

Using Ensemble Models for Malicious Web Links Detection

Claudia-Ioana Coste^a

Faculty of Mathematics and Computer Science, Babeş-Bolyai University,
Mihail Kogalniceanu Street, no. 1, Cluj-Napoca, Romania

Keywords: Malicious Web Links Detection, Machine Learning Algorithms, Ensemble Models, Particle Swarm Optimization, Nature-Inspired Algorithms, Web-Malware.

Abstract: Web technology advances faster than humans can adapt to it and develop the proper online skills. Most users are not experienced enough to have a good online knowledge on how to protect their data. Thus, many people can become vulnerable to threats. The most common online attacks are through malicious web links, which can deceive users into clicking them and running malicious code. The present approach proposed to advance the field of malicious web links detection through ensemble models by developing a nature-inspired ensemble. Our methodology is tested against two datasets, and we conduct an additional calibration step for all the models. For the first database, we managed to improve the detection accuracy from other solutions, by achieving 97.05%. In the case of the second dataset, our empirical strategy is not accurate enough, reaching just 91.12% accuracy. The proposed ensemble is heterogeneous, having a weight voting mechanism, where weights are generated with the Particle Swarm Optimization algorithm. To build the ensemble we compared 12 individual machine learning models, including Logistic Regression, Support Vector Machine, Adaptive Boosting, Random Forest, Decision Tree, K-Nearest Neighbor, Perceptron, Nearest Centroid, Passive Aggressive Classifier, Stochastic Gradient Descent, KMeans, and different variants for Naive Bayes.


1 INTRODUCTION

With the latest development of web applications, there were significant improvements in web security, but as well in producing new web malware. These threats have become more and more sophisticated affecting many people with poor online abilities. An important category of attacks are malicious links which can be spread through social media posts, private messages, SMSs, and emails. The main security issues associated with suspicious links are performing redirections or downloading programs without the user's consent (i.e. drive-by-downloads), executing malicious code that can use browser's resources and leaking personal information. Finally, malicious links can contribute to the disinformation phenomenon by pointing to fake news and low-quality online content such as clickbait articles.

In 2022, it was stated by (IT, 2023), that some of the most common top level domains are included in malicious links. For instance, 54% of them belong to .com domain and 8.4% have the .net domain. From the domains included under .com, some of the

most common ones are *Adobe, Google, Myportfolio, Backblaze2, Weebly*. Moreover, the shortened URLs are a constant threat since they may hide malicious links. In 2021, the most common malware link found in emails was a Trojan with the aim of stealing browser's information (e.g., stored credentials, login information etc.) (IT, 2023). Developing a solution for malicious links involves a complex pipeline where the classification should be robust, sensitive, and fast enough to not influence users' online experience (Saxe et al., 2018).

To our consideration, malicious web links detection can be seen as a binary classification problem, where the input is a text representing an URL and the output should be a class malicious or benign. The present solution proposes a URL preprocessing, where features are extracted from the URL to form the feature vector using the term frequency-inverse document frequency (TF-IDF) method. Thus, we can split our problem definition into two stages: feature extraction and classification. The feature extraction phase can be described as $f : \text{WebLinks} \rightarrow \mathbb{R}^d$, $f(\text{web_link}) = (x_1, x_2, x_3, \dots, x_d)$, where d is the number of features and x_d , for each $d \in \mathbb{N}$ is a feature. Next, the classification is taking as input the feature vector computed

^a  <https://orcid.org/0000-0001-8076-9423>

with TF-IDF formula and it returns a class benign or malicious, encoded as 0 or 1. $f : \mathbb{R}^d \rightarrow \mathbb{R}$, $f(X) = \text{class}$.

