Detecting Malicious URLs using Machine Learning Techniques

Frank Vanhoenshoven*, Gonzalo Nápoles*, Rafael Falcon†, Koen Vanhoof* and Mario Köppen‡
*Universiteit Hasselt Campus Diepenbeek
Agoralaan Gebouw D, BE3590 Diepenbeek, Belgium
Email: frank.vanhoenshoven@uhasselt.be; gonzalo.napoles@uhasselt.be; koen.vanhoof@uhasselt.be
†Electrical Engineering and Computer Science, University of Ottawa
800 King Edward Ave., Ottawa, ON K1N 6N5, Canada
Email: rfalcon@uottawa.ca
‡Network Design and Research Center, Kyushu Institute of Technology
680-4, Kawazu, Iizuka, Fukuoka 820-8502, Japan
Email: mkoeppen@ieee.org

***Abstract*—The World Wide Web supports a wide range of criminal activities such as spam-advertised e-commerce, financial fraud and malware dissemination. Although the precise motivations behind these schemes may differ, the common denominator lies in the fact that unsuspecting users visit their sites. These visits can be driven by email, web search results or links from other web pages. In all cases, however, the user is required to take some action, such as clicking on a desired Uniform Resource Locator (URL). In order to identify these malicious sites, the web security community has developed blacklisting services. These blacklists are in turn constructed by an array of techniques including manual reporting, honeypots, and web crawlers combined with site analysis heuristics. Inevitably, many malicious sites are not blacklisted either because they are too recent or were never or incorrectly evaluated.**

In this paper, we address the detection of malicious URLs as a binary classification problem and study the performance of several well-known classifiers, namely Naïve Bayes, Support Vector Machines, Multi-Layer Perceptron, Decision Trees, Random Forest and k-Nearest Neighbors. Furthermore, we adopted a public dataset comprising 2.4 million URLs (instances) and 3.2 million features. The numerical simulations have shown that most classification methods achieve acceptable prediction rates without requiring either advanced feature selection techniques or the involvement of a domain expert. In particular, Random Forest and Multi-Layer Perceptron attain the highest accuracy.

I. Introduction

With the undeniable prominence of the World Wide Web as the paramount platform supporting knowledge dissemination and increased economic activity, the security aspect continues to be at the forefront of many companies and governments' research efforts.

Symantec's 2016 Internet Security Report [1] elaborates on an ample array of global threats that includes corporate data breaches, attacks on browsers and websites, spear phishing attempts, ransonmware and other types of fraudulent cyber activities. The report also unveils several cyber tricks used by the scammers. One well-known and surprisingly quite effective strategy is baiting the users to click on a malicious Uniform Resource Locator (URL), which leads to the system being somehow compromised.

In order to identify these malicious sites, the web security community has developed *blacklisting services*. These blacklists are in turn developed by a myriad of techniques including manual reporting, honeypots, and web crawlers combined with site analysis heuristics [2]. While URL blacklisting has been effective to some extent, it is rather easy for an attacker to deceive the system by slightly modifying one or more components of the URL string. Inevitably, many malicious sites are not blacklisted either because they are too recent or were never or incorrectly evaluated.

Several studies in the literature tackle this problem from a Machine Learning standpoint. That is, they compile a list of URLs that have been classified as either malicious or benign and characterize each URL via a set of attributes. Classification algorithms are then expected to learn the boundary between the decision classes.

De las Cuevas et. al. [3] reported classification rates about 96% that climbed up to 97% with a rough-set-based feature selection preprocessing step that reduced the original 12 features to 9. The labeling of each URL was done after a set of security rules dictated by the Chief Security Officer (CSO) in a company. This resulted in an imbalanced classification problem that was dealt with via undersampling. In total, 57,000 URL instances were considered after removing duplicates. The authors noticed an improvement over the results attained in their previous work [4].

Kan and Thi [5] classified web pages not by their content but using their URLs, which is much faster as no delays are incurred in fetching the page content or parsing the text. The URL was segmented into multiple tokens from which classification features were extracted. The features modeled sequential dependencies between tokens. The authors pointed to the fact that the combination of high-quality URL segmentation and feature extraction improved the classification rate over several baseline techniques. Baykan et. al. [6] pursue a similar objective: topic classification from URLs. They trained separate binary classifiers for each topic (student, faculty, course and project) and were able to improve over the best

978-1-5090-4240-1/16/$31.00 ©2016 IEEE

reported F-measure.

