Detecting Web Spam in Webgraphs with Predictive Model Analysis

Naw Safrin Sattar Shaikh Arifuzzaman Minhaz F. Zibran Md Mohiuddin Sakib
University of New Orleans, 2000 Lakeshore Drive, New Orleans, LA, USA
{nsattar, smarifuz, mzibran, msakib}@uno.edu

Abstract— **Web spam is a serious threat for both end-users and search engines (w.r.t., query cost). Webgraphs can be exploited in detecting spam. In the past, several graph mining techniques were applied to measure metrics for pages and hyperlinks. In this paper, we justify the importance of webgraph to distinguish spam websites from non-spam ones based on several graph metrics computed for a labelled dataset (WEBSPAM-UK2007) and justify our model by testing on uk-2014 dataset, the most recently available dataset on the same (uk) domain. WEBSPAM-UK2007 dataset includes 0.1 million different hosts and four kinds of feature sets: Obvious, Link, Transformed Link and Content. We use five prominent machine learning (ML) techniques (i.e., Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression, Naïve Bayes and Random Forest) to build a ML-based classifier. To evaluate the performance of our classifier, we compute accuracy and F-1 score and perform 10-fold cross validation. We also compare graph based features with content based textual features and find that graph properties are similar or better than text properties. We achieve above 99% training accuracy for most of our machine learning models. We test our model with uk-2014 dataset with 4.7 million hosts for the graph-based feature sets and achieve accuracy in between 90-94% for most of the models. To the best of our knowledge, prior works on web spam detection with WEBSPAM-UK2007 dataset did not use different test dataset for their models. Our model classifier is capable of detecting web spam for any input webgraph based on its graph metrics features.**

Index Terms—**webgraphs,web spam, machine learning, graph mining, security**

I. Introduction

The study of webgraphs has a significant importance in Web mining, i.e., learning useful structural and organizational information of the Web [1]–[3]. Webgraph is a graph having static HTML pages as nodes (vertices) and directed hyperlinks among the pages as edges [4]. In graph (network) mining, computing various structural properties of webgraphs is challenging due to the size of such graphs. Webgraph has great research potentials concerning web security.

Detecting web spam is one of the aspects among several security vulnerabilities. Web spam is a technique being used by some websites to appear in search engines with high rank but low quality. Cloaking, link spam, buying backlinks, content spam, URL spam, redirection, etc., are some of the tactics of web spamming. Link spam consists of the creation of a link structure, usually a tightly knit community of links, aimed at affecting the outcome of a link-based ranking algorithm. Content spam is done by maliciously crafting the content of web pages [5], for instance, by inserting keywords that are more related to popular query terms than to the actual content of the pages. Cloaking consists of sending different content to a search engine than to the regular visitors of a web site [6]. The aforementioned disastrous effects of web spam motivates our work. Archives [7] are becoming more and more concerned about spam in view of the fact that, under different measurement and estimates, roughly 10% of the websites and 20% of the individual pages constitute spam. The above figures directly translate to 10% to 20% waste of archive resources in storage, processing and bandwidth with a permanent increase. The increasing resource waste will question the economic sustainability of the preservation effort in the near future [8].

Webgraph is a potential source of detecting web spam based on graph based features. Emerging graph mining techniques can be used to detect spam in a scalable manner considering the large size of webgraph. Triangle count, clustering coefficient, triangular density, vertex jaccard similarity, vertex cosine similarity, and centrality measures are among potential features of either pages or hyperlinks to be used as features calculated from webgraphs rather than the contents of the pages. During our study, we face the difficulty to find labelled data of spam/non-spam. It shows the need for a machine learning classifier to predict spam based on the currently available labelled dataset. For this reason, we choose the classic WEBSPAM-UK2007 labelled dataset for developing our model classifier.

Web spam filtering, the domain of devising methods to detect useless and spam web content with the target of manipulating search engine results, has drawn much attention in the past years [9]–[11]. Recently the achievements against the 'classical' web spam seems to be in slow pace [12] and the focus of researchers has apparently altered towards closely related areas such as spam in social networks [13]–[15]. In this study, we emphasize on detecting web spam from webgraph. We figure out the best machine learning technique for each feature set. We compare how the performance vary between graph based features and text based features. We generate graph-based feature set from webgraph for our test dataset and can be applied to any webgraph for feature generation. Our model is tested on different dataset and achieve around 94% accuracy. We also analyze if there is any performance change using different machine learning tools, e.g., Scikit-learn [16] and Weka [17].

The rest of this paper is organized as follows. We describe the related work in Section II. Background of machine learning techniques are discussed in Section III. In Section IV, we describe our dataset, our machine learning classifiers, our validation approaches, generating features for testing data, and improvements to the models with feature selection. A detailed analysis of the results is described in Section V . Finally, Section VI summarizes the findings and concludes the paper with a discussion of future possibilities.

