INFORMATICA, 2020, Vol. 31, No. 1, 143–160
© 2020 *Vilnius University*
DOI: https://doi.org/10.15388/20-INFOR404

Comparison of Classification Algorithms for Detection of Phishing Websites

Paulius VAITKEVICIUS* Virginijus MARCINKEVICIUS
Vilnius University, Institute of Data Science and Digital Technologies,
Akademijos str. 4, LT-08412 Vilnius, Lithuania
e-mail: paulius.vaitkevicius@mif.vu.lt

Received: September 2019; accepted: January 2020

Abstract. Phishing activities remain a persistent security threat, with global losses exceeding 2.7 billion USD in 2018, according to the FBI's Internet Crime Complaint Center. In literature, different generations of phishing websites detection methods have been observed. The oldest methods include manual blacklisting of known phishing websites' URLs in the centralized database, but they have not been able to detect newly launched phishing websites. More recent studies have attempted to solve phishing websites detection as a supervised machine learning problem on phishing datasets, designed on features extracted from phishing websites' URLs. These studies have shown some classification algorithms performing better than others on differently designed datasets but have not distinguished the best classification algorithm for the phishing websites detection problem in general. The purpose of this research is to compare classic supervised machine learning algorithms on all publicly available phishing datasets with predefined features and to distinguish the best performing algorithm for solving the problem of phishing websites detection, regardless of a specific dataset design. Eight widely used classification algorithms were configured in Python using the Scikit Learn library and tested for classification accuracy on all publicly available phishing datasets. Later, classification algorithms were ranked by accuracy on different datasets using three different ranking techniques while testing the results for a statistically significant difference using Welch's T-Test. The comparison results are presented in this paper, showing ensembles and neural networks outperforming other classical algorithms.

Key words: phishing detection, classification algorithms, phishing datasets.

1. Introduction

Phishing is a form of cybercrime employing both social engineering and technical trickery to steal sensitive information, such as digital identity data, credit card data, login credentials, and other personal data, etc. from unsuspecting users by masking as a trustworthy entity. For example, the victim receives an e-mail from an adversary with a threatening message such as a possible bank or social media account termination or fake alert on illegal transaction (Lin Tan *et al.*, 2016), directing him to a fraudulent website that mimics a legitimate one. The adversary can use any information that the victim enters in the phishing website to steal identity or money (Whittaker *et al.*, 2010).

*Corresponding author.

Although there are many existing anti-phishing solutions, phishers continue to lure more and more victims. In 2018, the Anti-Phishing Working Group (APWG) reported as many as 785,920 unique phishing websites detected, with a 69.5% increase during the last five years of monitoring, from 463,750 unique phishing websites detected in 2014 (Anti-Phishing Working Group, 2018). Global losses from phishing activities exceeded 2.7 billion USD in 2018, according to the FBI's Internet Crime Complaint Center (Internet Crime Complaint Center, 2019).

Deceptive phishing attacks are still so successful nowadays because, in essence, they are "human-to-human" assaults performed by professional adversaries who (i) have financial motivation for their actions, (ii) exploit lack of awareness and computer illiteracy of ordinary Internet users (Adebowale *et al.*, 2019), and (iii) manage to learn from their previous experience and improve their future attacks to lure new victims more successfully. For this reason, ordinary Internet users cannot keep up with new trends of phishing attacks and learn to differentiate a legitimate website's URL from a malicious one, relying solely on their efforts.

In order to protect Internet users from criminal assaults, automated detection techniques for phishing websites recognition were started to develop. The oldest approach included manual blacklisting of known phishing websites' URLs in centralized databases, later used by Internet browsers to alert users about possible threats. The negative aspect of the blacklisting method is that these databases do not include newly launched phishing websites and therefore do not protect from "the zero hour" attacks, as most of the phishing URLs are inserted in centralized databases only 12 hours after the first phishing attack (Jain and Gupta, 2018a). More recent studies have attempted to solve phishing websites detection as a supervised machine learning problem. Many authors have conducted experiments using various classification methods and different phishing datasets with predefined features (Chiew *et al.*, 2019; Marchal *et al.*, 2016; Sahoo *et al.*, 2017).

The following open questions motivate our research:

1. State-of-the-art methods of phishing website detection report classification accuracy (the classification accuracy measure is described in Section 3.4.1) well above 99.50% and use different classification algorithms: ensembles (Gradient Boosting) (Marchal *et al.*, 2017), statistical models (Logistic Regression) (Whittaker *et al.*, 2010), probabilistic algorithms (Bayesian Network) (Xiang *et al.*, 2011), classification trees (C4.5) (Cui *et al.*, 2018). There is no common agreement about what classification algorithm is the most accurate in phishing website prediction on datasets with predefined features (Chiew *et al.*, 2019).
2. State-of-the-art methods demonstrate such high classification accuracies on highly unbalanced datasets with minority and majority classes. Classification accuracy measure has low construct validity on datasets where class balance is not proportional and show better results for the preferred class. Doubts remain, whether these results were demonstrated due to dataset dependent method design (Chiew *et al.*, 2019) or algorithms used in state-of-the-art research are preeminent compared to others.
3. To the best of our knowledge, no studies comparing classic classification algorithms' performance on all publicly available phishing datasets with predefined features were conducted to answer the questions mentioned above.

Therefore, the objective of this experimental research is to answer the research question: *which classical classification algorithm is best for solving the phishing websites detection problem, on all publicly available datasets with predefined features?*

In this paper we compare eight classic supervised machine learning algorithms of different types (for more details see Section 3.2) on three publicly available phishing datasets with predefined features being used by the scientific community in experiments with classification algorithms (for more details on datasets see Section 3.3).