Ma et. al. [7] brought forth an approach to detect malicious websites from the lexical and host-based features of their URLs in light of the wealth of information they carry about the website's nature. Their system is able to sift through tens of thousands of features and identify the important URL components and metadata without requiring heavy domain expertise. The approach was evaluated with up to 30,000 instances and yielded promising results, namely a very high classification rate (95%-99%) and a low false positive rate. The same authors in [8] resorted to online algorithms in order to handle millions of URLs whose features evolve over time. A system was developed to gather real-time URL features. The authors paired it with a real-time feed of labeled URLs from a large web mail provider. A classification rate of 99% is reported on a balanced dataset using confidence-weighted learning.

Zhao and Hoi [9] avoid the (typical) class imbalance in the malicious URL detection problem and the need for a large amount of training data through their Cost-Sensitive Online Active Learning (CSOAL) framework. CSOAL queries only a small fraction of the available data (about 0.5% out of 1 million instances) and directly optimizes two cost-sensitive measures to address the class-imbalance issue. The empirical evidence indicated that their scheme achieved better or highly comparable classification performance when compared to the state-of-the-art cost-insensitive and cost-sensitive online classification algorithms using a huge amount of labeled data.

This paper addresses the detection of malicious URLs as a binary classification problem and studies the performance of several well-known classifiers, namely Naïve Bayes, Support Vector Machines, Multi-Layer Perceptron, Decision Trees, Random Forest and k-Nearest Neighbors. Furthermore, we examine and compare the results for three different feature sets, selected a set totaling 3.2 million features for 2.4 million URLs (instances).

The rest of this paper is organized as follows. Section II describes the problem under consideration and Section III outlines several popular classification techniques for solving it. The empirical results are put forth in Section IV. Concluding remarks are enunciated in Section V.

II. Problem Description

The data for this research, presented by Ma et al. [10], consists of 121 sets of URLs that have been collected for 121 different days. For Day0, 16,000 URLS have been collected, Day45 contains only 130 entries; For all other days, 20,000 URLs are registered. All URLs are characterized by multiple attributes and each is classified as either malicious or benign.

The entire dataset consists of over 2.3 million URLs, each having over 3.2 million features. The overwhelming majority of these features can be identified as binary attributes. The 64 features that are non-binary[1] contain both numerical and discrete, ordinal values.

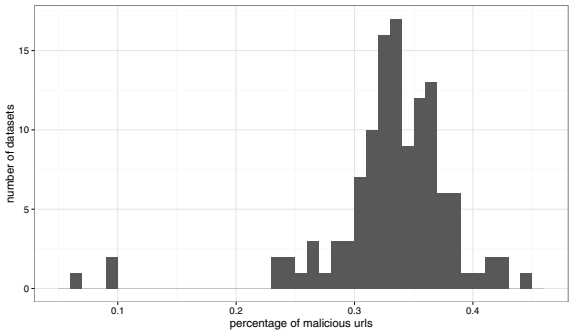

Fig. 1. Frequency of datasets having a given percentage of malicious URLs

Classification accuracy has to be achieved by relying mainly on the power of the methods used. There is no involvement of domain experts to comment on feature importance nor correlations.

For the entire dataset, roughly 33% of all URLs are classified as malicious. In Fig 1, a histogram shows the frequency (y-axis) of datasets having a given percentage of malicious URLs (x-axis). It can be seen that for the vast majority of datasets, between 30% and 40% of all URLs are malicious. The outliers to the left, containing less than 10% of malicious URLs, belong to Day34, 36, and 37. Day95 reports the highest proportion at almost 45%. Given the proportions of the classifications in each dataset, predicting *benign* for each URL in a randomly chosen test set would likely result in an accuracy of 66%. Similar predictions for test sets sampled from Days 34, 36, and 37 could even achieve an accuracy of 90% easily. As we expect classification algorithms to accurately predict the labels, we can use these proportions to assess the results.