II. Related Works

Several works have been done on web spam but only few focus on webgraph's graph properties to detect web spam. Most of the works have been done based on link spam, content spam and cloaking. Erdélyi et al. investigated how much various classes of web spam features, some requiring very high computational effort, added to the classification accuracy [18]. Zhou et al. developed novel and effective detection methods for link spam target pages using page farms [19]. Ntoulas et al. devised methods for detecting content spam using classifiers [20]. Mishne et al. used language model disagreement [21] for link spam detection. These content spam detection techniques were similar to spam in e-mail. Chellapilla et al. [6] proposed estimating query popularity and monetizability by analyzing search engine query logs and online advertising click-through logs, respectively. They also presented a new measure for detecting cloaked URLs that used a normalized term frequency ratio between multiple downloaded copies of web pages.

Becchetti et al. used triangle count and clustering coefficient to show the distinction between spam and non-spam [1]. Becchetti et al. later extended their work with several link and node based graph metrics for web spam detection [22]. Castillo et al. presented a spam detection system that used the topology of the webgraph by exploiting the link dependencies among the web pages as well as the content of the pages [23]. Authors in [24] used TWSVM (Twin SVM) with two non-linear kernels for spam page detection using WEBSPAM-UK2007, the same dataset as ours. Again, many works [24]–[26] have been done using WEBSPAM-UK2007 for spam detection recently but different from ours. Their work did not emphasize on the graph-based features generated from the webgraph. Iqbal et. al [26] showed that Random Forest is the best classifier for the given dataset. But they have not provided the detailed implementation (i.e. values for the parameters of the machine learning models used) for their work. Therefore, we can not compare our work with theirs. In our experimentation, we have found SVM reigning over the other models during training. The main reason of this distinction might be that they only focused on detecting spam on any of the features irrespective of the feature type. They intermingled all of the feature sets and found Random Forest working best for their model. But we mainly focused on revolving our result through Link based and Transformed Link based features those can be generated from the webgraphs, along with we also analyzed Obvious and Content feature sets as well to gain insights. Some other works [27], [28] also contributed to detect web spam using different machine learning models and datasets. Nevertheless none of those used a different test dataset to show the test accuracy of the models. Besides, they used pre-computed feature sets for the labelled dataset and did not provide any idea how to generate features for different data to test their models.They did not test their model on a different dataset and reported training accuracy only. We have tested our model on a different most recently available dataset, uk-2014 on the same uk domain, and achieved at most 94% accuracy.

III. Preliminaries

In this section we discuss the basics of machine learning techniques we have used throughout the paper.

A. Machine Learning Models

Here we describe the well-known machine learning algorithms for classification that we have used in our experiments.

1) Support Vector Machine: Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems [29]. SVM works by finding a line that best separates the data into two groups. This is done using an optimization process that only considers those data instances in the training dataset that are closest to the line that best separates the classes. The instances are called support vectors, hence the name of the technique [30]. Different SVM algorithms use different types of kernel functions. These functions can be of different types. For example, linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid. The kernel functions return the inner product between two points in a suitable feature space. A linear kernel is used as normal dot product between any two given observations. A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space. RBF can map an input space in infinite dimensional space. Equations for some of the SVM kernels are given in Table I [31]–[33].

TABLE I: Kernel Functions of SVM

Kernel	**Equation**	**Parameters**
Linear	$K(x, x_i) = sum(x * x_i)$	
Polynomial	$K(x, x_i) = 1 + sum(x * x_i)^d$	d is the degree of the polynomial
RBF	$K(x, x_i) = exp(-\gamma * sum((x - x_i^2))$	$\gamma = [0, 1]$
Sigmoid	$K(x, x_i) = tanh(\alpha x^T y + c)$	α is the slope

SVM does not perform well when the dataset is very large– that is due to the required training time becoming higher. It also does not perform very well if the data set has significant noise, i.e., the target classes are overlapping [34]. Considering our dataset size as well as classes, we choose SVM for our classifier.

2) K-Nearest Neighbor: The K-Nearest-Neighbor (KNN) is a non-parametric classification method. It is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions given in Table II). One of the most popular choices to measure this distance is known as Euclidean. KNN classifier requires storing the

TABLE II: Distance Functions for KNN

Distance Function	Equation	Parameters
Euclidean	$D(x,p) = \sqrt{(x-p)^2}$	x and p are the query point and a case from the examples sample
Euclidean squared	$D(x,p) = (x-p)^2$	
City-block	$D(x,p) = \mid (x-p) \mid$	
Chebyshev	$D(x,p) = Max(\mid (x-p) \mid)$	

whole training set and may be too costly when this set is large [35] . KNN algorithm also supports both classification and regression [36]. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors (K is a positive integer, typically small). If $K = 1$, then the object is simply assigned to the class of that single nearest neighbor.

