Using supervised machine learning algorithms to detect suspicious URLs in online social networks

Mohammed Al-Janabi
Keele University
m.f.al-janabi@keele.ac.uk.com

Ed de Quincey
Keele University
e.de.quincey@keele.ac.uk

Peter Andras
Keele University
p.andras@keele.ac.uk

***Abstract*– The increasing volume of malicious content in social networks requires automated methods to detect and eliminate such content. This paper describes a supervised machine learning classification model that has been built to detect the distribution of malicious content in online social networks (ONSs). Multisource features have been used to detect social network posts that contain malicious Uniform Resource Locators (URLs). These URLs could direct users to websites that contain malicious content, drive-by download attacks, phishing, spam, and scams. For the data collection stage, the Twitter streaming application programming interface (API) was used and VirusTotal was used for labelling the dataset. A random forest classification model was used with a combination of features derived from a range of sources. The random forest model without any tuning and feature selection produced a recall value of 0.89. After further investigation and applying parameter tuning and feature selection methods, however, we were able to improve the classifier performance to 0.92 in recall.**

Keywords-– Twitter; malicious URLs; phishing; spam; random forest; spam detection

I. INTRODUCTION

The main challenges for social network security administrators are not only protecting the social network management system and database, but also protecting OSN users from being exposed to malicious content that is spread over those social networks. 60% of social network users have received or been exposed to malicious content [1] such as spam, scams, and drive-by downloads. A number of OSNs are now developing malicious content detection systems for such attacks e.g. the Facebook Immune System detects suspicious activities such as like-jacking, social bots, and fake content [2].

Social network detection systems vary in their robustness yet are quite similar in their detection techniques. Examples of popular detection techniques used are blacklists and machine learning-based classifiers. Blacklists, which are considered one of the traditional techniques used in this field, are databases that collect records of previously detected attacks that have been reported either by users or by security communities. Usually, blacklists are considered as the first defensive line that OSNs use for protection. For example, Twitter uses Google Safebrowsing[1] in particular as a blacklist service [3]. In addition to Safebrowsing, the other well-known blacklist services include PhishTank[2], URIBL and SURBL. The blacklist technique offers real-time detection with a low false positive rate [4]; however, blacklist techniques cannot detect URLs that have not been included explicitly in these blacklists. Spammers therefore can exploit the time gap between spreading unknown URLs and the time required for the blacklists to be updated [5]. This is achieved by creating URLs with no historical profile via URL-shortening[6] and low-cost domain registration and hosting services [7]. The consequence to users is that they are exposed to links to malicious content in real-time, with spam campaigns achieving 80% of their spreading target within the first 24 hours [5].

Due to this weakness of blacklists and the sheer volume and complexity of data exchanged in OSNs, automated procedures for identifying malicious content are essential [8]. The majority of studies [9][10][11] that have attempted to mitigate malicious content in social networks have built efficient machine learning classifiers that can classify unseen malicious URLs into spam or benign with the minimum amount of human intervention. However, training these classifiers with an appropriate dataset is non-trivial [12][13], with an adequate training dataset needed to build an accurate supervised machine learning model. Blacklists can be used to provide a large set of labelled records of previously identified malicious URLs, which will be used to train the classifiers.

Recent studies have compared several machine learning algorithms to select the best algorithm for their collected dataset. They used mainly supervised machine learning algorithms for spam classification, such as Naïve Bayes (NB), k-Nearest Neighbors (k-NN), Random Forest (RF), and Logistic Regression (LR).

Most studies [11][14] conclude that RF gives a higher classification performance than the other supervised machine learning algorithms. However, in these studies specific details related to which algorithm parameters or feature selection methods were used are often not provided. This absence of information means that it is not possible to build an equivalent model to reproduce the results.

In this study, the RF classifier has been used to investigate how its performance can be optimized further by using feature selection and parameter tuning. For feature selection, we aimed to determine the most effective features that were derived from multiple sources, as previous studies [15] suggested that classifiers that are based on a multisource of features can give better

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

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

classification performance. For parameter tuning we aimed to determine the best setting for the number of trees, the size of leaf nodes and the depth of the trees in the RF classifier. However the purpose of this paper is not to compare RF with other methods, but to demonstrate a way of systematically analysing the application of RF to spam twitter detection and to highlight the importance of appropriate statistical analysis in the process of setting the RF parameters.

The rest of the paper is organized as follows. Section II, reviews the related work. In Section III, we describe the data that we used. In Section IV, we discuss model enhancement methods. Section V contains the discussion of the results, our conclusions and indications of the planned future work.