III. Classification Techniques

Pattern classification [11] is a common scenario in real-world problems and most available models are inspired by human reasoning and behavior when facing such scenarios. This problem consist in assigning the correct category to a new observation (pattern) of the problem, given a set of alternative reponses. Patterns are normally described by a set of attributes which could be numerical, categorical or both. Formally speaking, the pattern classification problem [11] is about building a mapping $f : \mathcal{U} \rightarrow \mathcal{D}$ that assigns to each pattern/object $x \in \mathcal{U}$ described by the attribute set $\Psi = \{\psi_1 \ldots, \psi_M\}$ a decision class d from the k possible ones in $\mathcal{D} = \{D_1, \ldots, D_k\}$. Generally, the mapping is learned using a *supervised* approach, i.e., using an available set of historical data about the classification problem for training the model.

[1]X4, X5, X6, X8, X11, X16, X17, X18, X19, X21, X22, X23, X30, X33, X35, X39, X41, X43, X55, X57, X59, X61, X63, X65, X67, X69, X71, X73, X75, X77, X79, X81, X83, X85, X87, X89, X91, X93, X95, X97, X99, X101, X103, X105, X107, X109, X111, X113, X120, X126, X132, X134, X136, X138, X140, X142, X144, X146, X148, X150, X161, X194, X270, X7801

The learning process is often driven by the minimization of a cost/error metric. More complex classification problems include learning in presence of class imbalance [12] [13], class noise [14], partially labeled data [15] [16] or multiple classes per object [17], among other scenarios.

The wide literature on classification models (henceforth simply called "*classifiers*") offers a range of possible solutions and approaches for facing classification problems. Decision trees, k-nearest neighbors, Bayesian networks, Random Forests, Support Vector Machines or artificial neural networks stand among the most popular classifiers, each having its own advantages and drawbacks. The reader may refer to the studies conducted in [18] and [19] for further information about the performance of these traditional classifiers.

A. Decision trees

In the context of machine learning, a decision tree is a tree-like graph structure, where each node represents a test on an attribute. Each branch represents the outcome of the test and the leaf nodes represent the class label obtained after all decisions made through that branch. The paths from root to leaf represent classification rules. The goal in this scheme then is to represent the data while minimizing the complexity of the model. Several algorithms for constructing such optimized trees have been proposed. C4.5 [20] for example, derives from the well-known divide-and-conquer technique and have been widely used in several application fields, being one of the most popular machine learning algorithms. This scheme builds decision trees from a set of training data using the concept of information entropy. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set according to the normalized information gain (difference in entropy). It is important to note that this simple, yet efficient technique, is capable of handling missing values in the data sets and both numerical and categorical attributes.

B. K-nearest neighbor

The k-Nearest Neighbors algorithm (kNN) is a method used for classification and regression [21]. For the inference process, this technique determine the k closest training examples to the testing instance. The output in a classification context is the mode of the class values of the selected neighbors. In case of regression is the mean of the values of its k nearest neighbors. Therefore kNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The use of weights for the neighbors is a common technique and can be based on the distance to the instance in question. A commonly used distance metric for continuous variables is Euclidean distance, or Hamming distance for discrete features. The neighbors are chosen from historical data, but no learning algorithm is performed. A shortcoming of the kNN algorithm is that it is sensitive to the local structure of the data.

C. Bayesian networks

Bayesian networks [22] are directed acyclic graphs where nodes represent random variables following the Bayesian theory: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges in this model represent conditional dependencies; unconnected nodes represent variables that have no dependency relation to the other nodes. Each node computes its value using a probability function that outputs the probability (or probability distribution) of the node, from a particular set of values of the preceding nodes.

The inference process in Bayesian networks is based on Bayes' rule and it computes the posterior probability of each explanation. The learning process needs to determine the graph structure and the parameters of the model (the conditional probability distribution at each node). If the variables are discrete, this can be represented as a table, which lists the probability that the child node takes on each of its different values for each combination of values of its parents. However all relations in the structure have to be learned from data or can be designed in prior, having sufficient knowledge of the dependencies in the application domain. An alternative is to use Naïve Bayes classifiers [22]. A family of algorithms have been proposed for learning the structure and parameters of these models, all of them based in a common assumption: the value of a particular feature is independent of the value of any other feature given the class variable, thus avoiding the modeling of such dependencies. Despite their apparently oversimplistic design, Naïve Bayes classifiers work quite well on diverse application problems, although other more recent techniques like Random Forest or Support Vector Machine frequently outperform this classifier.