Current solution proposes a nature-inspired ensemble that could advance the research in malicious web links detection. To our considerations, this type of ensemble was not yet applied to malicious web links detection. The ensemble is heterogeneous, and it is formed with some of the best performing machine learning (ML) algorithms and as a meta-classifier to adjust the weights, a Particle Swarm Optimization (PSO) algorithm is used. For the ML models depicted to form the ensemble, we tested a total of 12 methods, such as: Naive Bayes (NB) variants (e.g., Bernoulli NB, Multinomial NB, Complement NB), Perceptron, Passive Aggressive Classifier (PAC), Stochastic Gradient Descent Classifier (SGD), Nearest Centroid (NC), Logistic Regression (LR), Adaptive Boosting (ADA), Decision Tree (DT), Random Forest (RF), KMeans, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM) with different kernel values (e.g., polynomial, linear, radial basis function and sigmoid). The approach is tested against two datasets: D1 (GTKlondike, 2019) and D2 (Siddhartha, 2021). The best result for D1, an accuracy of 97.05%, is achieved with an ensemble formed out of seven ML algorithms, improving the solution provided in (Pakhare et al., 2021). On the second dataset, D2, an ensemble with 3 classifiers achieved just 91.12% accuracy, being not as efficient as the approach in (Alsaedi et al., 2022). Still, we proposed a novel detection model using TF-IDF for dataset D2.

The present approach is structured in four sections. In the next section, an overview of the current state of the art is detailed in the domain of malicious web links detection using ensemble models. The 3 section tackles the empirical methodology used for the experiments. The 4 part discusses the results of our experiments together with relevant comparisons. The last section draws conclusions and presents future directions for research.

2 PREVIOUS WORK

There has been work done considering ensemble models when detecting malicious web links. Some works take into consideration features extracted from the link and others from the web content of the web page. In both cases, ensembles are a better alternative to single models considering accuracy. Additionally, nature inspired algorithms are used for malicious web links detection, especially Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

2.1 Ensemble Models

Work done by (Pakhare et al., 2021) considers running individual models and ensemble models. The ensemble is formed with three ML models, and the final output is decided by an equally defined voting mechanism. The best ML individual model is LR, achieving 94.31% accuracy. The best ensemble was the one containing KNN, DT and sigmoid SVM with an accuracy of 94.93%. The preprocessing is done with the TF IDF method. Similarly, (Subasi et al., 2021) compares multiple types of homogeneous ensembles (e.g., AdaBoost, Random Subspace, Multi Boost, Bagging) with multiple ML algorithms. All models are tested using a private dataset containing 5000 URLs. Random Subspace used in conjunction with KNN outperforms the rest with 89.24% accuracy rate.

Another stacked ensemble type was proposed by (Alsaedi et al., 2022) with three RF models and a MLP as meta-classifier. Each RF model is specifically tuned for a particular feature category. All extracted features fall into three categories: lexical, WHOIS features and Google Cyber Threat Intelligence (CTI) characteristics. The RF model was chosen based on multiple experiments where its performance was compared with other ML models, including some deep learning methods. The proposed solution reaches 96.8% accuracy. The dataset used is found in (Siddhartha, 2021). For experiments, 20,000 links were randomly sampled for each class. This same dataset is used in (Shetty et al., 2023) and in (Zhang and Yan, 2023). In (Shetty et al., 2023) there is a comparison between tree-models in a multi-classification problem with four classes: benign, phishing, defacement, and malware. The weighted average across all classes is 97% for precision, recall and F1 score. Considering malicious web links detection as a multi-classification and a binary problem, (Zhang and Yan, 2023) is delivering a complex neural network (NN) model. The methodology involves extracting features from the URL at a word-piece level. For classification, it used a multi-head attention mechanism connected to a Convolutional Neural Network (CNN) with N-gram. Word piece representation of URL features is compared with char and char & word representation. (Zhang and Yan, 2023) is using two datasets to prove their approach and the Multi-filter CNN model was compared with other deep learning methods proposed by literature.

2.2 Nature-Inspired Solutions

Exploring ensemble methods and GA, (Sajedi, 2019), proposed a solution with REP trees and a meta-

classifier developed with GA. This approach is compared with other homogeneous ensemble types (e.g., AdaBoost, Bagging, Decorate, Rotation Forest) and multiple classifiers. The features are manually engineered and extracted from the HTML and JavaScript files. Their proposed approach has an accuracy of 95.38% on a small dataset containing a total of 4,400 web pages.