II. Related Work

The majority of studies in this area aim to find the most predictive features that they can acquire and the best algorithm to develop a classifier model [16]. Researchers in this field focus mainly on finding novel features with high discriminative power in addition to coming up with the most accurate machine learning model [17]. Finding high discriminative features in the area of Internet security and social networks is quite a challenge due to the variation in attacks and techniques used by spammers.

Due to the inventiveness of spammers detection systems are bypassed after some time and the set of features used for spam detection has to be regularly revised [18][19]. Similar to how security researchers study the attacks, spammers and hackers investigate detection systems; therefore, they can change user properties, content or the distribution mechanism to bypass certain restriction or detection rules [20]. For example, a study of detecting spam on Twitter [21] recommended that the number of followers is one of the highest discriminative power features. The feature's discriminative power has been increasingly weakened though by spammers making their accounts more popular. They do this by conducting spam campaigns that make their "fake" accounts connect with other fake accounts, increasing the follower and following numbers [22].

TABLE I. COMMON FEATURES USED IN LITERATURE

Feature	***Features Description***	***Ref***
Account age	Number of days since account created	[21][11]
No of followers	Number of accounts connected to this account	[21][11][25]
No following	Number of accounts connected to this account	[21][11][25]
No of user favorites	Number of favorite tweets	[21]
No of tweets	Number of posted tweets	[21][11][25]

The majority of previous studies [23][24][21] begin by collecting data using the Twitter streaming API[3]. Multiple features are then extracted and different feature sets utilized. Some studies have common features, and often these are features that do not require a preprocessing stage as they are provided by Twitter in a numerical format e.g. tweets number and number of followers. These can then be deployed directly into a machine learning algorithm. Table I shows examples of common features that have been used in previous studies [21][11].

Chen et al. [21] used 12 Twitter-only lightweight features. Lightweight features are those that do not require complicated preprocessing operations or significant resource to be extracted e.g. age of account and number of followers. They compared six different classification algorithms – BayesNet, Naive Bayes, Decision Tree (C4.5), RF, k-NN, and support vector machines (SVMs) – which are ordered here in terms of their F-measure scores. RF had the highest performance at 93.6% in the F-measure using an evenly distributed dataset. In the case of highly imbalanced datasets, however, such as a 1:19 ratio for spam/non-spam, the performance dropped to 56.6%. Classifiers that have been trained on imbalanced data are more likely to be subject to bias by majority class results.

Blum et al. [24] focused on detecting fraudsters' methods to get users to click on links that are designed to be similar to the websites they trust or use. They used only URL lexical features. The features were presented as a bag of words after splitting URLs into three text strings: protocol, domain, and path. The advantages of this detection method are the lightweight data acquisition and the speed of implementation.

Burnap et al. [23] used an entirely different method to detect malicious URLs. They deployed a high-interaction honey-net[4] to collect system state changes, such as the sending/receiving packets and CPU usage. The training dataset contained 2,000 examples with a 1:1 ratio for spam/non-spam. Ten attributes were used to build a classifier that reflected system status changes after opening the tweet's URL. Burnap et al. investigated the shortest time required to give a preliminary warning of the existence of malicious content in a particular URL. The best result was reported for Multilayer Perceptron (MLP) using features acquired after 210 seconds (0.723 in the F-measure metric). The features used by Burnap et al. require complex data analysis; however, they make it difficult for spammer sites to disguise their true nature.

Although the recent literature has compared several algorithms, there is a lack of information about important stages in building a machine learning model. In particular, little information is provided about how feature selection methods are managed and how parameter tuning is conducted. We address this issue in Section IV.

III. Study Methodology

This section describes in detail the main stages of this study, starting with the data collection and labelling of the dataset, followed by a brief comparison of the most common algorithms used in related studies.

A. Data collection and labelling

This section explains how the data collection process was conducted and how we built the ground truth dataset of malicious and benign tweets. The data collection involved three steps, as shown in Figure I: (i) collecting tweets that have URLs, (ii) crawling each URL using headless Selenium browsers, and (iii) labelling the tweet's URLs as malicious or benign.

[3] https://dev.twitter.com/streaming/overview

[4] https://www.honeynet.org/

For the data collection stage, we used Twitter's public streaming API for tweet collection that gives access to 1% of the total stream [23]. A Python script was written to connect to the Twitter API and retrieve tweets that contained at least one URL. The tweets were then stored in a MongoDB[5] database. Two million tweets were collected between June 15, 2016, and August 14, 2016, at random times of the day. Figure I shows the collection stage of tweets, starting by connecting to the Twitter stream API, then applying the filter rule to limit to tweets with URLs.