We have designed an experiment where we used such algorithms:

1. AdaBoost (Wang, 2012),
2. Classification and Regression Tree (CART) (Breiman *et al.*, 1984),
3. Gradient Tree Boosting (Friedman, 2002),
4. k-Nearest Neighbours (Dudani, 1976),
5. Multilayer Perceptron (MLP) with backpropagation (Widrow and Lehr, 1990),
6. Naïve–Bayes (Lewis, 1998),
7. Random Forest (Breiman, 2001),
8. Support-Vector Machine (SVM) with linear kernel (Scholkopf and Smola, 2001),
9. Support-Vector Machine with 1st degree polynomial kernel (Scholkopf and Smola, 2001),
10. Support-Vector Machine with 2nd degree polynomial (Scholkopf and Smola, 2001).

We trained and tested all these algorithms upon all three datasets. Later we ranked these algorithms by their classification accuracy measure on different datasets using three different ranking techniques while testing the results for a statistically significant difference using Welch's T-Test.

The rest of the paper is organized as follows: In Section 2 we give a review of related work. In Section 3 we describe our research methodology. In Section 4 we report our experiment results. We conclude the paper in Section 5.

2. Related Works

The scientific community has spent a lot of effort to tackle the problem of phishing websites detection. In general, approaches to solving this problem can be grouped into three different categories: (i) blacklisting and heuristic-based approaches (more in Section 2.1), (ii) supervised machine learning approaches (more in Section 2.2), and deep learning approaches (more in Section 2.3) (Sahoo *et al.*, 2017).

2.1. *Review of Blacklisting and Heuristics-Based Research*

Although there are initiatives to use centralized phishing websites' URLs blacklisting solutions (e.g., PhishTank,[1] Google Safe Browsing API[2]), this method was proven unsuccessful as it takes time to detect and report a malicious URL, because phishing websites

[1] https://www.phishtank.com/.

[2] https://developers.google.com/safe-browsing/.

have a very short lifespan (from a few hours to a few days) (Verma and Das, 2017). Therefore, new phishing websites' URL detection methods were started to be implemented by the scientific community.

Heuristic approaches are an improvement on blacklisting techniques where the signatures of common attacks are identified and blacklisted for the future use of Intrusion Detection Systems (Seifert *et al.*, 2008). Heuristic methods supersede conventional blacklisting methods as they have better generalization capabilities and can detect threats in new URLs, but they cannot generalize to all types of new threats (Verma and Das, 2017).

2.2. *Review of Supervised Machine Learning Based Research*

During the last decade, most of the machine learning approaches to solve phishing websites detection problem were based on the supervised machine learning methods on phishing datasets with predefined features. In Table 1, we present a detailed summary of other authors' results of this problem solving during the last ten years of study. Our review consists of the publication year, authors, used classifier, dataset composition (numbers of phishing and legitimate websites), and achieved classification accuracy. Results are sorted by accuracy from highest to lowest.

From this review, we can make the following observations:

- Two best approaches scored as high as 99.9% by accuracy.
- 15 best approaches scored above 99.0% by accuracy.
- The most popular algorithms among researchers are Random Forest (8 papers), Naïve–Bayes (7 papers), SVM (7 papers), C4.5 (7 papers[3]), Logistic Regression (6 papers).
- Best 5 approaches scored above 99.49% and were implemented using different types of classifiers: neural networks, regression, decision trees, ensembles, and Bayesian. We see no prevailing classification method or type of method among top results.
- Best 5 approaches use highly unbalanced datasets, therefore, evaluating classifier performance by accuracy is inadequate and does not tell how this classifier would perform on more balanced datasets.

2.3. *Review of Deep Learning Based Research*

During the past few years, novel approaches to solve phishing websites detection problem using deep learning techniques were introduced by the scientific community. Zhao *et al.* have demonstrated that Gated Recurrent Neural Network (GRU) without the need for manual feature creation is capable of classifying malicious URLs with 98.5% accuracy on 240,000 phishing and 150,000 legitimate websites URL samples (Zhao *et al.*, 2019). Saxe and Berlin have performed an experiment with Convolutional Neural Network (CNN), automating the process of feature design and extraction from generic raw character strings (malicious URLs, file paths, etc.) and gaining 99.30% accuracy on 19,067,879 randomly

[3]Including J48, which is WEKA's class for generating pruned or unpruned C4.5 decision tree (http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html).

Table 1
Classification approaches to the solution of the phishing websites detection problem.