The PSO algorithm is used to optimize classification process for phishing websites detection. For instance, in (Gupta and Singhal, 2017), PSO is used to better train an ANN model. Their proposed model outperforms the Back Propagation Neural Network (BPNN). The dataset is retrieved from (PhishTank, 2023), having 31 attributes and 11,055 records. Similarly, to improve detection of phishing links, (Ali and Malebary, 2020) is using PSO for feature weighting and the classification is done by multiple ML models (e.g., BPNN, SVM, KNN, DT, RF, NB). For feature selection, PSO-optimization is compared with GA, information gain, wrapper, and chi-square test. The best performance is reached by RF with PSO feature selection, with 96.83% accuracy. The dataset has 11,055 phishing and non-phishing websites. Another solution proposed by (Lee et al., 2020) is using PSO for malicious web links detection. More specifically, PSO is run for feature selection to improve the accuracy of NB and SVM classifiers. This approach reaches 99% accuracy for both estimators. Moreover, the PSO algorithm is used as well in an ensemble but to build an intrusion detection system. (Aburoman and Reaz, 2016) is proposing a KNN-SVM-PSO ensemble. PSO is used in comparison with meta-optimized PSO and with weighted majority voting.

3 METHODOLOGY

The methodology followed when driving experiments for our approach can be split into the next steps:

1. Preprocessing;
2. Balancing the datasets;
3. Run individual methods;
4. Run PSO ensemble models.

3.1 Preprocessing

The first step of our methodology involves preprocessing the database with TF-IDF vectorizer from Python Sklearn library (Pedregosa et al., 2011). For the experiments we used two databases, one from (GTKlondike, 2019) (D1) and the other one from

(Siddhartha, 2021) (D2). Both datasets were previously used in other malicious web links detection solutions. The dataset (GTKlondike, 2019) was used in (Pakhare et al., 2021) and the dataset (Siddhartha, 2021) was used for binary classification in (Alsaedi et al., 2022) and for multi-label classification in (Zhang and Yan, 2023) and (Shetty et al., 2023). There are not many details on how the data within the first dataset was collected, but there are a total of 420,464 samples (344,821 benign and 75,643 malicious). The second dataset is freely available on Kaggle and it was formed by aggregating multiple datasets such as ISCX-URL-2016 (Mamun et al., 2016), (malwaredomainlist, 2010), (Marchal et al., 2014), (PhishTank, 2023), and (Joerg, 2017). D2 provides 651,191 URLs (428,102 benign, 32,520 malwares, 94,111 phishing and 96,455 defacement). We preprocessed it for binary classification, the malware, phishing, and defacement classes were merged. Eventually, in D2 there were 428,102 legitimate samples and 223,086 malicious ones.

The TF-IDF preprocessing was done following the methodology from (Pakhare et al., 2021), using a TFIDF vectorizer.

3.2 Balancing the Datasets

Because in this domain of malicious web links detection is quite common to deal with dataset imbalance problems, we chose to equal the number of malicious and benign links. Thus, as in the methodology followed by (Pakhare et al., 2021), from the first dataset D1 (GTKlondike, 2019) we used in experiments 75,643 legitimate links and 75,643 malicious ones. From the second dataset D2 (Siddhartha, 2021) we chose to randomly depict just 20,000 records from each class, as in (Alsaedi et al., 2022).

3.3 Run Individual Methods

There were experiments driven with multiple ML algorithms, such as NB variants (e.g., Bernoulli NB, Multinomial NB, Complement NB), Perceptron, PAC, SGD classifier, NC, LR, ADA, DT, RF, KMeans, KNN, and SVMs with different kernel values (e.g., polynomial, linear, radial basis function and sigmoid). There are a total of 12 different ML models. We explicitly chose these algorithms such that they would be suitable to work with sparse data since the result of the TFIDF tokenization is a sparse matrix. All models used in the experiments were implemented in the Python Sklearn framework (Pedregosa et al., 2011). Moreover, the algorithms were run taking into consideration a parameters' calibration

step, which was implemented with GridSearchCV and RandomizedSearchCV (Pedregosa et al., 2011). The values for the parameters were chosen according to the documentation and our binary classification problem. For training 75% of the dataset was used, while the rest of 25% was used for testing. GridSearchCV and RandomizedSearchCV are both using 5-fold cross-validation to find the best classifier.

3.4 Run PSO Ensemble Models

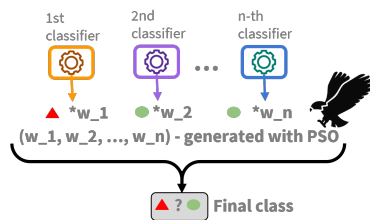


Figure 1: The PSO-ensemble model.