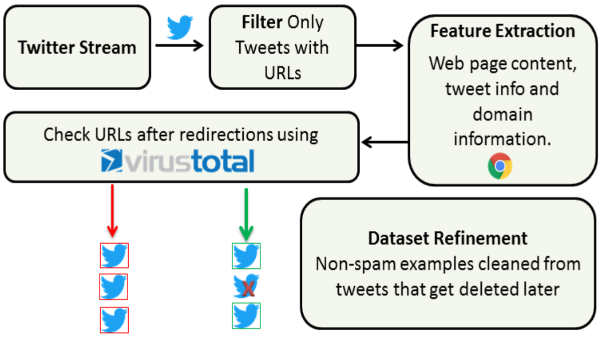
Twitter Stream
Filter Only Tweets with URLs
Feature Extraction
Web page content, tweet info and domain information.
Check URLs after redirections using
virustotal
Dataset Refinement
Non-spam examples cleaned from tweets that get deleted later

Fig. 1. Data collection and labelling stage

When building a supervised machine learning model, a labelled dataset is needed. In this study, this meant labelling each tweet in the dataset as either “spam” or as a “normal” tweet. To build this ground truth dataset, the tweets’ URLs were checked using VirusTotal[6]. VirusTotal is a multisource online database that is used to check whether a particular URL exists in any blacklist database. VirusTotal provides an API for retrieving information about URLs using up to 50 reputable online blacklists, such as Google Safebrowsing (Google), BitDefender, Dr.Web Link Scanner, Kaspersky URL Advisor (Kaspersky), PhishTank (OpenDNS), Spam404, and Trend Micro Site Safety Center (Trend Micro[7]).

As an additional refinement stage, each benign example in our dataset, i.e. a tweet containing a URL that was not blacklisted, was checked to determine whether they had been deleted by Twitter as this may indicate tweets that they contained malicious URLs that are not on a blacklist. According to Twitter’s deletion rules[8], there are three major reasons to eliminate a user’s[9] tweet: breaking copyrights, abusive tweeting activity, and that it is spam from Twitter’s perspective. In order to check if a tweet had been deleted, the twitter streaming API was used to retrieve a specific tweet (using its ID). If nothing was retrieved via the API then we consider it as deleted. This procedure was conducted several times during data collection, the last checking was done in December 2016.

We removed the deleted tweets from our dataset to eliminate from the dataset any tweet that could be deleted because it is spam or considered to be malicious. As a result, we were able to create 150,000 examples of ground truth data, divided into 120,000 non-spam examples and 30,000 malicious examples, which ranged from malware, phishing, scam pages, and overloaded ads to low-quality web pages. We also validated our dataset periodically using the two methods mentioned above, as some spam URLs required longer time to be blacklisted or deleted by Twitter.

In addition to the collected tweets, we stored the content of each webpage that the tweet’s URL led to. For this purpose, we used headless browsers to open each link and catch page loading behaviors such as the redirection hubs and get the final landed webpage. We used a high speed connected machine and high processing speed to retrieve all URLs in our dataset. We set up the headless browsers on a Core i7 32GB RAM machine to visit each tweet's URL to collect additional source data. We extracted web page content and URL redirection behavior. To determine the domain age, we used the WHOIS info API to acquire information about the domain's registration date.

B. *Feature extraction and engineering*

In the context of machine learning, features are used to provide discriminative power in the classification process. The preliminary feature set used was based on the literature [11], and also contained features selected by studying cases of real spamming content distributed over OSNs. A total of 36 features were extracted from the Twitter stream API, domain information, and web page content. Table II shows features that were extracted directly from the tweet's metadata, provided by the Twitter streaming API. These can be considered as lightweight features as they do not need further pre-processing.

TABLE II. TWITTER FEATURES READY TO BE USED

Feature	***Feature Source***
User name signs	User Info
Default profile image	User Info
Have media	Tweet Content
User listed count	User Info
User followers count	User Info
User friends count	User Info
User favorites count	User Info
User name length	User Info
Account age	User Info
Is user geo-enabled?	User Info
User statuses count	User Info
User name digits (number)	User Info

Models that rely entirely on Twitter metadata features though [26][27] could be subverted by spammers using techniques such as having fake followers [28], that make their spam user accounts look more legitimate. Therefore, to make this more difficult for spammers to subvert, a more powerful feature set is required. Recent studies [23], [29] recommend using features that are derived from several sources, such as domain WHOIS info and web page content. These features, shown in Table III though require resource intensive preprocessing (see

[5] https://www.mongodb.com/
[6] https://www.virustotal.com/
[7] https://global.sitesafety.trendmicro.com/
[8] https://support.twitter.com/articles/18311
[9] There is also the chance that the user themselves deletes the tweet.

previous section) to be extracted and converted for use in machine learning algorithms. Table VII shows the full list of features that were used for building the machine learning models used in this study.