3) Logistic Regression: Logistic regression is a binary classification algorithm [37]. The algorithm learns a coefficient for each input value, which are linearly combined into a regression function and transformed using a logistic (s-shaped) function shown in Equation 1. The function maps any real value into another value between 0 and 1. Logistic regression is a fast and simple technique, but can be very effective on some problems [30].

$$S(z) = \frac{1}{1+e^{-z}} \tag{1}$$

where,
$s(z) =$ output between 0 and 1 (probability estimate)
$z =$ input to the function (algorithm‘s prediction e.g. $mx + b$)
e = base of natural log

4) Naïve Bayes: Naïve Bayes uses a simple implementation of Bayes Theorem (hence naive) where the prior probability for each class is calculated from the training data and assumed to be independent of each other (technically called conditionally independent). There are multiple variations of the Naive Bayes algorithm depending on the distribution of $P(x_i \mid y)$. Three of the commonly used variations are Gaussian, Multinomial and Bernoulli [38]. The Gaussian Naïve Bayes algorithm assumes distribution of features to be Gaussian or normal, i.e.,

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

where, $P(x_i \mid y)$ denotes the conditional probability of an object with a feature vector x_i belonging to a particular class y
σ =standard deviation, μ =mean
The Multinomial Naïve Bayes algorithm is used when the data is distributed multinomially, i.e., multiple occurrences matter a lot. The Bernoulli algorithm is used when the features in the data set are binary-valued. The decision rule for Bernoulli Naïve Bayes is based on Equation 2. It explicitly penalizes the non-occurrence of a feature i that is an indicator for class y [16].

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i) \tag{2}$$

Naïve Bayes has been shown to be a very effective classification algorithm [30]. The Naïve Bayes classifier is surprisingly effective in practice since its classification decision may often be correct even if its probability estimates are inaccurate [39].

5) Random Forest: Random forests are used for robust classification, regression and feature selection analysis. Random Forests are an ensemble of k untrained Decision Trees (trees with only a root node) with M bootstrap samples (k and M do not have to be the same) trained using a variant of the random subspace method or feature bagging method [40]. It is very user-friendly in the sense that it has only two parameters (the number of variables in the random subset at each node and the number of trees in the forest), and is usually not very sensitive to their values [41].

B. Validation of Models

In this section we describe the validation techniques we have used for our classifier. In machine learning we usually split our data into two subsets: training data and testing data so that the model can be trained and tested on different data. It provides a better estimate of out-of-sample performance, but still a ”high variance” estimate. It is useful due to its speed, simplicity, and flexibility [42]. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. The test dataset (or subset) is used in order to test our model‘s prediction on this subset. When dataset is small, splitting data into train and test sets reduces the efficiency of the model as all of the data cannot be used for training. In such case, k-fold cross-validation is used where full dataset can be used for training the model.

Cross validation is a re-sampling procedure used to evaluate machine learning models on a limited data sample [43]. It is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds [44]. In k-fold cross-validation, sample is partitioned into k folds. Each fold is left out of the design process and used as a testing set, and the estimate is the overall proportion of error committed on all folds [45].

IV. Methodology

We describe our dataset, the building of our machine learning classifier, feature selection and validation in this section.

A. Dataset

We have worked with publicly available WEBSPAM-UK2007 dataset [46] consisting of 105,896,555 nodes representing pages and approximately 3.7 billion edges representing hyperlinks to train our model. The collection contains 114,529 different hosts. The dataset was collected by the research group of the Laboratory of Web Algorithmics at the Università degli Studi di Milano. Within the labelled dataset 5.19% was labelled as ’spam’ and 88.33% was ’non-spam’. The rest

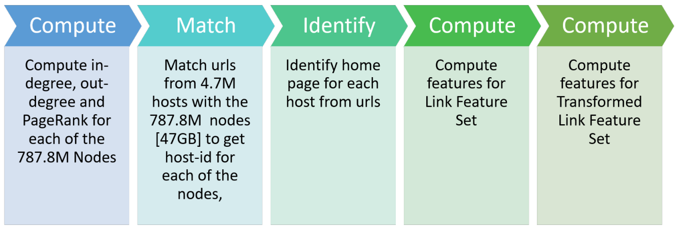
6.48% was labelled 'undecided'. We have not included the undecided data in our classification and filtered out 'spam' and 'non-spam' within the labelled dataset. We have used pre-computed feature set calculated from webgraph and html contents of the pages. The features are computed from the full webgraph for graph-based features. So eliminating the undecided points for building the model does not affect the already computed feature values. The reason is, the features reflect the connectivity of the network, different network properties irrespective of a particular host is taking part in classification model or not. Connectivity among particular groups (spam/non-spam) or the intra-connection among a particular group is not being taken into consideration in current feature set. So, omitting the undecided points does not impact the values of feature sets. A brief description of the features are described in Table III. The obvious feature set has 2 features: the number of pages in the host and the number of characters in the host name. The Link Feature set has been computed from the following graph metrics: PageRank, in-degree, out-degree, Truncated PageRank, and TrustRank. The detailed description of some of the features are given in Subsection IV-B. The Link Feature set consists of features taking the logarithms and ratio of the features from Link Feature Set. Content Feature Set has features generated from text by counting words in the webpages.

Although the data was crawled a long time back, we have chosen to work with it for some specific reasons. The unavailability of labelled data is one of the main reasons. Another important reason is that we mainly want to focus on the webgraph properties and those specific values related to spam class. Also, our work is comparable to others who have used the same dataset very recently. As, we do not emphasize on the text based classification of spam that changes over time, it is quite reasonable to work with a well-known labelled dataset.