The proposed ensembles are heterogeneous, and we opted for a Voting Classifier implemented in (Pedregosa et al., 2011) with two voting strategies (i.e., 'hard' or 'soft') and a set of weights to adjust the contribution of each single classifier compounding the ensemble. A graphical representation of the ensemble is presented in Figure 1. The final decision is a weighted average of each output given by the classifiers. The ML algorithms that are going to form the ensemble were depicted from TOP 3, TOP 5, or TOP 7 individual algorithms. This algorithm selection was proposed because, to our considerations, an uneven number of models forming the ensemble would be more efficient. Moreover, in (Pakhare et al., 2021) and (Alsaedi et al., 2022), the solutions proposed by them are using ensembles constructed with three classifiers.

The weights are generated based on the PSO algorithm. The PSO algorithm was first proposed in (Kennedy and Eberhart, 1995) and it was inspired by how birds cooperate in a flock when searching for food. Each possible solution is represented by a bird, which is described by a position and a velocity. Each candidate stores its best position and the best global position obtained by the flock. At each iteration, birds are updating their positions and velocities according to their previous records using the inertia coefficient (w) and based on the flock's previous positions by considering a cognitive factor ($c1$) and a social parameter ($c2$).

In the ensemble's case, we used 75% of data for training and 25% for testing. The training set is further split into 75% exclusively for training and 25%

Table 1: Details on how the PSO ensemble was calibrated.

Parameters	Values
Type (Ens.)	{TOP 3, TOP 5, TOP 7}
Strategy - weight voting (Ens.)	{weight_voting_hard, weight_voting_soft}
no.iterations (PSO)	10, 100, 300, 500
no.particles (PSO)	10, 30, 50, 80, 100
$c1, c2, w$ (PSO)	randomly uniformly distributed [0.5, 0.99]
$c1, c2, w$ strategy (PSO)	{exp_decay, nonlin_mod, lin_variation}

for validation. Based on the validation set, the PSO algorithm tries to optimize the ensemble accuracy by adjusting the weights. Then after the weights are generated based on the validation, the testing is done. The calibration process of the PSO ensemble is based on multiple experiments. The parameters used to configure our nature-inspired approach are: $c1$ (cognitive parameter), $c2$ (social fact) and w (inertia coefficient), number of iterations, number of particles and ensemble's voting strategy (e.g., soft or hard). The calibration was done in two steps to better manage the computer's resources and time. The first stage involves the calibration of the adjusting strategies for the PSO options parameters (i.e., $c1, c2, w$), their initial values and the ensemble's strategy. During this stage the number of PSO iterations are established at 200 and the number of particles at 20 for the first dataset and to 100 and 10 respectively, for the second one. The initial values for the PSO options are randomly generated from the 0.5 and 0.99 interval, as in (Aburomman and Reaz, 2016). The results of the first phase are sorted according to the testing accuracy and an average value is computed for the $c1, c2$, and w parameters. These mean values are going to be the initial values for the $w, c1, c2$ parameters for the second phase of calibration experiments. During the second stage, the number of iterations is calibrated together with the number of particles. Table 1 has more details on the parameter calibration done for the PSO-ensemble solution.

PSO was chosen to adjust the weights of the ensemble model based on the work developed by (Sajedi, 2019), where a Genetic Algorithm is a meta-classifier for the ensemble. We chose PSO as a meta-classifier because of its previous usages for feature optimization as in (Gupta and Singhal, 2017) and (Lee et al., 2020). Because of its good results on feature selection, we propose to experiment with it for the weighting of the classifiers within an ensemble.

4 EXPERIMENTS

The next section presents details about the experiments run on both single models and on ensemble models. All experiments are developed with Python version 3.9, using models already implemented in Python scikit-learn library (Pedregosa et al., 2011) version 1.0.2. The PSO implementation was from the pyswarms library version 1.3.0 (Lester, 2017). The empirical analysis was run on a 64-core machine with 64 GBs of RAM.