TABLE III. FEATURES THAT REQUIRED EXTRACTION AND PREPROCESSING

Feature	***Feature Source***
Domain age	Domain WHOIS Info
Is it secured https?	User Info
Link length	Tweet Content
Link letters (number)	URL Info (after redirection)
Number of dots in link	URL Info (after redirection)
Number of link signs	URL Info (after redirection)
Number of digits in link	URL Info (after redirection)
User name length	User Info
Number of input forms	Web Page Content
Number of external links	Web Page Content
Number of ad blocked[10] links	Web Page Content
Number of webpage links	Web Page Content

C. *Model selection*

To explore the best-performance machine learning algorithms for the classification of spam and non-spam URLs associated with tweets, the top four common algorithms reported in the literature review (see section 2) were used. These algorithms are:

LR classifier: a probabilistic classifier, typically working on binary classification problems

RF: one of the ensemble classification techniques. RF builds many decision trees that are used to classify new data by the majority vote. RF shows good generalization due to the random sampling and random selection of features.

NB: one of the commonly used learning algorithms. The NB classifier is a probabilistic model based on the Bayes rule. 'Naïve' refers to the assumption of conditional independence among features.

k-NN: a supervised learning method used in classification problems. k-NN maps the training input feature vectors X = {x_1, x_2, ..,x_n} in n-dimensional space, then classifies new data based on the majority class for the k neighbors. k refers to the number of training samples closest to the point of entry.

All the algorithms above were implemented in this study by using Scikit-learn[11], which is an open source machine learning library in Python.

We trained and tested four classifiers using the same set of 36 features and the same training and testing datasets described in Section III.B. The ground truth dataset was randomly divided into a 75% training and 25% testing set. The four classifiers (RF, LR, k-NN, and NB) were trained and tested. We used the Scikit-learn default parameter values for all four algorithms. To assess performance, evaluation metrics were used that are known to be less sensitive to dataset imbalance [30]:

TABLE IV. RANDOM FOREST MAIN PARAMETERS

Data Set		Classifier decision	
		SPAM	Not SPAM
True	Spam	True Positive	False Negative
	Not SPAM	False Positive	True Negative

- Area under the curve (AUC) represents the classifier's ability to detect classes. If the AUC is 1, that means that the classifier perfectly detects class labels whereas 0.5 is equal to random selection. AUC has been found to be insensitive to an imbalanced [31] dataset.

- Precision is the ratio of true level of positive or negative detection of the classifier to overall test samples.

$$TP / (TP + FP)$$

- Recall is the ratio of correct true positive classifier decisions to the all true positive examples in the test set.

$$TP / (TP + FN)$$

- F-measure (F1) represents the previous metrics precision and recall combined as follows.

$$2 * (precision * recall) / (precision + recall)$$

Table V presents the overall performance of all implemented classifiers using the four evaluation metrics that are showed.

TABLE V. OVERALL PERFORMANCE (AVERAGE OF 10 EXPERIMENTS) USING ONE CLASSIFIER FOR ALL ATTRIBUTES

Model	AUC	F1	Precision	Recall
RF	**0.92**	**0.92**	**0.96**	**0.89**
LR	0.67	0.63	0.67	0.60
NB	0.58	0.62	0.51	0.78
k-NN	0.80	0.78	0.79	0.76

The results shown above confirm that RF had the best performance, which is aligned with the results reported in most of the literature [11][14]. Although one of RF's main advantages is that it does not require a fine-tuning process for its parameters [32], choosing the optimized values of parameters could protect the model from falling into overfitting.

The preliminary results show that the RF classifier with Scikit-learn default parameters (10 trees, undefined max depth and leaf size) reached 89% in the recall performance metric. The next section describes two approaches to enhance the classifier's performance.

IV. MODEL PARAMETER TUNING AND PERFORMANCE ENHANCEMENT

One of the main stages in building a machine learning model is model enhancement, i.e. changing the model's structure or parameters with the aim to improve its performance. Here we use two approaches to improve the model's performance and

[10] https://github.com/adblockplus/python-abp

[11] http://scikit-learn.org/stable/

decrease its complexity. First we tuned the RF model's parameters and second we trialed several feature evaluation methods to determine the feature set that gives the most accurate results.

A. Model enhancement by parameter tuning

Although RF does not require high effort fine-tuning, setting proper RF parameters prevents overfitting and enhances the detection performance. The RF main parameters [33][34] that we considered are tree number, max depth and leaf size (stopping criteria) (as shown in Table VI).

TABLE VI. RANDOM FOREST MAIN PARAMETERS

Parameter	***Description***
Tree number	Number of trees in building the RF classifier
Max depth	The maximum depth that the tree can grow
Leaf size	The minimum number of leaves a branch can have

To find the best parameter values for our model, we used the Scikit-learn grid search method. Using this approach, the parameters could be varied based on a range of pre-specified values. All options for all parameters cannot be considered, especially those that can be infinite, such as tree number and max depth.