D. Random forest

Random Forest (RF) [23] is a well-known ensemble learning method for supervised classification or regression. This machine learning technique operates by building an ensemble of random decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Therefore a RF is a classifier consisting in a collection of tree structured classifiers which uses random selection in two moments. In a first step, the algorithm selects several (e.g. 500) bootstrap samples from the historical data. For each bootstrap selection k, the size of the selected data is roughly 2/3rd of the total training data (exactly 63.2%). Cases are selected randomly with replacement from the original data and observations in the original data set that do not occur in a bootstrap sample are called out-of-bag (OOB) observation. In a second step, a classification tree is trained using each bootstrap sample, but only a small number of randomly selected variables (commonly the square root of the number of variables) are used for partitioning the tree. The OOB error rate is computed for each tree, using the rest (36.8%) of historical data. The overall OOB error rate is then aggregated, observe that RF does not require a split sampling method to assess accuracy of the model. The final output of the model is the mode (or mean) of the predictions from each individual tree. Random Forest comes at the expense of a some loss of interpretability, but generally greatly boosts the performance of the final model,

becoming one of the most likely to be the best performing classifier in real-world classification problems [18] [19].

E. Support vector machine

Support Vector Machines (SVM) [24] are another powerful supervised learning technique that can be used for classification and regression analysis. The base idea of SVM training algorithm starts from a given set of training examples (support vectors), where each one is marked for belonging to one of two categories, as a non-probabilistic binary linear classifier. Then, the SVM can be seen as a representation of the examples as points in space, with the goal of separating examples in categories divided by a clear gap. More formally, SVM tries to find a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, such as a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (functional margin), minimizing the generalization error of the classifier. Unclassified instances then are mapped into the same space for classification.

The mapping of the original finite-dimensional space into a much higher-dimensional space, presumably makes the separation easier to identify in that space. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations of images of feature vectors. Then, SVM schemes are defined in terms of a kernel function $k(x, y)$ selected to suit the problem, ensuring that dot products may be computed in terms of the variables in the original space.

Several extensions of SVM such as support vector clustering [25] or transductive Support Vector Machines [26] are worthnoting. SVM remains as a competitive performing machine learning technique for supervised classification [18] [19].

F. Multi-layer perceptron

A multilayer perceptron (MLP) [27] is a feed-forward artificial neural network model that attempts to map a set of input data onto a set of corresponding outputs. An MLP can be described as a direct graph where nodes are called neurons and they are organized in three types of layers: input nodes, output nodes and hidden layers. Each layer is fully connected to the next one and each neuron (except for inputs) has a nonlinear activation function (e.g. hyperbolic tangent, logistic function). Each neuron updates its value taking into account the values of the connected neurons and the weights of these connections.

Several supervised learning algorithms have been proposed in literature for changing connection weights after each instance of historical data is presented, based on the amount of error in the output compared to the expected result. Backpropagation is perhaps the most extended one, consisting in a generalization of the least means squares algorithm in the linear perceptron. MLP have been widely used in literature for classification problems of diverse origin, however, other techniques like RF or SVM are reported to be very competitive [18] [19].

TABLE I
TRANSFORMING NOMINAL FEATURES TO BINARY VALUES

	country		usa	ghana	russia	australia
1	usa	$\Longrightarrow$	1	0	0	0
2	ghana	$\Longrightarrow$	0	1	0	0
3	russia	$\Longrightarrow$	0	0	1	0
4	australia	$\Longrightarrow$	0	0	0	1
5	ghana	$\Longrightarrow$	0	1	0	0

IV. Experimental Results

A. Feature Selection

Feature selection is a delicate and difficult issue in a data set consisting of over 2 million entries, each having over 3 million attributes, hence making pattern detection and feature correlation two computationally heavy tasks.

In the dataset, nominal features of an URL are coded as a set of binary attributes, each corresponding to one of the possible values, see Table I for an unrelated example. When spreading out such a categorical value across multiple binary attributes, none of the individual attributes contains full information about the feature, unless its value happens to be 1. This implies that simply identifying a set of binary attributes as related to the same nominal feature, is not sufficient for adequate feature selection. One has to be able to detect those attributes for whom a value of 1 is the most significant in classifying an URL.