Reference	Classifier	Dataset # phish.	# legit.	Accuracy
(Marchal *et al.*, 2017)	Gradient Boosting	100,000	1000	99.90%
(Whittaker *et al.*, 2010)	Logistic Regression	16,967	1,499,109	99.90%
(Xiang *et al.*, 2011)	Bayesian Network	8,118	4,780	99.60%
(Cui *et al.*, 2018)	C4.5[4]	24,520	138,925	99.78%
(Zhao and Hoi, 2013)	Classic Perceptron	990,000	10,000	99.49%
(Patil and Patil, 2018)	Random Forest	26,041	26,041	99.44%
(Zhao and Hoi, 2013)	Label Efficient Perceptron	990,000	10,000	99.41%
(Chen *et al.*, 2014)	Logistic Regression	1,945	404	99.40%
(Cui *et al.*, 2018)	SVM	24,520	138,925	99.39%
(Patil and Patil, 2018)	Fast Decision Tree Learner (REPTree)	26,041	26,041	99.19%
(Zhao and Hoi, 2013)	Cost-sensitive Perceptron	990,000	10,000	99.18%
(Patil and Patil, 2018)	CART[5]	26,041	26,041	99.15%
(Jain and Gupta, 2018b)	Random Forest	2,141	1,918	99.09%
(Patil and Patil, 2018)	J48[6]	26,041	26,041	99.03%
(Verma and Dyer, 2015)	J48	11,271	13,274	99.01%
(Verma and Dyer, 2015)	PART[7]	11,271	13,274	98.98%
(Verma and Dyer, 2015)	Random Forest	11,271	13,274	98.88%
(Shirazi *et al.*, 2018)	Gradient Boosting	1,000	1,000	98,78%
(Cui *et al.*, 2018)	Naïve–Bayes	24,520	138,925	98,72%
(Cui *et al.*, 2018)	C4.5	356,215	2,953,700	98.70%
(Patil and Patil, 2018)	Alternating Decision Tree	26,041	26,041	98.48%
(Shirazi *et al.*, 2018)	SVM (Linear)	1,000	1,000	98,46%
(Shirazi *et al.*, 2018)	CART	1,000	1,000	98,42%
(Adebowale *et al.*, 2019)	Adaptive Neuro-Fuzzy Inference System	6,843	6,157	98.30%
(Vanhoenshoven *et al.*, 2016)	Random Forest	1,541,000	759,000	98.26%
(Jain and Gupta, 2018b)	Logistic Regression	2,141	1,918	98.25%
(Patil and Patil, 2018)	Random Tree	26,041	26,041	98.18%
(Shirazi *et al.*, 2018)	k-Nearest Neighbuors	1,000	1,000	98,05%
(Vanhoenshoven *et al.*, 2016)	Multi Layer Perceptron	1,541,000	759,000	97.97%
(Verma and Dyer, 2015)	Logistic Regression	11,271	13,274	97.70%
(Jain and Gupta, 2018b)	Naïve–Bayes	2,141	1,918	97.59%
(Vanhoenshoven *et al.*, 2016)	k-Nearest Neighbours	1,541,000	759,000	97.54%
(Shirazi *et al.*, 2018)	SVM (Gaussian)	1,000	1,000	97,42%
(Vanhoenshoven *et al.*, 2016)	C5.0[8]	1,541,000	759,000	97.40%
(Karabatak and Mustafa, 2018)	Random Forest	6,157	4,898	97.34%
(Vanhoenshoven *et al.*, 2016)	C4.5	1,541,000	759,000	97.33%
(Vanhoenshoven *et al.*, 2016)	SVM	1,541,000	759,000	97.11%
(Karabatak and Mustafa, 2018)	Multilayer Perceptron	6,157	4,898	96.90%
(Karabatak and Mustafa, 2018)	Logistic Model Tree (LMT)	6,157	4,898	96.87%
(Karabatak and Mustafa, 2018)	PART	6,157	4,898	96.76%
(Karabatak and Mustafa, 2018)	ID3[9]	6,157	4,898	96.49%
(Zhao *et al.*, 2019)	Random Forest	40,000	150,000	96.40%
(Karabatak and Mustafa, 2018)	Random Tree	6,157	4,898	96.37%
(Chiew *et al.*, 2019)	Random Forest	5,000	5,000	96.17%
(Jain and Gupta, 2018b)	SVM	2,141	1,918	96.16%
(Vanhoenshoven *et al.*, 2016)	Naïve–Bayes	1,541,000	759,000	95.98%

(*continued on next page*)

Table 1
(*continued*)

Reference	Classifier	Dataset # phish.	# legit.	Accuracy
(Shirazi *et al.*, 2018)	Naïve-Bayes	1,000	1,000	95,97%
(Karabatak and Mustafa, 2018)	J48	6,157	4,898	95.87%
(Ma *et al.*, 2009)	Logistic Regression	20,500	15,000	95.50%
(Karabatak and Mustafa, 2018)	JRip[10]	6,157	4,898	95.01%
(Marchal *et al.*, 2014)	Random Forest	48,009	48,009	94.91%
(Verma and Dyer, 2015)	SVM	11,271	13,274	94.79%
(Chiew *et al.*, 2019)	C4.5	5,000	5,000	94.37%
(Karabatak and Mustafa, 2018)	Randomizable Filtered Classifier	6,157	4,898	94.21%
(Chiew *et al.*, 2019)	JRip	5,000	5,000	94.17%
(Chiew *et al.*, 2019)	PART	5,000	5,000	94.13%
(Zhang *et al.*, 2017)	Extreme Learning Machines (ELM)	2,784	3,121	94.04%
(Karabatak and Mustafa, 2018)	Stochastic Gradient Descent	6,157	4,898	93.95%
(Karabatak and Mustafa, 2018)	Naïve–Bayes	6,157	4,898	93.39%
(Karabatak and Mustafa, 2018)	Bayesian Network	6,157	4,898	92.98%
(Chiew *et al.*, 2019)	SVM	5,000	5,000	92.20%
(Thomas *et al.*, 2011)	Logistic Regression	500,000	500,000	90.78%
(Chiew *et al.*, 2019)	Naïve–Bayes	5,000	5,000	84.10%
(Verma and Dyer, 2015)	Naïve–Bayes	11,271	13,274	83.88%

sampled websites URLs (Saxe and Berlin, 2017). Vazhayil *et al.* have performed a comparative study, demonstrating the 98.7% accuracy of CNN and 98.9% accuracy of CNN Long Short-Term Memory (CNN-LSTM) deep learning networks on 116,101 URL samples (Vazhayil *et al.*, 2018). Selvaganapathy *et al.* have implemented a method where feature selection is made using Greedy Multilayer Deep Belief Network (DBN) and binary classification is done using Deep Neural Networks (DNN), capable of classifying malicious URLs with 75.0% accuracy on 17.700 phishing and 10,000 legitimate websites URL samples (Selvaganapathy *et al.*, 2018).