In the context of the spam content classification task, the classifier performs very well if the number of trees is sufficiently large, the max tree depth is sufficiently high, and the minimum leaf size is sufficiently low. The results show that the number of trees has a relatively small impact, and beyond nine trees there is no significant change in the performance. The minimum leaf size has a greater effect, especially for classifiers with high max tree depth, for which even small changes in the minimum size of the leaf have a significant impact on the performance. Finally, the max tree depth has a significant effect on the performance for low values of this parameter, and the effect diminishes below significance for depth values above 16 or 24 for small and large minimum leaf size, respectively (Figure 2).

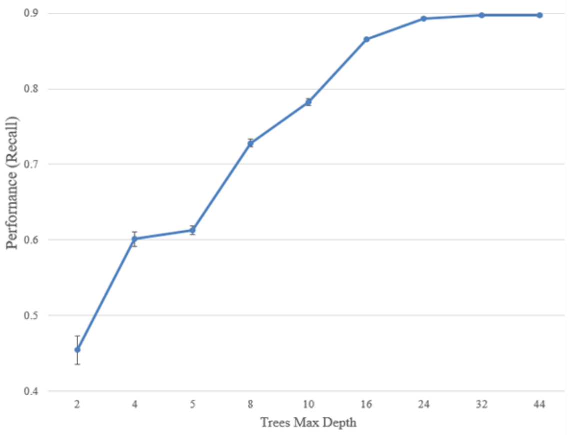

Fig. 2. Random forest performance using 19 trees and leaf size equal to 10 with varying tree depth

This implies that the number of trees and max tree depth should be set to moderate values to achieve good performance without an excessive computational burden. A too small minimum leaf size combined with an excessively large max tree depth is likely to lead to overfitting (note that the overfitting is because of the trees and not because of the forest arrangement of the trees [35], [36]). Therefore, controlling the minimum leaf size is important, and again it should be set to a moderate value to prevent overfitting and excessive unnecessary computation.

We determined that our optimal model parameter was 19 trees, a max depth of 24, and ten as the max leaf size. These parameter values will be used later in the process in feature selection. It is worth mentioning that all results are the average of 20 times trials using ten stratification fold validation.

B. Model enhancement by feature selection

The feature selection procedure was conducted in two stages. In the first stage, each feature in our original feature set was evaluated by applying a number of available feature selection metrics. Secondly, the top k features were chosen which should have the highest discriminative power [37]. Evaluating features, which is also known as the feature importance score, is an essential process to understand the dataset we rely on to build models. Moreover, it enables the researcher to distinguish between good features and irrelevant features. Eliminating redundant and noisy features could cause performance improvement [38]. There are several existing methods to perform feature selection, such as the wrapper and filter methods [39].

The wrapper method concept is based on model performance, and every chosen subset of features is used to build a classifier and evaluate its performance until the optimal subset is found with the lowest error rate. For a high dimensional dataset, the wrapper method could be a costly and time-consuming method; however, the wrapper method is one of the highly efficient methods as its feature evaluation relies directly on the classifier performance. It has been shown [40] that the wrapper method achieves higher classification accuracy than the filter method. Despite the high computational resources required, the wrapper method is recommended to be applied for such classification problems.

The mean decrease accuracy (MDA) [41] is ranking features based on the decrease in performance value after removing features one at a time. Essential features should show a negative impact when they are removed. Conversely, useless features should have no significant impact when removed. However, as mentioned earlier, some features, acting as noise, could have a negative impact on the model. Removing such features might improve the performance of the model.

The filter method uses importance measurement methods to assess the information content of features and possibly their correlation with the target classification. Unlike the wrapper method, the filter method does not rely on classifier performance to rank features' importance, making its application much faster Table VI, shows the features' importance ranking based on three methods, information gain, Gini Index and mean decrease accuracy (MDA).

The features' importance is varied; each method ranks features' importance somewhat differently, although there is general agreement on the top and bottom, which are the best and worst features. In this stage, we wanted to select the top k features that give the best classification performance. To conduct feature selection, we started by eliminating the lowest ranked features from the three ranking lists that were produced by three different evaluating techniques. Therefore, at each number of features, we built an RF classifier based on the new feature set, then compared it to the original performance we achieved by using the feature set with all the original 36 features. Our stopping criterion was whenever we got performance that was statistically less than the first classifier performance we built using all features, which was 0.89 in the recall. Figure III shows the performance of classifiers against the number of features used. Each time we evaluated the RF classification model, we used cross-validation to evaluate classifiers based on the recall metric.