TABLE III: Feature Sets used in our Experimentation

Feature Set	**Source**	**Feature Description**	**Count**
Obvious	Graph	the number of pages in the host and the number of characters in the host name	2
Link	Graph	in-degree, out-degree, pagerank and more [22]	41
Transformed Link	Graph	ratio of indegree and outdegree, average, reciprocity, log and more [22]	137
Content	Text	number of words in the home page, average word length, average length of the title, etc., for a sample of pages on each host [23]	96

For Testing our validated model classifier, we have worked with a different dataset that is **uk-2014** webgraph [47]–[49]. This graph is a large snapshot of the .uk domain taken at the end of 2014 . The maximum number of pages per host was set to 10000. The webgraph has 787.8 million nodes and 107 billion edges. The total number of hosts is 4.7 millions, the number of instances for our test dataset.



Compute
Compute in-degree, out-degree and PageRank for each of the 787.8M Nodes
Match
Match urls from 4.7M hosts with the 787.8M nodes [47GB] to get host-id for each of the nodes,
Identify
Identify home page for each host from urls
Compute
Compute features for Link Feature Set
Compute
Compute features for Transformed Link Feature Set

Fig. 1: Overview of Feature Generation Steps

B. Feature Generation for Test Data

We have computed the features for Link and Transformed Link Feature Sets by computing the graph metrics from the uk-2014 webgraph. The steps to generate the feature sets is given in Fig. 1. We have included only those features where the computation involves PageRank, in-degree and out-degree. We have omitted features involving computation of TrustRank and TruncatedPageRank because the computation for these metrics for the labelled dataset is not described and may introduce variation in values. For each host, we include the computation for both home page and the page with maximum PageRank.

1) Link Feature Set: The features we compute for Link Feature Set are the following: *In-degree, Out-degree, PageRank, Assortativity coefficient (ratio of degree and average degree of neighbors) , Average in-degree of out-neighbors, Average out-degree of in-neighbors, Standard deviation of the PageRank of in-neighbors* . So, in total we get 15 features for Link Feature Set.

2) Transformed Link Feature Set: In case of Transformed Feature Set, we similarly compute for both home page and page with maximum PageRank. We take logarithm of each of the previous features for the Transformed Feature Link Set. Besides, we also compute *Ratio of Log of sum of the in-degree of out-neighbors, Log of sum of the out-degree of in-neighbors, Log of ratio of in-degree and PageRank, Log of ratio of out-degree and PageRank, log of ratio of Standard deviation of the PageRank of in-neighbors and PageRank*. Total number of features for Transformed Link Feature Set is 25.

Pseudocode for feature generation is given in Algorithm 1

C. Environment

The experiments have been performed on an Intel Core i7-4770 CPU @ 3.4GHz×8 processor and 16 GB RAM machine. During feature generation, we have used MATLAB to compute the in-degree, out-degree, PageRank from uk-2014 webgraph. Further we use Python Pandas DataFrame to calculate the features. We have used Python Scikit-learn [16] [50] and Weka (Java) [17] [51] to build our classifier based on various machine learning models.

D. Building Classifier

We have used 5 well-known machine learning models to build our classifier. In machine learning there are no best algorithms according to Wolpert's "no free lunch" theorem. Some algorithms work better with some application or data.

Algorithm 1: Graph-based Feature Generation from Webgraph

```
Data: Input Webgraph, $G$ in BVGraph Format; Host Graph, $G_H$
Result: Feature Sets
1  for Each node in G do
2      calculate_PageRank()
3      calculate_In-Degree()
4      calculate_Out-Degree()
5  end
6  for Each node in G do
7      match_url$(G, G_H)$
8      set_pair(node-id,host-id)
9  end
10 for Each node $i$ in $G_H$ do
11     host_list ← get_list(i)
12     find_homepage(i)
13     for Each node $j$ in host_list do
14         $PageRank\_Max[i] \leftarrow$ MAX(PageRank(j))
15         $node\_max \leftarrow j$
           /* Calculate features for Link
              Feature Set                          */
16         in_degree[i] ← SUM(In-degree(j))
17         out_degree[i] ← SUM(Out-degree(j))
18         PageRank[i] ← AVG(PageRank(j))
           /* ............ Calculate Rest
              Features ...............             */
19
20     end
21 end
/* Calculate features for Transformed
   Link Feature Set from the previous
   generated features                              */
```

So we have chosen some of the best algorithms that might work better for our dataset. A brief description of tweaking the parameters for each of the models to build our classifier is described in this section.

1) SVM: For Scikit-learn, we have used different kernels (linear, polynomial, sigmoid, rbf) and gamma values. We get the best result using kernel=rbf, C=1 and gamma=scale. Whereas for Weka, the best accuracy is achieved with polynomial kernel.