3. Research Methodology

In this section, we describe our research methodology by defining:

- experimental design for our research (Section 3.1),

[4]C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan.
[5]Classification And Regression Tree.
[6]WEKA's class for generating a pruned or unpruned C4.5 decision tree.
[7]Rule based learner which combines C4.5 trees and RIPPER learning.
[8]C5.0 is an algorithm used to generate a decision tree developed by Ross Quinlan.
[9]ID3 (Iterative Dichotomiser 3) is an algorithm used to generate a decision tree developed by Ross Quinlan.
[10]A propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which was proposed by William W. Cohen.

- algorithms used in the experiment and grounds for algorithm selection (Section 3.2),
- datasets used in the experiment (Section 3.3),
- metrics (i.e. Classification Accuracy), methods (i.e. T-test, ranking techniques, etc.), used in the experiment (Section 3.4).

We discuss the validity of our results in Section 3.5.

3.1. *Experimental Design*

In this subsection, we present our experimental design, employed to perform the experiment, and answer the research question. The experiment was divided into three parts: (i) training the classifiers for each dataset, (ii) ranking the classifiers, and (iii) creating unified classifier ranking.

3.1.1. *Part I: Training the Classifiers*

The objective of this part is to train all the classifiers from Section 3.2 on all the datasets from Section 3.3 for their best possible classification accuracy described in Section 3.4.1, formula (2). For every dataset, for every classifier, we take the following steps:

1. Set up the classifier for a specific dataset in Python's environment using the Scikit Learn library (Pedregosa *et al.*, 2011).
2. Manually select a set of hyper-parameters, referring to Scikit Learn's user guide.
3. Train and test the classifier using Scikit Learn's cross-validation (CV) function, with 30 stratified folds.
4. Plot learning curves.
5. Analyse learning curves and make a decision on tuning the hyper-parameters by answering the following questions:
    - Is the algorithm learning on training data or memorizing it? If the training curve is flat at 100%, then the algorithm is not learning but memorizing the data. To solve this issue, we take actions, e.g. reduce the number of weak learners in an ensemble, reduce the depth of the tree, increase the regularization parameter, etc.
    - Is the algorithm prone to overfitting (low bias, high variance) or underfitting (high bias, low variance), or learns "just right"? If the gap between training and CV curves is small, the algorithm is underfitting; if the gap is big, it is overfitting. To solve this issue we take actions to reduce high bias or high variance, e.g. (i) add more training examples, use a smaller set of features, increase the regularization parameter, etc. to reduce high variance, and (ii) use bigger set of features, add polynomial features, increase the number of layers in the neural network, reduce the regularization parameter, etc. to reduce high bias.

    If the decision is made to tune the hyper-parameters to avoid high bias or high variance, then we start over from Step 2; if not, we go to Step 6.
6. Perform a Wilk–Shapiro test, as described in Section 3.4, formula (4) to check if the accuracies of classifier's classification from 30-fold CV testing are normally distributed. If not, take action to normalize the values.
7. Save the results for further actions.

We finish this part when all the classifiers are trained on all the datasets, and we have normally distributed sets of classification accuracies for each classifier on each dataset.

3.1.2. *Part II: Ranking the Classifiers*

The objective of this part is to rank all the classifiers by their classification results within each individual dataset.

For every dataset, we take the following steps:

1. Using Welch's T-test, described in Section 3.4.2, formula (3), check every possible pair of classifiers if their classification results produced in Part I have statistically significant differences. The classification results are distributed by normal distribution.
2. Arrange all classifiers by their mean classification accuracy in descending order.
3. Assign each classifier three ranks using ranking techniques described in Section 3.4.4. **Important notice:** classifiers whose results have no statistically significant differences receive the same rank.
4. For each ranking technique, distribute points from the highest 10 to the lowest 1 for each classifier, depending on the received rank. Points are calculated using formula (1).

$$Points_i = N_{methods} - Rank_i + 1, \tag{1}$$

   where

   - $N_{methods}$ – number of algorithms participating in ranking,
   - $Rank_i$ – rank of i^{th} algorithm.

5. Save the results for further actions.

We finish this part when all the classifiers receive ranks and ranking points by all ranking techniques on all datasets.

3.1.3. *Part III: Creating the Unified Classifier Ranking*

The objective of this part is to summarize the performance of selected classifiers on all datasets by creating a unified ranking. To do this, we combine rankings for each classifier by adding all the points received upon all datasets. Our experiment is complete after finishing this part.

3.2. *Algorithms*

In the review of supervised machine learning approaches in Section 2.2, we showed that five best implementations employ different classifiers from separate types of supervised machine learning algorithms: neural networks, decision trees, ensembles, regression, and Bayesian. We also disclosed that the top 3 classifiers by popularity are Random Forest (8 papers), Naïve–Bayes (7 papers), SVM (7 papers).

For our research, we built the set of algorithms consisting of:

- three most popular algorithms from the review of related works (Section 2.2),

- five more algorithms from the Scikit Learn library, belonging to the best performing types of classifiers in the review of related works (Section 2.2).

All possible classical classification algorithms were not used due to the limitation of resources available for this research.

Therefore, in our experiment, we chose to use classic supervised machine learning algorithms such as AdaBoost, Classification and Regression Tree, Gradient Tree Boosting, k-Nearest Neighbours, Multilayer Perceptron with backpropagation, Naïve–Bayes, Random Forest, and Support-Vector Machine.

3.3. *Datasets*

In our experiment, we used three publicly available phishing websites datasets with predefined features. To our knowledge, these are the only phishing datasets with predefined features made publicly available by other researchers.

3.3.1. *UCI-2015*

UCI-2015 dataset from UCI repository[11] was donated in March 2015 by Mohammad, McCluskey (University of Huddersfield), and Thabtah (Canadian University of Dubai). This dataset contains 6,157 phishing and 4,898 legitimate website samples. A total of 30 different URLs, DNS, HTML, JavaScript, and External statistics based features were extracted from these websites.