TABLE VII. RANKING OF FEATURES BASED ON INFORMATION GAIN, GINI INDEX, AND MEAN DECREASE AVERAGE

#	*Feature*	*Info. Gain*	*Gini Index*	*MDA*
1	Domain age	1	2	1
2	Number of digits in link	2	1	2
3	Number of external links	7	6	5
4	Ratio of age to number of tweets	3	3	8
5	Link letters (number)	4	5	11
6	Number of links (webpage)	8	7	7
7	Number of images (webpage)	11	9	4
8	Number of dots in link	13	13	6
9	Ratio of words to external links	6	10	12
10	Number of input forms	12	12	3
11	Number of words (webpage)	10	11	10
12	Link length	5	8	13
13	User statuses count	9	4	15
14	Number of link signs (webpage)	14	14	14
15	Number of ad blocked links	20	18	9
16	User friends count	15	15	16
17	Account age	17	16	17
18	User followers count	16	17	22
19	User favorites count	18	19	18
20	Number of hashtags (tweet)	21	21	20
21	User listed count	19	20	24
22	Does link contain 'www'[12]?	25	24	19
23	Does webpage have password input?	27	25	21
24	Link letters (number)	22	23	27
25	Does tweet have media?	28	27	23
26	Number of mentions (tweet)	23	22	26
27	User name length	24	26	30
28	Is https protocol used in URL?	26	28	25
29	User name digits (number)	29	29	31
30	Is tweet is a reply tweet?	33	31	28
31	Number of URLs (tweet)	32	33	29
32	User name signs	30	30	33
33	Is user geo-enabled?	31	32	32
34	Default profile image	34	34	34
35	Is user account verified?	35	35	35
36	Is user account protected?	36	36	36

In Figure 3, the horizontal axis represents the number of top features used to build the classifier, and the vertical axis represents the performance in recall. To assess the impact of features on the classification performance we removed features from the three ranking lists, starting from the original 36 features. The performance of the classifier that was built using the original

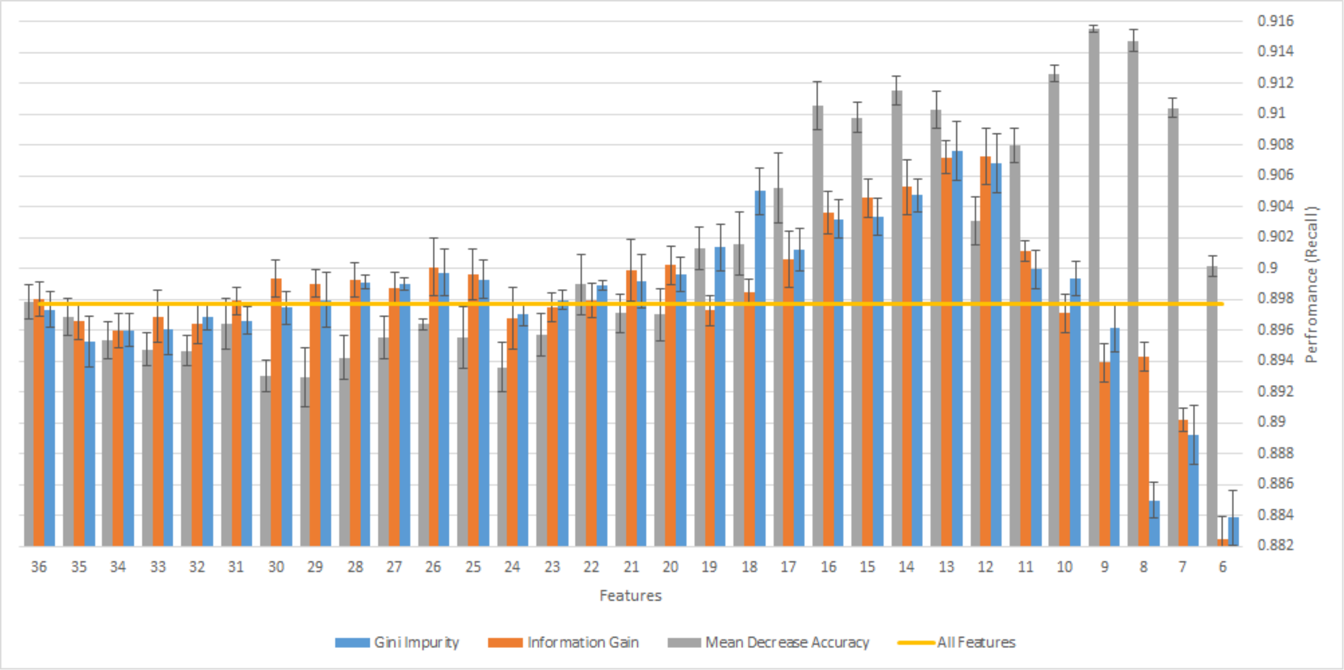

Fig. 3. Random forest classification performance based on the selected features.