2) KNN: At first, we have determined the optimal value of K from Scikit-learn shown in Fig. 2. We have used the same value of K in Weka as well. The optimal values of K for Obvious Feature Set are 6 and the values greater than 7. The maximum accuracy for Link Feature Set can be found for $k = 8, 10; k >= 20$. For Transformed Link Feature Set, $K = 10; K >= 20$ are the optimal values of K. The optimal value for Content Feature Set is 4.

3) Logistic Regression: We have used Scikit-learn default parameters with saga solver and 50000 maximum iterations to get the best accuracy. Weka in its default setting shows the best accuracy.

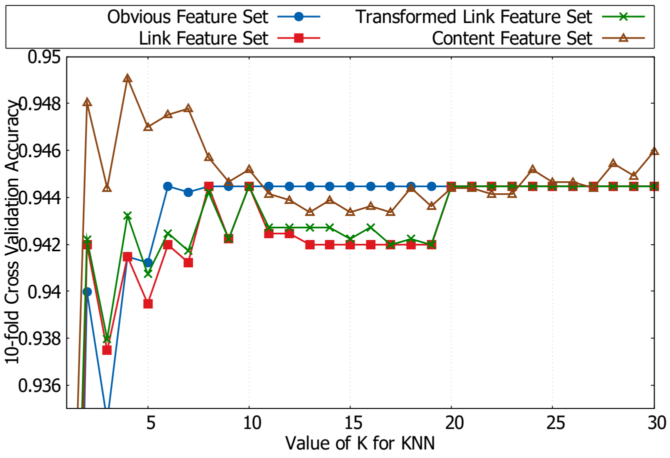


Fig. 2: Determining K value for the Feature Sets

4) Naïve Bayes: We have used both Gaussian and Multinomial Naïve Bayes for Scikit-learn as well as Weka, but better result has been found from Gaussian Naïve Bayes.

5) Random Forest: In Scikit-learn as well as Weka we have used $max\_features = sqrt(n\_features), depth = 0, seed = 1$ and $max\_iterations = 100$. Other parameters have been kept default for both Scikit-learn and Weka.

E. Validation

To validate our model, we have chosen k-fold Cross Validation. k-fold Cross Validation ensures the full dataset is used to train the model. We have used widely accepted value $k = 10$.

We also measure the F1 score of our model. F1 score is harmonic mean of precision and recall. Having a higher Recall means there are less FALSE NEGATIVES. As much as less False Negatives or Zero FN means, model prediction is really good. Whereas having higher Precision means, there are less FALSE POSITIVES. Similarly, Less or Zero False Positives means Model prediction is really good. Thus having a higher F1 score implies good Model Prediction.

F. Feature Selection

We have used "Select Attributes" functionality of Weka for selecting features to improve our classifier. We have used Information Gain as well as Gain Ratio functions to select the attributes. We eliminate the attributes with 0 value for both functions. Both functions choose the same attributes for elimination.

G. Testing Classifier with uk-2014 dataset

Our Test data uk-2014 has 4.7 million instances. Weka runs out of memory if we try to load the full dataset at once. So we divide the test data into multiple chunks for testing. Finally, we take the average of accuracy and F-Measure found from all of the chunks to get result for the full dataset.

V. Result

In this section, we describe several analyses with the derived results for evaluating the accuracy of our classification model.

TABLE IV: 10-fold Cross Validation Accuracy for Scikit-learn and Weka

Model	**Obvious**		**Link**		**Transformed Link**		**Content**	
	Scikit-learn	Weka	Scikit-learn	Weka	Scikit-learn	Weka	Scikit-learn	Weka
SVM	94.44	100	94.42	100	100	99.95	94.60	99.97
KNN	94.45	100	94.45	99.52	94.45	96.27	94.91	99.82
Logistic Regression	94.45	99.95	94.1	99.8	93.75	99.67	94.54	99.71
Naïve Bayes	92.87	99.2	73.11	98.2	83.44	94.17	6.99	36.01
Random Forest	90.75	100	99.98	100	99.45	96.97	99.4	98.47

TABLE V: F-measure for Scikit-learn and Weka

Model	**Obvious**		**Link**		**Transformed Link**		**Content**	
	Scikit-learn	Weka	Scikit-learn	Weka	Scikit-learn	Weka	Scikit-learn	Weka
SVM	0.9	1	0.92	1	1	0.999	0.92	1
KNN	0.92	1	0.92	0.995	0.92	0.954	0.93	0.997
Logistic Regression	0.92	0.999	0.93	0.998	0.92	0.997	0.92	0.997
Naïve Bayes	0.92	0.992	0.83	0.983	0.85	0.95	0.03	0.471
Random Forest	0.95	1	0.97	1	0.99	0.964	0.98	0.984

A. 10-fold Cross Validation Accuracy

Table IV represents the 10-fold cross validation accuracy for Python Scikit-learn and Java Weka for each of the models and feature sets. For Scikit-learn, we can see that SVM has 100% accuracy for Transformed Link feature Set and 94% accuracy for the rest feature sets. KNN and Logistic Regression both have around 94% accuracy for all of the feature sets. For Naïve Bayes only Obvious feature set shows a better accuracy of 92.87%. All feature sets show accuracy above 99.4% except Obvious feature set for Random Forest Model.