3.3.2. *UCI-2016*

UCI-2016 dataset from UCI repository,[12] contributed by Abdelhamid (Auckland Institute of Studies) in November 2016. This dataset contains 805 phishing and 548 legitimate website samples. A total of 9 features were extracted from these websites.

3.3.3. *MDP-2018*

MDP-2018 dataset from Mendeley Data portal[13] was published by Choon Lin Tan (Universiti Malaysia Sarawak) in March 2018. This balanced dataset contains 5,000 phishing and 5,000 legitimate website samples. A total of 48 features were extracted from these websites.

3.4. *Measures and Methods*

3.4.1. *Classification Accuracy*

Classification accuracy in our experiment is the rate of phishing and legitimate websites which are identified correctly with respect to all the websites, defined as follows:

$$ACCURACY = \frac{TP + TN}{TP + FP + TN + FN}, \tag{2}$$

[11] https://archive.ics.uci.edu/ml/datasets/phishing+websites.
[12] https://archive.ics.uci.edu/ml/datasets/Website+Phishing.
[13] https://data.mendeley.com/datasets/h3cgnj8hft/1.

where

- *TP* – number of websites, correctly detected as phishing (True Positive),
- *TN* – number of websites, correctly detected as benign (True Negative),
- *FP* – number of legitimate websites, incorrectly detected as phishing (False Positive),
- *FN* – number of phishing websites, incorrectly detected as legitimate (False Negative).

We chose classification accuracy as our classification quality quantification metric because: (i) most other researchers use classification accuracy to define results of their experiments (see Section 2), therefore the comparability of research results is homogeneous throughout our work; (ii) in our experiment we used datasets with equal or close to equal class distributions (there is no significant disparity between the number of positive and negative labels), therefore we do not have the majority and minority classes; (iii) we used cross-validation function with stratification option which generates test sets such that all contain the same distribution of classes, or as close as possible; (iv) we do not directly compare classification results of different datasets by accuracy and do not draw any conclusions from this information; to distinguish top classifiers we employ ranking techniques (see Section 3.4.4). In these circumstances, classification accuracy is a useful non-bias measure.

3.4.2. *Welch's T-Test*

Welch's T-test in our experiment is used to determine whether the means of classification accuracy results produced by any two classifiers within the same dataset have a statistically significant difference. The two-sample T-test for unpaired data is defined as follows (Snedecor and Cochran, 1989).

Let $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ be two independent samples from normal distributions, and μ_x, μ_y be the means of these distributions. Then, the hypothesis to be tested is defined as

$$H_0 : \mu_x = \mu_y \text{ vs. } H_A : \mu_x \neq \mu_y.$$

The test statistic for testing the hypothesis is calculated as follows:

$$T = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{S_x^2}{n} + \frac{S_y^2}{m}}}, \tag{3}$$

where

- $\bar{X}$ and $\bar{Y}$ are the sample $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ means,
- n and m are the sample $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ sizes,
- S_x^2 and S_y^2 are the sample $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ variances.

We reject the *null* hypothesis H_0 that the two means are equal if $|T| > t_{1-\alpha/2,v}$, where $t_{1-\alpha/2,v}$ is the critical value of the t distribution with v degrees of freedom with our chosen $\alpha = 0.05$. Welch's T-test can only be performed on samples from normal distributions. We used *scipy.stats* package for Python to perform a T-test.

3.4.3. *Shapiro–Wilk Test*

Shapiro–Wilk test is used to check whether samples came from a normally distributed population (Shapiro and Wilk, 1965). This test is defined as follows:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}, \tag{4}$$

where

- $x_{(i)}$ are the ordered sample values, $x_{(1)}$ being the smallest,
- a_i are constants generated from the means, variances and covariances of the order statistics of a sample of size n from normal distributions,
- $\bar{x}$ is the sample mean,
- n is the sample size.

We reject the *null* hypothesis H_0 that the sample belongs to normal distribution if $W < W_\alpha$, where W_α is the critical threshold. We used *scipy.stats* package for Python to perform a Shapiro–Wilk test.

3.4.4. *Ranking Techniques*

Ranking techniques used in our research are:

1. Standard Competition Ranking (SCR), where equal items get the same ranking number, and then a gap is left in the ranking numbers, i.e. “1224” ranking.
2. Dense Ranking (DR), where equal get the same ranking number, and the next item gets the immediately following ranking number, i.e. “1223” ranking.
3. Fractional Ranking (FR), where equal items get the same ranking number, which is the mean of what they would have under ordinal rankings, i.e. “1 2.5 2.5 4” ranking.

3.5. *Validity*

In our experiment we used classification accuracy measure described in Section 3.4.1, formula (2) and balanced datasets (see Section 3.3). Classification accuracy has a high construct validity on balanced datasets.

We used the cross-validation procedure with 30 stratified folds to evaluate classification accuracy, which provides an objective measure of how well the model fits and how well it will generalize to new data.

Welch’s T-test was used to measure if the means of classification accuracy results produced by any two classifiers within the same dataset have a statistically significant difference. This test eliminated the possibility to miss-rank the classifiers, whose results had no statistically significant differences.

Three different ranking techniques were introduced to overcome ranking bias, where distinct ranking techniques give different outcomes.

We provide the source code of our experiment to other researchers at https://github.com/PauliusVaitkevicius/Exp001.

4. Results

In this section we present our experiment results based on the research methodology described in Section 3.