[12] http://www.yes-www.org/why-use-www/

feature set (with 36 features) is shown as the horizontal line fixed with the performance at 0.897 in recall in Figure 3. This was used as a benchmark performance to assess the extent of improvement or degradation in classification performance cause by elimination of features. Figure 3 does not show classifiers' performance for less than six features, as the performance drops very much for further reduction of the number of top features. The classifier performance is improved as we removed lower ranking features.

The filter methods reached their peak performance (0.908) for 13 features for the Gini impurity features ranking list and for 12 features (0.907) for the information gain features ranking list. On the other hand, the MDA-based elimination of features reached its best classification performance (0.916) for 9 features. All methods improved the classifier performance and reduced the feature set to less than half.

V. Discussion, Conclusion and future work

In this paper, we aimed to find the highest performance model using the smallest number of features and the smallest structural parameters (tree number, max tree depth and maximum leaf size) in order to find the least complex but high performing classifier. First we determined the required minimal structural parameter values using all features and then we reduced the feature set to the minimally required set. The best feature set reduction was achieved using the computationally costly MDA wrapper method, but relatively close performance (although statistically significantly lower) and feature set reduction was achieved by using the two chosen filtering methods as well.

In the social networks, the spam detection task requires a fast response to the new emerging techniques and tricks that hackers use. Features in this field need to be re-ranked and evaluated periodically since machine learning models need to be built on reliable and validated feature sets to achieve high-quality performance. Features that used to be highly discriminative can become less effective if spammers change their methods or content. Our work provides a practical example of how to employ parameter tuning and feature selection methods to develop a low complexity and efficient machine learning classification tool for spam filtering in social media context.

We believe that it is important to tune the parameters of such spam classification tools and to optimize the feature set that they use in order to achieve reliable good classification performance. It is also important to report the parameter values and the details of the feature set optimization method that is applied in order to guarantee the reproducibility of the results reported in the paper.

For our future work, we aim to conduct a systematic analysis to evaluate the feature selection procedure and performance based on the wrapper method using RF and XGBoost[13], which is a new library of gradient-boosting trees. We also aim to compare other methods such as Random forest, XGBoost, Support Vector Machine and Deep Neural Network (DNN) classifiers.

The source code of our spam classification tool and the data set that we used are available from the authors on request.

Acknowledgment

We would like to thank the Iraqi Ministry of Higher Education and Research for the PhD scholarship provided for Mohammed Al-Janabi.

References

[1] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 973–993, 2014.

[2] T. Stein, E. Chen, and K. Mangla, "Facebook immune system," *Proc. 4th Work. Soc. Netw. Syst.*, vol. m, no. 5, pp. 1–8, 2011.

[3] K. Thomas and D. M. Nicol, "The Koobface botnet and the rise of social malware?," *Proc. 5th IEEE Int. Conf. Malicious Unwanted Software, Malware 2010*, pp. 63–70, 2010.

[4] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An Empirical Analysis of Phishing Blacklists," 2009.

[5] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: The Underground on 140 Characters or Less," in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*, 2010, p. 27.

[6] F. Klien and M. Strohmaier, "Short Links Under Attack: Geographical Analysis of Spam in a URL Shortener Network," in *Proceedings of the 23rd ACM conference on Hypertext and social media - HT '12*, 2012, p. 83.

[7] P. Ponce-Cruz and F. D. Ramírez-Figueroa, "Intelligent control systems with LabVIEW???," *Intell. Control Syst. with LabVIEW???*, pp. 1–216, 2010.

[8] J. Tang, Y. Chang, and H. Liu, "Mining Social Media with Social Theories: A Survey," *SIGKDD Explor. Newsl*, vol. 15, no. Iid, pp. 20–29, 2014.

[9] I. Kayes and A. Iamnitchi, "A Survey on Privacy and Security in Online Social Networks," *ACM Comput. Surv.*, no. 1, pp. 323–325, Jan. 2015.

[10] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to Spam filtering," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10206–10222, Sep. 2009.

[11] A. Aggarwal, A. Rajadesingan, and P. Kumaraguru, "PhishAri: Automatic realtime phishing detection on twitter," *eCrime Res. Summit, eCrime*, pp. 1–12, 2012.

[12] C. Yang, J. Zhang, and G. Gu, "A taste of tweets: reverse engineering Twitter spammers," *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, pp. 86–95, 2014.

[13] H. Li, Z. Chen, A. Mukherjee, B. Liu, and J. Shao, "Analyzing and Detecting Opinion Spam on a Large-scale Dataset via Temporal and Spatial Patterns," *Proc. Ninth Int. AAAI Conf. Web Soc. Media*, no. MAY, pp. 634–637, 2015.