Again, for Weka, we find that SVM has 99.9-100% accuracy for all of the feature sets. KNN has above 99.5% accuracy for all feature sets except Transformed Link feature set. For all feature sets, Logistic Regression Model has accuracy above 99.67%. Content Feature Set does not fit to Naïve Bayes Model having a poor accuracy whereas Obvious and Link feature sets show accuracy greater than 98.2%. Obvious and Link feature sets show 100% accuracy for Random Forest Model. Overall, Obvious and Link feature sets show better accuracy of 99.6% on average for all of the models compared to the other two feature sets.

B. F-measure

We have calculated the F-measure score of all of the feature sets for each of the machine learning models shown in Table V. Most of the values are higher and closer to 1 indicating high precision and recall of our models. In our classification task, we intend to build a classifier with high precision and recall. Our Model decides a website is spam or non-spam.We want our model to do the following:

- precisely identify non-spam websites from spam websites (precision)
- identify each website from both spam and nos-spam classes (recall)

It means that we need to select the model that performs well on both metric. So, a high F1 score indicates such model.

By comparing Table IV and Table V, we see that the accuracy and F-measure values do not deviate at all that indicates the uneven class distribution does not affect our model. It implies accuracy is a good measure.

C. Finding Best Model for each Feature Set

In this section we have compared accuracy for all machine learning models and found the best Machine Learning Model for each feature set. For brevity, we have not included the values of F-measure in Table VI as both accuracy and F-measure indicate similar result in our model.

1) Obvious Feature Set: SVM, KNN and Random Forest Models show 100% accuracy in Weka. Again, Naïve Bayes and Logistic Regression Models also have accuracy above 99% in Weka. So, all of the Machine Learning Models are well-fitted for Obvious Feature Set.

2) Link Feature Set: Random Forest Model shows around 100% accuracy in Weka as well as Scikit-learn. SVM also shows 100% accuracy in Weka. Again, KNN, Naïve Bayes and Logistic Regression Models show above 98% accuracy in Weka. So, all of the Machine Learning Models are well-fitted for Link Feature Set.

3) Transformed Link Feature Set: We get around 100% accuracy for SVM shown in both Scikit-learn and Weka. Above 99% accuracy is found for Random Forest Model in Scikit-Learn and for Logistic Regression in Weka. KNN and Naïve Bayes perform moderate. So, SVM is best-fitted for Transformed Link Feature Set considering both Scikit-learn and Weka.

TABLE VI: Machine Learning Model and Tool Selection for each Feature Set

Feature Sets	SVM		KNN		Logistic Regression		Naïve Bayes		Random Forest		Modelling Tools	
	A.(S.,W.)	R.	A.(S.,W.)	R.	A.(S.,W.)	R.	A.(S.,W.)	R.	A.(S.,W.)	R.	Weka	Scikit-learn
Obvious	94.44,100	1	94.45,100	1	94.45,99.95	3	92.87,99.2	4	90.75,100	2	SVM, KNN, L.R., N.B., R.F.	-
Link	94.42,100	2	94.45,99.52	4	94.1,99.8	3	73.11,98.2	5	99.98,100	1	SVM, KNN, L.R., N.B., R.F.	R.F.
Transformed Link	100,99.95	1	94.45,96.27	4	93.75,99.67	3	83.44,94.17	5	99.45,96.97	2	SVM, KNN, L.R., N.B.	SVM, R.F.
Content	94.6,99.97	2	94.91,99.82	3	94.54,99.71	4	6.99,36.0	5	99.4,98.47	1	SVM, KNN, L.R., R.F.	R.F.
Here, A.=Accuracy(%), R.=Rank, S.=Scikit-learn, W.=Weka												

4) Content Feature Set: For Content Feature Set, SVM, KNN and Logistic Regression Models show around 100% accuracy in Weka. Random Forest Model has above 98% accuracy in both Scikit-learn and Weka but Naïve Bayes Model performs poorly and cannot be considered for Content Feature Set.

We have summarized the results in Table VI. We have ranked the machine learning models in ascending order according to highest to lowest accuracy for each feature set. For instance, according to rank, the best machine learning models for Transformed Link Feature Set are SVM, Random Forest, Logistic Regression, KNN and Naïve Bayes respectively. We have also determined the best modelling tool for each of the feature sets. To illustrate, KNN, Logistic Regression and Naïve Bayes Models are well-suited to Weka only whereas Random Forest Model is well-suited to Scikit-learn but SVM works well for both Weka and Scikit-learn for Transformed Link Feature Set. We can see that except Random Forest Model for Transformed Link Feature Set, Weka gives better performance for all of the models and feature sets. For Scikit-learn only Random Forest Model performs well for all the feature sets except Obvious Feature Set.