First, we configured selected classification algorithms (described in Section 3.2) for each dataset (described in Section 3.3). We used implementations for all selected algorithms from the Scikit Learn library (version 0.20.1) in Python (version 3.7.1), which provides open-source tools for data mining and data analysis (Pedregosa *et al.*, 2011). Later, we chose the best fitting hyper-parameters for each algorithm on each dataset with 30-fold cross validation, following our experimental design, described in Section 3.1. Selected best hyper-parameters for each classifier are described in Table 2. Selected best hyper-parameters differ in algorithm configurations for different datasets due to applying the hyper-parameter selection technique, described in Section 3.1 Part I, to datasets with different designs and data quantities.

Subsequently, we trained and tested all the classifiers chosen for this experiment on all the datasets. We measured classification performance by accuracy: the ratio of phishing and legitimate URLs, which are classified correctly with respect to all the URLs in the dataset as described in Section 3.4.1, formula (2). Classification results are given in Table 3. Initial results showed that the Gradient Tree Boosting algorithm performed best on MDP-2018 and UCI-2016 datasets, and Multilayer Perceptron with backpropagation performed best on the UCI-2015 dataset.

Later, we evaluated all classification results against each other within an individual dataset using Welch's T-test, as described in Section 3.4.2, formula (3), to check if they have statistically significant differences. Afterward, we ordered all the classifiers by their performance upon each dataset using three different ranking techniques: SCR, FR, and DR, as described in Section 3.1. Classifiers, whose results had no statistically significant differences, were given equal ranks. Next, points from the highest 10 to the lowest 1 were distributed to each classifier depending on the assigned rank.

Ranking results for the UCI-2015 dataset are presented in Table 4, with Multilayer Perceptron ranking in the first place for all ranking techniques.

Results for the UCI-2016 dataset are presented in Table 5, showing Multilayer Perceptron, Gradient Tree Boosting, CART, and Random Forest all scoring maximum points, as their classification accuracy had no statistically significant difference.

And last, ranking results for the MDP-2018 dataset are presented in Table 6, with Gradient Tree Boosting, AdaBoost, and Random Forest all ranking in the first place for all ranking techniques.

Finally, combined dataset rankings were calculated in Table 7, summing up all the points each classifier has scored for each dataset, showing various sets of algorithms ending up in the 1st place with different ranking techniques. If we rank results using the Standard Competition Ranking technique, we get Random Forest and Gradient Tree Boosting ranked at the top. If we rank results using the Fractional Ranking technique, we get Multilayer Perceptron ranked at the top. If we rank results using the Dense Ranking technique, we get Random Forest, Multilayer Perceptron, and Gradient Tree Boosting ranked at the top. There is no single algorithm ranked at the top using all three ranking techniques.

Table 2
Hyper-parameters used in the experiment.

Algorithm	UCI-2015	UCI-2016	MDP-2018
AdaBoost	*# of estimators*: 200	*# of estimators*: 50 *Algorithm*: SAMME;	*# of estimators*: 200
CART	*Max tree depth*: 9; *Split evaluation criteria*: entropy; *Min samples at leaf node*: 2;	*Max tree depth*: 9; *Split evaluation criteria*: entropy; *Min samples at leaf node*: 2;	*Max tree depth*: 5; *Split evaluation criteria*: entropy; *Min samples at leaf node*: 2;
Gradient Tree Boosting	*Max estimator depth*: 1; *Learning rate*: 1;	*Min samples at leaf node*: 2; *Learning rate*: 1; *# of estimators*: 50;	*Max estimator depth*: 1; *Learning rate*: 1;
k-Nearest Neighbours	*Number of neighbours:* 5; *Weights:* uniform weights – all points in each neighbourhood are weighted equally; *Algorithm:* auto;	*Number of neighbours:* 5; *Weights:* uniform weights – all points in each neighbourhood are weighted equally; *Algorithm:* auto;	*Number of neighbours:* 5; *Weights:* uniform weights – all points in each neighbourhood are weighted equally; *Algorithm:* auto;
Naïve–Bayes	Multivariate Bernoulli models;	Multivariate Bernoulli models;	Multivariate Bernoulli models;
Multilayer Perceptron	*Hidden layers*: 30; *Number of max iterations*: 3000;	*Hidden layers*: 150; *Number of max iterations*: 1000;	*Hidden layers*: 100; *Number of max iterations*: 1000;
Random Forest	*# of estimators*: 7; *Max tree depth*: 11; *Split evaluation criteria*: entropy;	*# of estimators*: 7; *Max tree depth*: 8; *Split evaluation criteria*: entropy;	*# of estimators*: 7; *Max tree depth*: 11; *Split evaluation criteria*: entropy;
SVM with linear kernel	*Penalty parameter C:* 1.0; *Kernel:* Linear;	*Penalty parameter C:* 1.0; *Kernel:* Linear;	*Penalty parameter C:* 1.0; *Kernel:* Linear;
SVM with 1st deg. pol. kernel	*Penalty parameter C:* 1.0; *Kernel:* Polynomial; *Degree of the polynomial kernel function:* 1;	*Penalty parameter C:* 1.0; *Kernel:* Polynomial; *Degree of the polynomial kernel function:* 1;	*Penalty parameter C:* 1.0; *Kernel:* Polynomial; *Degree of the polynomial kernel function:* 1;
SVM with 2nd deg. pol. kernel	*Penalty parameter C:* 1.0; *Kernel:* Polynomial; *Degree of the polynomial kernel function:* 2;	*Penalty parameter C:* 1.0; *Kernel:* Polynomial; *Degree of the polynomial kernel function:* 2;	*Penalty parameter C:* 1.0; *Kernel:* Polynomial; *Degree of the polynomial kernel function:* 2;

Table 3
Classification results by average classification accuracy.