[14] C. Yang, R. C. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving twitter spammers," *IEEE Trans.*

[13] https://xgboost.readthedocs.io

Inf. Forensics Secur., vol. 8, no. 8, pp. 1280–1293, 2013.

[15] Q. Huang, V. K. Singh, and P. K. Atrey, "Cyber Bullying Detection Using Social and Textual Analysis," in *Proceedings of the 3rd International Workshop on Socially-Aware Multimedia - SAM '14*, 2014, pp. 3–6.

[16] C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, "Statistical Features-Based Real-Time Detection of Drifted Twitter Spam," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 914–925, 2017.

[17] G. Canfora and C. A. Visaggio, "A set of features to detect web security threats," *J. Comput. Virol. Hacking Tech.*, pp. 1–19, 2016.

[18] S. Liu, Y. Wang, C. Chen, and Y. Xiang, "An Ensemble Learning Approach for Addressing the Class Imbalance Problem in Twitter Spam Detection," in *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I*, J. K. Liu and R. Steinfeld, Eds. Cham: Springer International Publishing, 2016, pp. 215–228.

[19] S. K. Trivedi and S. Dey, "Effect of feature selection methods on machine learning classifiers for detecting email spams," *Proc. 2013 Res. Adapt. Converg. Syst. - RACS '13*, no. August 2016, pp. 35–40, 2013.

[20] A. Bollinger, H. J. Simon, L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, vol. 92, no. 1, pp. 43–58.

[21] C. Chen, J. Zhang, X. Chen, Y. Xiang, and W. Zhou, "6 million spam tweets: A large ground truth for timely Twitter spam detection," *IEEE Int. Conf. Commun.*, vol. 2015–Septe, pp. 7065–7070, 2015.

[22] H. Shen and X. Liu, "Detecting Spammers on Twitter Based on Content and Social Interaction," *Proc. - 2015 Int. Conf. Netw. Inf. Syst. Comput. ICNISC 2015*, pp. 413–417, 2015.

[23] P. Burnap, A. Javed, O. F. Rana, and M. S. Awan, "Real-time classification of malicious URLs on Twitter using machine activity data," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, 2015, pp. 970–977.

[24] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," *Proc. 3rd ACM Work. Artif. Intell. Secur. - AISec '10*, no. August 2016, p. 54, 2010.

[25] M. McCord and M. Chuah, "Spam detection on twitter using traditional classifiers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6906 LNCS, pp. 175–186, 2011.

[26] X. Zheng, Z. Zeng, Z. Chen, Y. Yu, and C. Rong, "Detecting spammers on social networks," *Neurocomputing*, vol. 159, no. 0, pp. 27–34, Jul. 2015.

[27] Z. Chu, I. Widjaja, and H. Wang, "Detecting social spam campaigns on Twitter," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7341 LNCS, pp. 455–472, 2012.

[28] A. Aggarwal and P. Kumaraguru, "Followers or Phantoms? An Anatomy of Purchased Twitter Followers," Aug. 2014.

[29] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time URL spam filtering service," in *Proceedings - IEEE Symposium on Security and Privacy*, 2011, pp. 447–462.

[30] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. Wiley-IEEE Press, 2013.

[31] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[32] J. C. Ross and P. E. Allen, "Random Forest for improved analysis efficiency in passive acoustic monitoring," *Ecol. Inform.*, vol. 21, pp. 34–39, 2014.

[33] A. Liaw, M. Wiener, and J. Hebebrand, "Classification and regression by randomForest," *R news*, vol. 2, no. 3, pp. 18–22, Dec. 2002.

[34] V. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake, "Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5528, pp. 447–456, 2009.

[35] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley, "Pruning decision trees with misclassification costs," *Mach. Learn. ECML- 98*, vol. 1398, pp. 131–136, 1998.

[36] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[37] G. Forman, "An Extensive Empirical Study of Feature Selection Metrics for Text Classification George," *CrossRef List. Deleted DOIs*, vol. 1, no. 7–8, pp. 1289–1305, 2000.

[38] Q. Xu, E. Xiang, J. Du, J. Zhong, and Q. Yang, "SMS Spam Detection using Content-less Features," *IEEE Intell. Syst.*, no. January, 2012.

[39] M. Dash, H. Liu, M. Dash ', and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, no. 3, pp. 131–156, 1997.

[40] Y. Zhang, S. Wang, P. Phillips, and G. Ji, "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection," *Knowledge-Based Syst.*, vol. 64, pp. 22–31, 2014.

[41] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," *Adv. Neural Inf. Process. Syst. 26*, pp. 431–439, 2013.