Having over 3 million features, the calculation of interdependencies between those features is a computationally demanding task. Given the amount of binary attributes, only 67 out of 3 million have real-values, pattern detection does not guarantee a ready-to-use solution. Instead, we opted for different feature selection methods in which individual features that have the highest absolute Pearson correlation coefficient with the actual class of the URL.

Feature set A contains binary and real-value attributes. The first data set, Day0, is used to calculate the Pearson correlation of each attribute with the class of the URL. Attributes that show an absolute correlation coefficient that is higher than a arbitrary value of 0.2, are selected.

Feature set B contains only binary value attributes. The first data set, Day0, is used to calculate the Pearson correlation of each attribute with the class of the URL. To correct for the absence of the real-value numbers, a lower threshold value of 0.1 is used to select all attributes with a higher absolute Pearson coefficient.

Feature set C contains all and only real-value attributes, regardless of their correlation with the prediction class. These features are more likely than the binary to possess a non-zero value, thus having a better chance of containing descriptive information about the URL. Moreover, the real-value features can be retrieved without interference from a domain expert, nor is it required to do descriptive data analysis to detect patterns for feature selection.

As a final remark in this section, we would like to mention that independent classifications with each of the methods, for 5 different randomly chosen attributes yield very bad results. *All* predictors classified every URL as benign, which is true for the majority of the URLs, hereby achieving an overall classification accuracy of 66.5%.

B. Method Comparison

Given the large amount of data available, a non-stratified independent random sample with equal probability is taken from a set of row numbers between 1 and 20,000. For each of the 121 data sets, the URLs corresponding to the row numbers in the sample, would be added to the training set. This results in 2.5% of all URLs used for learning the models.

The test set consists of 121 different data sets, one for each day, containing all URLs that have not been used for training. All numbers from the experimental results have been achieved by calculating the prediction accuracy for each of the 121 different test sets.

In order to evaluate the models, we use three different metrics. *Accuracy*, Eq. 1, can be seen as the overall success rate of the method in terms of predictions. *Precision*, Eq. 2, is the ratio of positive predictions that are correctly classified. Low precision suggests the method is inclined to classify URLs as malicious, even when they are not. *Recall*, Eq. 3, is the ratio of actually positive cases that are also identified as such.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

The formula's used in Eq. 1, 2, and 3 use the following symbols:

- *TP* : number of *true positives*, malicious URLs that are classified as such
- *TN* : number of *true negatives*, benign URLs that are classified as such
- *FP* : number of *false positives*, benign URLs that are classified as malicious
- *FN* : number of *false negatives*, malicious URLs that are classified as benign

Precision and recall are included to counter the unbalanced data sets, containing more benign than malicious URLs.

As low recall rates indicate that a method is failing to identify malicious URLs, it could be argued that this measure is most important in a context where the cost of an undetected malicious URL outweighs the cost of rejecting a benign URL. When there is high recall and low precision, the method is classifying too many URLs as malicious. Ideally, both measures are as high as possible, but, more importantly, they should have roughly the same value, indicating a balanced and unbiased prediction.

TABLE II
AVERAGE ACCURACY PER CLASSIFICATION METHOD AND PER FEATURE SET

	A	B	C	
RF	98.26% (1)	96.91% (1)	97.91% (1)	97.69%
MLP	97.97% (2)	96.57% (4)	97.31% (2)	97.28%
C4.5	97.33% (5)	96.78% (2)	96.33% (3)	96.82%
kNN	97.54% (3)	95.23% (6)	95.98% (4)	96.25%
SVM	97.11% (6)	96.01% (5)	95.17% (5)	96.10%
C5.0	97.40% (4)	96.72% (3)	93.65% (7)	95.92%
NB	95.98% (7)	91.36% (7)	94.25% (6)	93.86%
	97.37%	95.65%	95.80%	

TABLE III
POST-HOC PROCEDURES : PAIRED COMPARISONS OF ACCURACY WITH REGARD TO ACCURACY OBTAINED BY RF

RF	Hochberg	Holm	Hommel
MLP	0.00104	0.00104	0.00104
C4.5	4.6E-14	4.6E-14	4.6E-14
kNN	<2E-16	<2E-16	<2E-16
SVM	<2E-16	<2E-16	<2E-16
C5.0	<2E-16	<2E-16	<2E-16
NB	<2E-16	<2E-16	<2E-16

The results for kNN are obtained with $k = 2$. Random Forest uses 500 trees. In SVM, the radial basis function is used, with $\gamma = \frac{1}{numberoffeatures}$. MLP has 10 neurons in the first hidden layer and 30 in the second.