Algorithm	UCI-2015	UCI-2016	MDP-2018
AdaBoost	0.9352	0.8495	0.9728
CART	0.9363	0.8930	0.9574
Gradient Tree Boosting	0.9381	**0.9034**	**0.9742**
k-Nearest Neighbours	0.9481	0.8641	0.8564
Naïve–Bayes	0.9057	0.8225	0.9177
Multilayer Perceptron	**0.9722**	0.9028	0.9671
Random Forest	0.9525	0.8916	0.9715
SVM with linear kernel	0.9271	0.8365	0.9422
SVM with 1st deg. pol. kernel	0.9257	0.8328	0.9334
SVM with 2nd deg. pol. kernel	0.9388	0.7152	0.9549

Table 4
Classifier rankings on UCI-2015 dataset.

SCR rank	FR rank	DR rank	Algorithm	SCR points	FR points	DR points
1	1	1	Multilayer Perceptron	10	10	10
2	4.5	2	Random Forest	9	6.5	9
2	4.5	2	k-Nearest Neighbours	9	6.5	9
2	4.5	2	SVM with 2nd deg. pol. kernel	9	6.5	9
2	4.5	2	Gradient Tree Boosting	9	6.5	9
2	4.5	2	CART	9	6.5	9
2	4.5	2	AdaBoost	9	6.5	9
8	8.5	3	SVM with linear kernel	3	2.5	8
8	8.5	3	SVM with 1st deg. pol. kernel	3	2.5	8
10	10	4	Naïve–Bayes	1	1	7

Table 5
Classifier rankings on UCI-2016 dataset.

SCR rank	FR rank	DR rank	Algorithm	SCR points	FR points	DR points
1	2.5	1	Gradient Tree Boosting	10	8.5	10
1	2.5	1	Multilayer Perceptron	10	8.5	10
1	2.5	1	CART	10	8.5	10
1	2.5	1	Random Forest	10	8.5	10
5	7	2	k-Nearest Neighbours	6	4	9
5	7	2	AdaBoost	6	4	9
5	7	2	SVM with linear kernel	6	4	9
5	7	2	SVM with 1st deg. pol. kernel	6	4	9
5	7	2	Naïve–Bayes	6	4	9
10	10	3	SVM with 2nd deg. pol. kernel	1	1	8

Table 6
Classifier rankings on MDP-2018 dataset
.

SCR rank	FR rank	DR rank	Algorithm	SCR points	FR points	DR points
1	2	1	Gradient Tree Boosting	10	9	10
1	2	1	AdaBoost	10	9	10
1	2	1	Random Forest	10	9	10
4	4	2	Multilayer Perceptron	7	7	9
5	5.5	3	CART	6	5.5	8
5	5.5	3	SVM with 2nd deg. pol. kernel	6	5.5	8
7	7.5	4	SVM with linear kernel	4	3.5	7
7	7.5	4	SVM with 1st deg. pol. kernel	4	3.5	7
9	9	5	Naïve–Bayes	2	2	6
10	10	6	k-Nearest Neighbours	1	1	5

Table 7
Combined classifier rankings

Algorithm	SCR points	FR points	DR points
Multilayer Perceptron	27	25.5	29
Gradient Tree Boosting	29	24	29
Random Forest	29	24	29
AdaBoost	25	19.5	28
CART	25	20.5	27
SVM with 2nd deg. pol. kernel	16	13	25
k-Nearest Neighbours	16	11.5	23
SVM with linear kernel	13	10	24
SVM with 1st deg. pol. kernel	13	10	24
Naïve–Bayes	9	7	22

5. Conclusions

In this paper, we provide an answer to our research question: *which classical classification algorithm is best for solving the phishing websites detection problem, on all publicly available datasets with predefined features?* From our research, we make the following conclusions:

1. Neural Networks, in our case Multilayer Perceptron and ensemble type algorithms (Random Forest, Gradient Tree Boosting, and AdaBoost) perform best for solving the phishing websites detection problem, on datasets used in the experiment.
2. Instance similarity-based and Bayesian classifiers, i.e. SVM, k-Nearest Neighbours, and Naïve–Bayes performance is the poorest for solving the phishing websites detection problem, regardless of a specific dataset design.
3. Results discussed in conclusions #1 and #2 coincide with general trends in related works review (Section 2.2): best classification results are achieved with neural networks, decision trees, and ensemble types of classification algorithms.
4. Classifiers showing above a 99.0% classification accuracy on highly unbalanced datasets in related works review (Section 2.2), i.e. Random Forest, SVM, Perceptron, and CART did not score such high accuracy on any balanced dataset in our experiment.

In future work, hyper-parameter tuning can be automated using the Grid Search algorithm instead of manual expert hyper-parameter evaluation.

References

Adebowale, M., Lwin, K., Sánchez, E., Hossain, M. (2019). Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. *Expert Systems with Applications*, 115, 300–313.

Anti-Phishing Working Group, I. (2018). Phishing Activity Trends Reports.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984). *Classification and regression trees*. Wadsworth International Group, Belmont, CA, p. 432.

Chen, T.C., Stepan, T., Dick, S., Miller, J. (2014). An anti-phishing system employing diffused information. *ACM Transactions on Information and System Security*, 16(4), 1–31.

Chiew, K.L., Tan, C.L., Wong, K., Yong, K.S., Tiong, W.K. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Information Sciences*, 484, 153–166.

Cui, B., He, S., Yao, X., Shi, P., Yao, X., He, S., Cui, B. (2018). Malicious URL detection with feature extraction based on machine learning. *International Journal of High Performance Computing and Networking*, 12(2), 166.

Dudani, S.A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4), 325–327.

Friedman, J.H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4), 367–378.

Internet Crime Complaint Center (2019). *2018 Internet Crime Report*. Tech. Rep., Internet Crime Complaint Center at the Federal Bureau of Investigation of United States of America.

