, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Ankit Kumar Jain, and B.B. Gupta, "Phishing Detection: Analysis of Visual Similarity Based Approaches," *Security and Communication Networks*, vol. 2017, pp. 1-20, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[20] Haidar Jabbar, and Samir Al-Janabi, "AI-Driven Phishing Detection: Enhancing Cybersecurity with Reinforcement Learning," *Journal of Cybersecurity and Privacy*, vol. 5, no. 2, pp. 1-21, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[21] Subhash Ariyadasa, Shantha Fernando, and Subha Fernando, "A Reinforcement Learning-Based Intelligent Anti-Phishing Solution to Detect Spoofed Website Attacks," *International Journal of Information Security*, vol. 23, no. 2, pp. 1055-1076, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Richard S. Sutton, and Andrew G. Barto, "Reinforcement Learning," *Journal of Cognitive Neuroscience*, vol. 11, no. 1, pp. 126-134, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[23] H.S. Harisudhan, NLP Transformers-The Backbone of Today's Language Models, Medium, 2025. [Online]. Available: https://medium.com/@speaktoharisudhan/nlp-transformers-the-backbone-of-todays-language-models-d752a2bf0752

[24] J.O. Schneppat, Transformer Neural Networks, Schneppat AI, 2017. [Online]. Available: https://schneppat.com/transformer-neural-networks.html