4.1 Individual Models Results

The current subsection tackles the performance of the single algorithms on dataset D1 (Pakhare et al., 2021) and on dataset D2 (Siddhartha, 2021). The results for the individual algorithms run on D1 are graphically presented in Figure 2a. It can be observed that the best seven models are in order: SGD, PAC, Linear SVC or SVC, Bernoulli, Complement and Multinomial NB, LR. Thus, the ensembles for D1 are formed with:

- TOP 3 ensemble: SGD + PAC + linear SVC;
- TOP 5 ensemble: SGD + PAC + linear SVC + Bernoulli NB + Complement NB;
- TOP 7 ensemble: SGD + PAC + linear SVC + Bernoulli NB + Complement NB + Multinomial NB + LR.

Comparing the results achieved by Linear SVC and SVC, we chose to continue with Linear SVC model because there are small differences between their metrics and because it is more efficient considering time. Moreover, the best estimator for the SVC model is using a linear kernel. In addition, in Figure 2a, it could be observed that KMeans is underperforming, reaching just 48.09% accuracy. This may be due to the reason that having sparse TF-IDF data and splitting it into just two clusters is not accurate enough. Maybe working with other features, especially less features, and in a multiclass problem, KMeans would perform better for malicious web links detection.

The results for the single models run on D2 dataset (Siddhartha, 2021) are presented in Figure 2b. There is a resemblance between the results for D1 and D2, the top seven models are almost the same. In both cases, SGD has the best performance. While in the D1 case, SGD has a test accuracy of 96.73%, in D2 case it achieves just 91.07% accuracy. Moreover, it can be observed that the RF classifier manages to reach fifth place on the D2 dataset, while for D1, RF misses the top seven algorithms. In addition, there is not much of

a difference between the ranking obtained by NB variants and LR model by comparing the results for the two datasets. A similarity between the results for both datasets is the linear SVC and SVC reach almost the same accuracy, placing them on the third and fourth rank. For the same reasons as in the case of dataset D1, we decided to further use Linear SVC instead of SVC. For D2, KMeans is the worst performing algorithm as it is in the D1's case. The similarity between the input data for D1 and D2 could be responsible for such similar results. The top algorithms compounding the ensembles are:

- Top 3 ensemble: SGD + PAC + Linear SVC;
- Top 5 ensemble: SGD + PAC + Linear SVC + RF + Bernoulli NB;
- Top 7 ensemble: SGD + PAC + Linear SVC + RF + Bernoulli NB + Multinomial NB + LR.

4.2 PSO Ensemble Results

Current subsection details the performance of the ensemble models on both datasets. The experiments were run on the data quantifications presented in Table 2.

Table 2: The datasets for the experiments run on the ensemble models.

Dataset	D1	D2
Training	85,098	22,500
Validation	28,366	7,500
Testing	37,822	10,000
Total	151,286	40,000

Table 3: The parameters for PSO-ensemble configuration.

Parameters	D1	D2
Type (Ens.)	TOP 7	TOP 3
Strategy - weight voting (Ens.)	soft	soft
no_iterations (PSO)	100	100
no_particles (PSO)	30	10
w (PSO)	0.815	0.747
c1 (PSO)	0.739	0.7904
c2 (PSO)	0.829	0.671
w_strategy (PSO)	exp_decay	exp_decay
c1_strategy (PSO)	lin_variation	exp_decay
c2_strategy (PSO)	nonlin_mod	nonlin_mod

For both datasets, the calibration phase of the ensemble's parameters was done over two phases. In the first stage, the strategy parameters were calibrated (i.e., w_strategy, c1_strategy, c2_strategy, ensemble's