Regarding the machine learing models, SVM is faster in training, better in accuracy with stability/robustness and works well for each of the Feature Sets. Random Forest is good for balancing error in class population unbalanced data sets reflected in our case. Logistic regression assumes no error in the output variable (y). As we consider removing outliers and possibly misclassified instances from our training data, it works well and is reflected in our result. KNN is a very simple and easy algorithm that even works well for our dataset. Naïve Bayes performs well in case of categorical input variables compared to numerical variables. As our feature sets consist mostly of numerical values, it works poorly for all feature sets specifically, Content Feature Set.

D. Feature Selection for Improvement in Naïve Bayes Model

The accuracy of Naïve Bayes Model with Scikit-learn is less than 90% for all of the feature sets. So, we remove some of the attributes in order to improve the accuracy. Based on the criteria as explained in Section IV-F, we have eliminated several features from our Feature Sets. A summary of our feature selection has been shown in Table VII.

TABLE VII: Feature Selection for Performance Gain

Feature Set	Total Features	Eliminated Features	Existing Features
Link	41	7	34
Transformed Link	137	52	85
Content	96	14	82

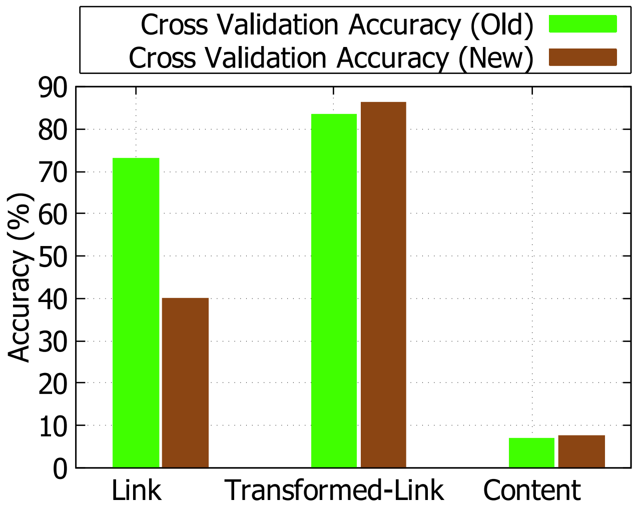


Fig. 3: Changes in Naïve Bayes Accuracy with Feature Selection

Eliminated Link features include:

- Neighbors at distance 3 of home page
- Fraction of out-links that are also in-links of home page
- Fraction of out-links that are also in-links of page with maximum pagerank
- Assortativity coefficient of the home page (ratio of degree and average degree of neighbors)
- trustrank of home page
- trustrank of page with maximum pagerank

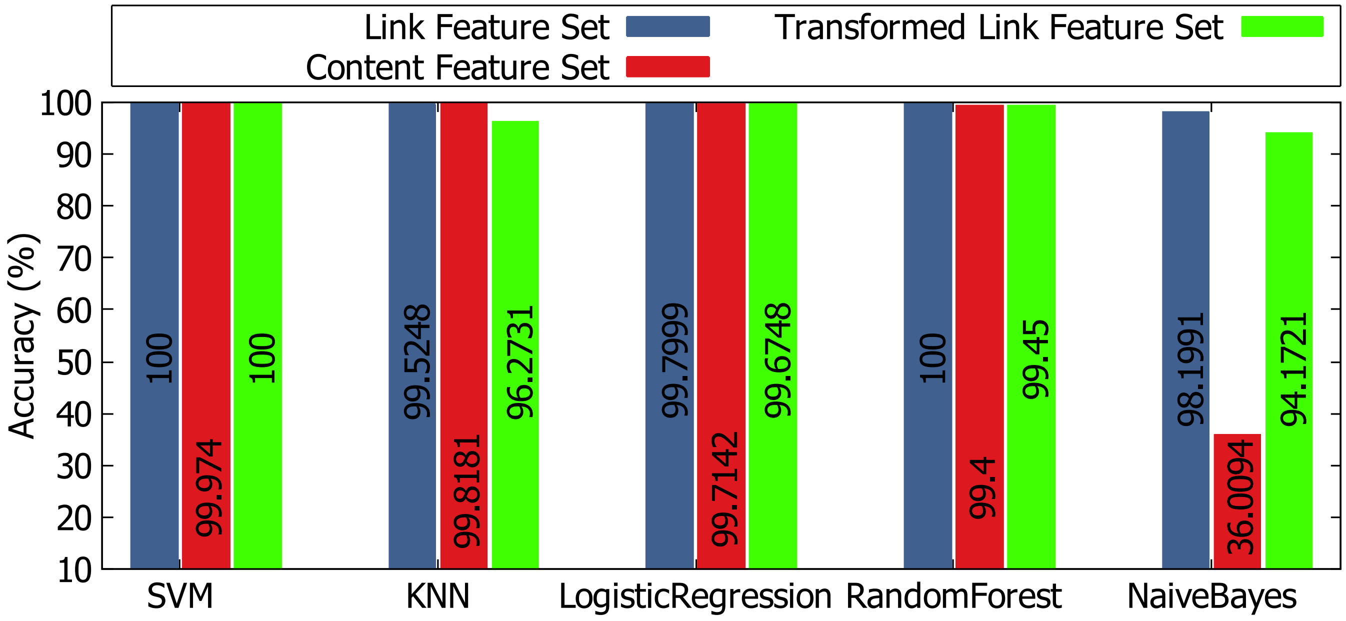


Fig. 4: Comparison between Graph based (Link and Transformed Link) Feature Sets and Text based (Content) Feature Set