Jain, A.K., Gupta, B.B. (2018a). A machine learning based approach for phishing detection using hyperlinks information. *Journal of Ambient Intelligence and Humanized Computing*, 1–14.

Jain, A.K., Gupta, B.B. (2018b). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, 68(4), 687–700.

Karabatak, M., Mustafa, T. (2018). Performance comparison of classifiers on reduced phishing website dataset. In: *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*. IEEE, pp. 1–5.

Lewis, D.D. (1998). Naive (Bayes) at forty: the independence assumption in information retrieval. In: *ECML 1998: Machine Learning: ECML-98*. Springer, Berlin, Heidelberg, pp. 4–15.

Lin Tan, C., Leng Chiew, K., Wong, K.S., Nah Sze, S., Tan, C.L., Chiew, K.L., Wong, K.S., Sze, S.N. (2016). PhishWHO: phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems*, 88, 18–27.

Ma, J., Saul, L.K., Savage, S., Voelker, G.M. (2009). Beyond blacklists. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD '09*. ACM Press, New York, USA, p. 1245.

Marchal, S., Armano, G., Grondahl, T., Saari, K., Singh, N., Asokan, N. (2017). Off-the-hook: an efficient and usable client-side phishing prevention application. *IEEE Transactions on Computers*, 66(10), 1717–1733.

Marchal, S., Francois, J., State, R., Engel, T. (2014). Phish storm: detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4), 458–471.

Marchal, S., Saari, K., Singh, N., Asokan, N. (2016). Know your phish: novel techniques for detecting phishing sites and their targets. In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, pp. 323–333.

Patil, D.R., Patil, J.B. (2018). Malicious URLs detection using decision tree classifiers and majority voting technique. *Cybernetics and Information Technologies*, 18(1), 11–29.

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., Duchesnay, É. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Sahoo, D., Liu, C., Hoi, S.C.H. (2017). *Malicious URL Detection using Machine Learning: A Survey*.

Saxe, J., Berlin, K. (2017). eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint arXiv:1702.08568.

Scholkopf, B., Smola, A.J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Seifert, C., Welch, I., Komisarczuk, P. (2008). Identification of malicious web pages with static heuristics. In: *2008 Australasian Telecommunication Networks and Applications Conference*. IEEE, pp. 91–96.

Selvaganapathy, S., Nivaashini, M., Natarajan, H. (2018). Deep belief network based detection and categorization of malicious URLs. *Information Security Journal: A Global Perspective*, 27(3), 145–161.

Shapiro, S.S., Wilk, M.B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3–4), 591–611.

Shirazi, H., Bezawada, B., Ray, I. (2018). "Kn0w Thy Doma1n Name". In: *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies – SACMAT '18* Vol. 18. ACM Press, New York, USA, pp. 69–75.

Snedecor, G.W., Cochran, W.G. (1989). *Statistical Methods*, eight ed. Iowa State University Press, Ames, Iowa.

Thomas, K., Grier, C., Ma, J., Paxson, V., Song, D. (2011). Design and evaluation of a real-time URL spam filtering service. In: *2011 IEEE Symposium on Security and Privacy*. IEEE, pp. 447–462.

Vanhoenshoven, F., Napoles, G., Falcon, R., Vanhoof, K., Koppen, M. (2016). Detecting malicious URLs using machine learning techniques. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 1–8.

Vazhayil, A., Vinayakumar, R., Soman, K. (2018). Comparative study of the detection of malicious URLs using shallow and deep networks. In: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, pp. 1–6.

Verma, R., Das, A. (2017). What's in a URL. In: *Proceedings of the 3rd ACM on International Workshop on Security And PrivacyAnalytics – IWSPA '17*. ACM Press, New York, USA, pp. 55–63.

Verma, R., Dyer, K. (2015). On the character of phishing URLs. In: *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy – CODASPY '15*. ACM Press, New York, USA, pp. 111–122.

Wang, R. (2012). AdaBoost for feature selection, classification and its relation with SVM, a review. *Physics Procedia*, 25, 800–807.

Whittaker, C., Ryner, B., Nazif, M. (2010). Large-scale automatic classification of phishing pages. In: *The 17th Annual Network and Distributed System Security Symposium (NDSS '10)*.

Widrow, B., Lehr, M.A. (1990). 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9), 1415–1442.

Xiang, G., Hong, J., Rose, C.P., Cranor, L. (2011). CANTINA+: a feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security*, 14(2), 1–28.

Zhang, W., Jiang, Q., Chen, L., Li, C. (2017). Two-stage ELM for phishing Web pages detection using hybrid features. *World Wide Web*, 20(4), 797–813.

Zhao, J., Wang, N., Ma, Q., Cheng, Z. (2019). Classifying malicious urls using gated recurrent neural networks. In: *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Springer, pp. 385–394.

Zhao, P., Hoi, S.C. (2013). Cost-sensitive online active learning with application to malicious URL detection. In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD '13*. ACM Press, New York, USA, p. 919.

P. Vaitkevicius is a doctoral student at Vilnius University, Institute of Data Science and Digital Technologies. His research interests include machine learning, artificial intelligence, cybersecurity, and natural language processing.

V. Marcinkevicius in 2010 received a doctoral degree in computer science (PhD) from Vytautas Magnus University. Since 2001 he is an employee of Vilnius University, Institute of Data Science and Digital Technologies. His present employment is senior researcher and the head or intelligent technologies research group of the Vilnius University, Institute of Data Science and Digital Technologies. His research interests include machine learning, artificial intelligence, cybersecurity, and natural language processing. He is the author of more than 70 scientific publications. He is a member of the Lithuanian Computer Society and Lithuanian Mathematical Society.