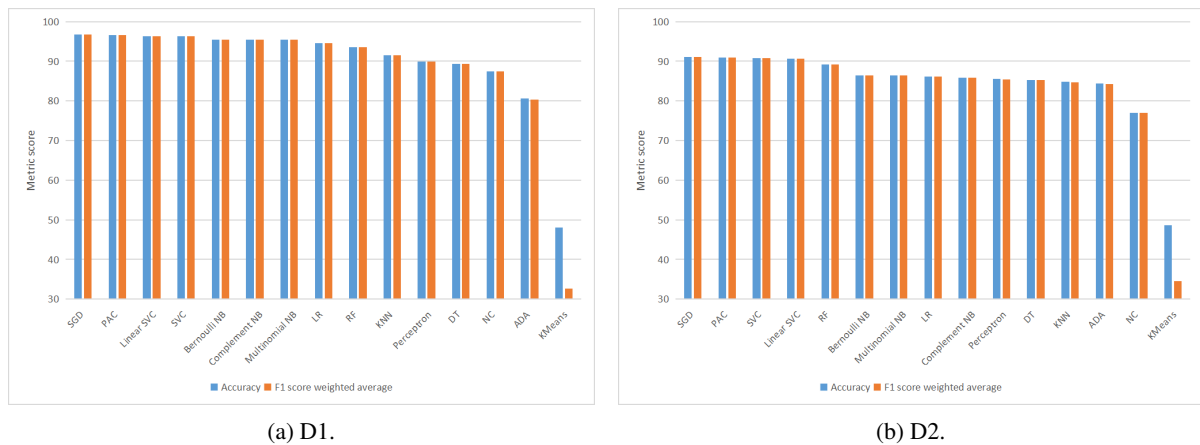


Figure 2: The accuracy and weighted average of F1 score for D1 and D2 for individual algorithms (metrics computed on the testing set as an average of 10 different data splits).

strategy, its ML algorithms) and the initialization values for w , $c1$, and $c2$. Little differences have been found in the experiments considering the first calibration stage. Still, we depicted the values for the best performance and continued with the second phase of calibration. The chosen values for the parameters are detailed in Table 3 for both datasets. It can be observed that the calibration is rather similar, with some exceptions such as the type of the ensemble, the number of particles, the initialization values for $c2$, the social parameter, and its updating policy. The *lin_variation* strategy is tuning the parameter linearly decreasing or increasing it, while *exp_decay* is updating the parameter exponentially. The *nonlin_mod* strategy means that the parameter is updated considering a nonlinear modulation index. The initialization values for w , $c1$, and $c2$ were computed as an average of the best ten ensemble configurations. From the experiments run, there were little differences, at most 1%, between the performance obtained for different values for the parameters.

All in all, the final experiment involves running the obtained configuration on five different data splits to get the proper results, which are going to be compared with other similar approaches. The obtained accuracy for D1 is detailed in Table 4 together with the generated weights. It can be observed that the best performing algorithm from the individual models has the highest weight in the ensemble. This is true, as well, for the least performing ML method, LR. The rest of the intelligent mechanisms are weighted according to the validation set, trying to adjust the ensemble to be robust and flexible.

Considering the first stage of experiments for the D2 ensemble, there were no significant differences between the parameters. Still, the methodology tries to maximize the performance of our algorithm and

continues with the values that obtain the best accuracy scores. The tuned values for the PSO-ensemble are presented in 3. The last experiment on D2 runs the fine-tuned configuration on five data splits. The D2 ensemble reaches an accuracy of 91.112% as it is displayed in Table 4. Moreover, in the same table, there are detailed the obtained weights from running PSO. As was observed in the case for D1 dataset, the best performing individual algorithm has the largest weight. In this case, the SGD algorithm has the largest weight, and it is followed by the Linear SVC, which is the third performing algorithm. PAC has the lowest weight, even though it is in second place considering its efficiency as a solo algorithm. This may be because the PSO-ensemble is trained to be robust and more flexible, such that on the testing set (unseen data) it would still perform well enough. Taking into consideration the time used for training and testing an ensemble as seen in Table 4, running an ensemble is considerably more expensive than using a single model. SGD has a training time of approximately 2,416 ms on D1 and 612 ms on D2. The same thing is observable for the testing time which is smaller compared to the testing time of the ensembles. SGD has 2 ms for testing on D1 and 0.58 ms for D2. In this case, using ensembles is a trade-off between accuracy and running time because accuracy is improved compared with the single models.

4.3 Comparisons and Discussion

For this approach, we chose to follow the data preprocessing methodology from (Pakhare et al., 2021). In this way, the comparison provided is very accurate and relevant. Moreover, the dataset was freely available, and the methodology was proposing TF-IDF lexical features. Lexical features are not time

Table 4: The generated weights and the final metrics (i.e. accuracy, F1 score - weighted average, training and testing time) for the PSO ensembles.