Table II display the average accuracy rate per classification method and per feature set. The classification methods are sorted based on mean accuracy across the different feature sets. The table includes summaries per method and per feature set, as well as a ranking of the performance of each classification method per feature set.

A first observation is that all methods and all feature sets achieve reasonably high accuracy scores, with very small differences in between. On average, only Näive Bayes achieves an overall accuracy below 95%. In total, only three times an average accuracy below 95% has been observed. This is the case for Näive Bayes with feature sets B and C and C5.0 with feature set C.

The ranking of the methods is fairly similar across all feature sets. The lower ranking for kNN using feature sety B might be explained by the fact that a random, minor, noise factor had to be added to the set of binary features. This is needed to diminish the number of ties that would occur after all euclidean distances between URLs have been calculated.

The highest accuracy score has been obtained by RF, using feature set A. Even though the differences between the methods are generally small, pair-wise statistical tests using Hochberg, Holm and Hommel post-hoc procedures, see Table III seem to support the claim that, overall, RF significantly outperforms the other classification methods in terms of prediction *accuracy*.

Despite the fact that all *accuracy* scores ressemble eachother, the Bergmann post-hoc procedure in Table IV considers most differences significant at a level .01. The test does not lend proof to conclude that there is a significant difference

TABLE IV
BERGMANN POST-HOC PROCEDURE : ACCURACY ACROSS ALL FEATURE SETS

	RF	MLP	C4.5	kNN	SVM	C5.0
MLP	0.00E+00					
C4.5	0.00E+00	3.23E-05				
kNN	0.00E+00	0.00E+00	2.22E-10			
SVM	0.00E+00	0.00E+00	0.00E+00	4.03E-08		
C5.0	0.00E+00	0.00E+00	1.42E-14	1.56E-01	3.23E-05	
NB	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.51E-12	0.00E+00

TABLE V
BERGMANN POST-HOC PROCEDURE : ACCURACY FOR FEATURE SET A ONLY

	RF	MLP	kNN	c50	c45	SVM
MLP	1.12E-03					
kNN	0.00E+00	3.41E-08				
C5.0	0.00E+00	3.23E-13	8.71E-02			
C4.5	0.00E+00	0.00E+00	6.86E-03	1.81E-01		
SVM	0.00E+00	0.00E+00	2.41E-12	1.05E-07	7.15E-05	
NB	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.55E-13	2.77E-03

in accuracy between the results obtained from C5.0 and thos obtained using kNN. Nevertheless; these results suggest that, for the largest part, the ranking of the methods as depicted in the table can be considered significant, electing RF as the best classifier across all feature sets, followed by MLP and C4.5.

As the results suggest that feature set A has the best performance in terms of accuracy, Table V compares the results of Bergmann's post-hoc procedure using the accuracy for feature set A only. The methods are sorted based on highest mean accuracy obtained for that feature set, resulting in a slightly different sequence than can be seen in Table IV. Showing significance at level .01, the procedure again lends credibility to the ranking of most methods. There is no sufficient proof to conclude that theaccuracy obained by kNN is superior to the accuracy achieved via *C5.0*. In turn, *C5.0* does not significantly outperfrom *C4.5* at the .01 level. It should be noted that the difference in accuracy between kNN and *C4.5* is nevertheless to be considered significant.

Having RF as the highest ranked method, the ranking of the other classification methods is slightly different depending on the feature set. This ranking per feature set has been included between brackets next to each accuracy score in Table II. Mention can be made of C5.0 which has the poorest ranking for feature set C, but ends up in third place for the other two sets.

Feature set B, containing only binary parameters apparently offers little variation to the classification methods as all classifications end up with an accuracy score of around 96.5%, with the exception of Näive Bayes.

Overall, feature set A seems to obtain the highest performance results overall for every classification method. Feature sets B and C alternate as second ranked, depending on the method used, and end up on virtually the same overall average accuracy score. This claim is supported by the pair-wise tests, where no significant difference in *accuracy* can be found between feature set B and C, see Table VI.