	Weights obtained for the PSO ensemble (weight, classifier)	Acc.	F1 score	Training time (ms)	Testing time (ms)
D1	{(1.61, SGD), (1.05, ComplementNB), (1.01, BernoulliNB), (0.73, PAC), (0.57, MultinomialNB), (0.29, LinearSVC), (0.26, LR)}	97.0546	97.0545	3,291,308	58
D2	{(0.58, SGD), (0.38, LinearSVC), (0.26, PAC)}	91.112	91.107	107,317	4

dependent; they are not directly dependable if the online content of a web link changes over time. Thus, their methodology enables the experiments to be easily reproducible. In (Pakhare et al., 2021), the authors propose LR, as the best individual algorithm with an accuracy of 94.31% and the majority voting ensemble KNN-DT-SVM with 94.93% accuracy. The graphical representation from Figure 3 presents the accuracies previously obtained in literature and our obtained performances. It can be observed that with SGD (96.66% accuracy) as a single algorithm, we managed to surpass the best accuracy reached for the KNN-DT-SVM ensemble in (Pakhare et al., 2021). Moreover, our Top 7 ensemble (97.05% accuracy) tested against dataset D1 manages to further improve the accuracy. It is clear and logical for an ensemble model to improve accuracy of single models. As well, it can be observed that, the ensemble presented in (Pakhare et al., 2021) improved the classification with 0.62% compared with the single model, while in our case, we improved performance with just 0.39%.

The model proposed in (Alsaedi et al., 2022) reaches 96.8% accuracy. The algorithm is an ensemble as well, formed out of three RFs, and a MLP, that works as a meta-classifier. We proposed to keep the TF-IDF methodology developed on D1 and apply it on D2, such that we were able to intra-compare our methods. Moreover, their used features such as Google CTI-based characteristics and WHOIS information are time-dependent features, which may change over time. The process of feature retrieval is time-consuming, and it may not even be accurate. Even though the comparison between our proposed models and the ensemble proposed by (Alsaedi et al., 2022) may not be error-less or precise, there is still a contribution to their research by applying a new methodology and a new ensemble model. Moreover, there are differences because (Alsaedi et al., 2022) split the dataset into training 70% and 30% for testing and they added a feature selection step.

Even though the current results on D2 are satisfactory, they do not manage to improve the results obtained in (Alsaedi et al., 2022). With SGD as a single algorithm, an accuracy of 90.95% is obtained as it can be seen in 3. With the TOP 3 ensemble, an improve-

ment is made with respect to the individual model, the ensemble achieving 91.122%. Even though our TF-IDF and PSO-ensemble approach is inaccurate in this case, we must conclude that maybe the characteristics extracted by (Alsaedi et al., 2022) (lexical, WHOIS information and Google CTI-based features) are more relevant than TF-IDF features. This could support the fact that maybe time-dependent features could add valuable information to the input data and reduce the complexity of the ML model. To conclude that our model is worse than theirs, there needs to be more experiments using the same feature categories as input. Still, we managed to propose a novel approach using a PSO-weights ensemble and TF-IDF features on the D2 dataset. An improvement we could add to our proposed methodology could be adding a feature selection step. In addition, the TOP 3 ensemble used for D2 may overfit and to prevent that we could rerun the ensemble experiments on the TOP 5 and TOP 7 ensemble as well.

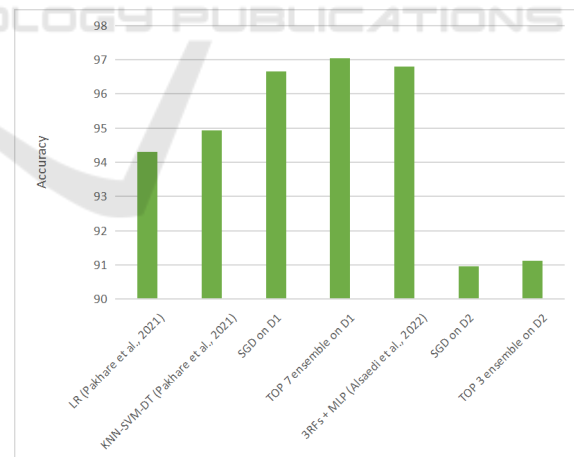


Figure 3: Comparisons between our approaches and models presented in (Pakhare et al., 2021) and (Alsaedi et al., 2022).