TABLE VI
POST-HOC PROCEDURES FOR PAIRED COMPARISONS USING DIFFERENT FEATURE SETS

A	Hochberg	Holm	Hommel
B	<2E-16	<2E-16	<2E-16
C	<2E-16	<2E-16	<2E-16
B			
C	0.082	0.082	0.082

Fig. 2 depicts boxplots for the different performance measures per classification method and per feature set. The methods are sorted based on mean accuracy per method, across feature sets.

As with *accuracy*, *precision* and *recall* seem to be generally high, but display a much wider range across the data sets. Especially the *precision* measure seems to have an occasional outlier with a relatively low value compared to recall and accuracy. According to the definition, this suggests that occasionally, a high number of URLs may be incorrectly classified as malicious.

By definition, classifications with a *recall* measure that is relatively low compared to its *accuracy* and *precision*, have a tendency to classify URLs as benign, resulting in a relatively bigger number of malicious URLs to remain undetected. Fig. 2 suggests that Näive Bayes has a tendency towards low *recall* scores. In general, there do not seem to be big differences between *recall* and *precision* for feature set A and B. It should be noted though, that feature set B has a tendency to reveal slightly lower *recall*. This tendency is even more pronounced for feature set C, which is especially the case for Support Vector Machines, Näive Bayes, and C4.5.

V. CONCLUSION

The numerical simulations and the corresponding statistical tests strongly suggest that the differences between the methods are significant. Therefore, the ranking of the methods as shown in Table II can be accepted as a correct and significant ranking in terms of prediction accuracy. Although all methods achieve fairly high prediction accuracy, *Random Forest* appears to be the most appropriate classification algorithm for this problem, followed by *MLP*. Moreover, *Random Forest* achieves high scores for both *precision* and *recall*, which not only indicates well-balanced and unbiased prediction results, but also provides credibility to the method's ability to maximize the detection of malicious URLs within reasonable boundaries.

Regarding feature selection, the feature set containing the features that have the highest absolute Pearson coefficient with the prediction class, outperforms the two other sets at a significant level. No significant differences in accuracy can be found when comparing feature set B, highly correlated binary features, with feature set C, only the real-value features.

Despite the significance of the difference in accuracy it should be noted that the differences between the feature sets remain relatively small. Depending on the level of accuracy that is considered to be acceptable, one might pick either

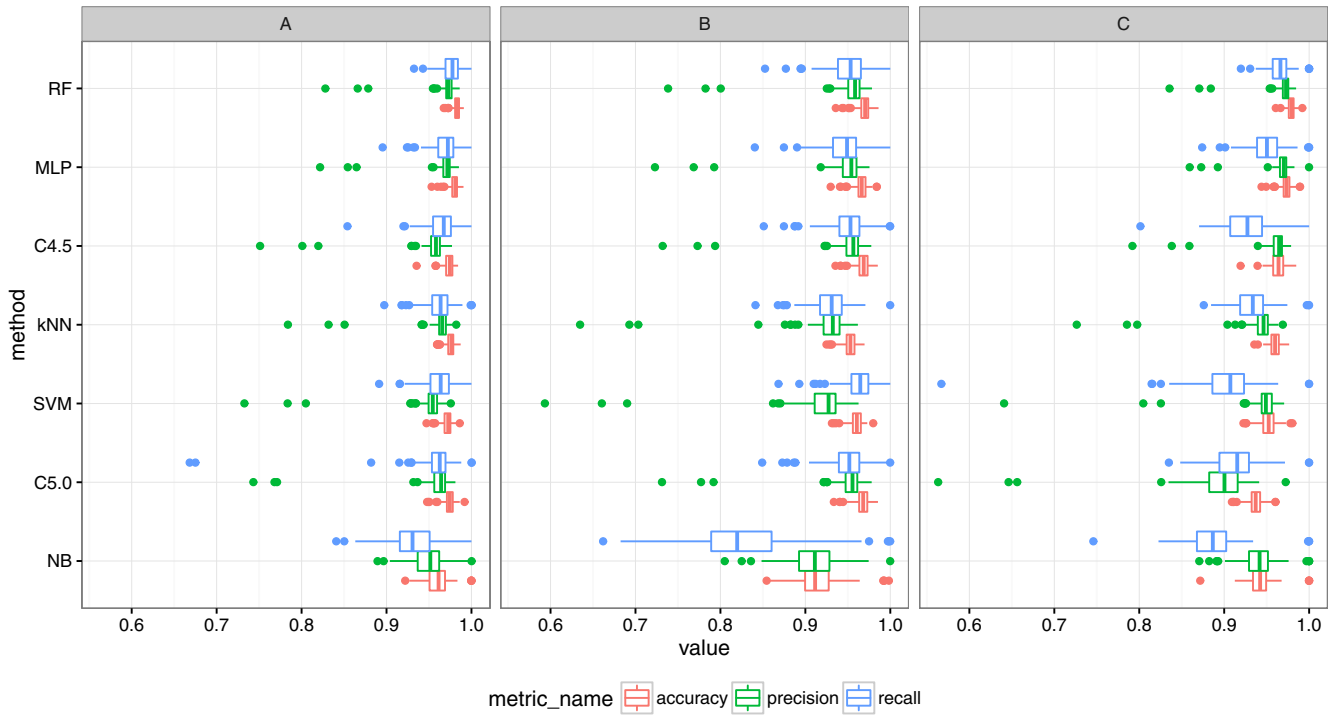

Fig. 2. Accuracy, precision and recall per classification method and per feature set

feature set B or feature set C as proxies for set A. The benefit of using feature set C over the other sets would be that it does not require a preliminary exploration analysis to determine the correlations, relying solely on the descriptive power of a real-value feature over a binary feature.

It is worth highlighting that these accuracy rates have been achieved without requiring either advanced feature selection techniques or the involvement of a domain expert. Nevertheless, the methods seem to achieve competitive results. We believe that classification problems for over 2 million entries with over 3 million features is as much about feature selection as it is about the identification of an adequate classifier. The fact that the feature set is sparsely populated and contains mainly binary attributes makes feature selection a more challenging task. Categorical features are spread out over multiple binary attributes, causing none of the attributes to contain full knowledge about the feature. Given that numerical features are not coded this way and, as a consequence, do not suffer from the aforementioned drawbacks, they are interesting candidates to use during training. The results of this paper suggest that the classification methods achieve competitive prediction accuracy rates for URL classification when only the numerical features are used for training.

REFERENCES

[1] Symantec, "2016 Internet security threat report," https://www.symantec.com/security-center/threat-report, 2016, [Online; accessed 11-Aug-2016].

[2] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–5.

[3] P. de las Cuevas, Z. Chelly, A. Mora, J. Merelo, and A. Esparcia-Alcázar, "An improved decision system for URL accesses based on a rough feature selection technique," in *Recent Advances in Computational Intelligence in Defense and Security*. Springer, 2016, pp. 139–167.

[4] A. Mora, P. De las Cuevas, and J. Merelo, "Going a step beyond the black and white lists for URL accesses in the enterprise by means of categorical classifiers," *ECTA*, pp. 125–134, 2014.

[5] M.-Y. Kan and H. O. N. Thi, "Fast webpage classification using url features," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 325–326.

[6] E. Baykan, M. Henzinger, L. Marian, and I. Weber, "Purely URL-based topic classification," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 1109–1110.

[7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1245–1254.

[8] ——, "Learning to detect malicious URLs," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 30, 2011.

[9] P. Zhao and S. C. Hoi, "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*. ACM, 2013, pp. 919–927.

[10] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying Suspicious URLs: An Application of Large-scale Online Learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 681–688.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. John Wiley & Sons, 2012.

[12] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: a review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[13] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current

trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.
[14] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 845–869, 2014.
[15] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, and T. S. Huang, "Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 12, pp. 1553–1566, 2004.
[16] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information Systems*, vol. 42, no. 2, pp. 245–284, 2015.
[17] G. Tsoumakas and I. Katakis, "Multi-label classification: an overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.
[18] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?" *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014.
[19] M. Wainberg, B. Alipanahi, and B. J. Frey, "Are random forests truly the best classifiers?" *Journal of Machine Learning Research*, vol. 17, no. 110, pp. 1–5, 2016.
[20] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kauffman Publishers, 1993.
[21] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[22] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.
[23] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[24] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *Intelligent Systems and their Applications, IEEE*, vol. 13, no. 4, pp. 18–28, 1998.
[25] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of machine learning research*, vol. 2, no. Dec, pp. 125–137, 2001.
[26] T. Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, vol. 99, 1999, pp. 200–209.
[27] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," DTIC Document, Tech. Rep., 1961.