5 CONCLUSIONS

Because of the continuous improvements made in the web security field, attackers come with new ways of deceiving the detection mechanisms. Consumers are

affected as well; they are easily lured into clicking malicious links. The current approach proposed a novel PSO ensemble to improve and advance research done in the maliciousness detection problem of web links. Our ensemble is heterogeneous, combining multiple ML algorithms that proved to be efficient individually. The combining mechanism uses weights, which are generated with the PSO algorithm on a validation set. The experiments follow a calibration stage, and they are tested on two different datasets. The results achieved on the first dataset (97.05% accuracy) improve the previous solution. In contrast, for the second dataset, our approach is not so accurate compared to the solution found in literature. There are still things to improve, but, to our considerations, we manage to propose an innovative empirical approach on malicious web links detection.

Considering future work, we propose to develop a real-time reporting framework that aims to collect data associated with a link, including time-dependent features that can improve detection algorithms and reduce their complexity. Time-dependent features include network information if the web link was included in blacklists or whitelists. The framework is supposed to get a snapshot with the current link data. Moreover, we propose to test this PSO-ensemble against larger datasets to further prove its efficiency and robustness.

REFERENCES

- Abuomman, A. A. and Reaz, M. B. I. (2016). A novel svm-knn-psy ensemble method for intrusion detection system. *Applied Soft Computing*, 38:360–372.
- Ali, W. and Malebary, S. (2020). Particle swarm optimization-based feature weighting for improving intelligent phishing website detection. *IEEE Access*, 8:116766–116780.
- Alsaedi, M., Ghaleb, F. A., Saeed, F., Ahmad, J., and Alasl, M. (2022). Cyber threat intelligence-based malicious url detection model using ensemble learning. *Sensors*, 22(9):3373.
- GTKlondike (2019). Machine-learning-for-security-analysts. Dataset website.
- Gupta, S. and Singhal, A. (2017). Phishing url detection by using artificial neural network with psy. In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, pages 1–6. IEEE.
- IT, A. (2023).
- Joerg, S. (2017). Using-machine-learning-to-detect-malicious-urls. faizan dataset website.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.
- Lee, O. V., Heryanto, A., Ab Razak, M. F., Raffei, A. F. M., Phon, D. N. E., Kasim, S., and Sutikno, T. (2020). A malicious urls detection system using optimization and machine learning classifiers. *Indonesian Journal of Electrical Engineering and Computer Science*, 17(3):1210–1214.
- Lester, J. V. (2017). Welcome to pyswarms's documentation!
- malwaredomainlist (2010). Malware domain list. malware-domainlist.
- Mamun, M. S. I., Rathore, M. A., Lashkari, A. H., Stakhanova, N., and Ghorbani, A. A. (2016). Detecting malicious urls using lexical analysis. In *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10*, pages 467–482. Springer.
- Marchal, S., François, J., State, R., and Engel, T. (2014). Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471.
- Pakhare, P. S., Krishnan, S., and Charniya, N. N. (2021). Malicious url detection using machine learning and ensemble modeling. In *Computer Networks, Big Data and IoT*, pages 839–850. Springer, Singapore.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- PhishTank (2023). PhishTank - Out of the Net, into the Tank - Developer Information. PhishTank website.
- Sajedi, H. (2019). An ensemble algorithm for discovery of malicious web pages. *International Journal of Information and Computer Security*, 11(3):203–213.
- Saxe, J., Harang, R., Wild, C., and Sanders, H. (2018). A deep learning approach to fast, format-agnostic detection of malicious web content. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 8–14, San Francisco, CA, USA. IEEE, IEEE.
- Shetty, U., Patil, A., and Mohana, M. (2023). Malicious url detection and classification analysis using machine learning models. In *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, pages 470–476. IEEE.
- Siddhartha, M. (2021). Malicious urls dataset. Kaggle - Malicious URLs dataset.
- Subasi, A., Balfagih, M., Balfagih, Z., and Alfawwaz, K. (2021). A comparative evaluation of ensemble classifiers for malicious webpage detection. *Procedia Computer Science*, 194:272–279.
- Zhang, L. and Yan, Q. (2023). Detect malicious websites by building a neural network to capture global and local features of websites. *Research Square*.