TABLE VIII: 10-fold Cross Validation Accuracy and F-measure with Link Feature Set and Transformed Link Feature Set for uk-2014 Test Dataset using Scikit-learn and Weka

Model	**Link**				**Transformed Link**			
	Scikit-learn		**Weka**		**Scikit-learn**		**Weka**	
	Accuracy	F Measure	Accuracy	F Measure	Accuracy	F Measure	Accuracy	F Measure
SVM	90.812	0.854	94.5	0.901	91.2	0.911	90.57	0.905
KNN	85.65	0.699	87.5	0.839	87.78	0.877	88.35	0.854
Logistic Regression	89.2	0.787	86.67	0.84	85.2	0.839	90.2	0.878
Naive Bayes	60.72	0.75	79.4	0.755	68.91	0.73	72.93	0.69
Random Forest	91.25	0.889	92.33	0.865	90.06	0.897	89.89	0.887

Some of the eliminated features from Transformed Link Set are: Ratio of pagerank of the page with maximum pagerank, pagerank of home page; Ratio of neighbors at distance 4 of home page, neighbors at distance 4 of page with maximum pagerank; Ratio of trustrank of page with maximum pagerank, trustrank of home page and many more. A close observation points out that the features related to trustrank have been eliminated from both Linked and Transformed Link feature sets.

After the selection, we get less than 10% improvement shown in Fig. 3. Again, for Link Feature Set the accuracy decreased around 33%. So, selected features did not contribute to an improved performance. For our dataset, feature selection has not been proved to be an acceptable technique.

E. *Graph based vs. Text based Feature Sets*

Extracting the features from contents of the link/ page is somewhat exaggerating process because of huge amount of texts. Whereas calculating several graph-based metrics such as clustering-coefficient, triangle count, ratio of indegree and outdegree etc., from the webgraph is more convenient considering the emerging graph mining techniques. From our analysis we have found that graph metrics based Link Feature Set always provides better or similar accuracy than text based Content Feature Set as shown in Fig. 4.

F. *uk-2014 Dataset Test Result*

Based on our training model classifier, we have tested our model with uk-2014 dataset. We focus only on the Graph based (Link and Transformed Link) Feature Sets. Table VIII shows the 10-Fold Cross Validation Accuracy and F-score for these two feature sets using both Scikit-learn and Weka. For Link Feature Set, we get 90-94% accuracy for both Scikit-learn and Weka using SVM and Random Forest models. KNN and Logistic Regression performs moderately with 85-89% accuracy. In case of Transformed Link Feature Set, the accuracy is around 90% for SVM, Logistic Regression and Random Forest model in Weka. SVM and Random Forest also has 90-91% accuracy with Scikit-learn. Overall, both SVM and Random Forest Model perform well for our test dataset. The higher values of F1-score also indicate good performance of our classifier.

G. *Weka versus Scikit-learn*

We compare the implementation differences of Weka and Scikit-learn for the Machine Learning Models. While working with both of the modelling tools, we have found some dissimilarities. In both of the tools, some parameters do not match at all. For instance, the parameter setting for Logistic Regression Model for Scikit-learn and Weka are different. Parameters related to Scikit-learn are *penalty, dual, tol, C, fit_intercept, intercept_scaling, class_weight, random_state, solver, max_iter, multi_class, verbose, warm_start, n_jobs, l1_ratio*. Whereas for Weka the parameters are: *batchsize, max_its, ridge, useConjugateGradient*. Only common parameter between both is maximum iteration.

We could not compare both in the same scale, still we have kept all of the parameters same those match and compared the result. From our analysis in Section V-C, we see that Weka works better for all of the models for each of the Feature sets. Exception is for only Random Forest Model for Transformed Link Feature Set perceived from Table VI. Overall, we conclude that the default parameter settings for Weka provides better accuracy and is useful for end-users. On the other hand, we need to calibrate the parameters with Trials and Errors to obtain a better accuracy in Scikit-learn. The professional developers are much comfortable with Scikit-learn. Weka saves our valuable time as we need not give much time to adjust the parameters.

H. *Comparison with existing work*

We face difficulty to compare our work with the existing works as most of the works did not provide the model parameters for reproducibility. Again, the intermingling of feature sets do not match. We then compare the performance of our machine learning classifier with an existing work [24], although some information was missing. The authors developed SVM with two kernels. We have achieved a performance gain in simple SVM with rbf kernel as well as their TWSVM Model for Content Feature Set represented in Fig. 5. We have used same dataset as well as same 10-fold Cross Validation accuracy to compare our result with theirs. The authors have not mentioned the parameter settings of their SVM Model, so we have used our own parameter settings providing the best output and achieved the improved performance.

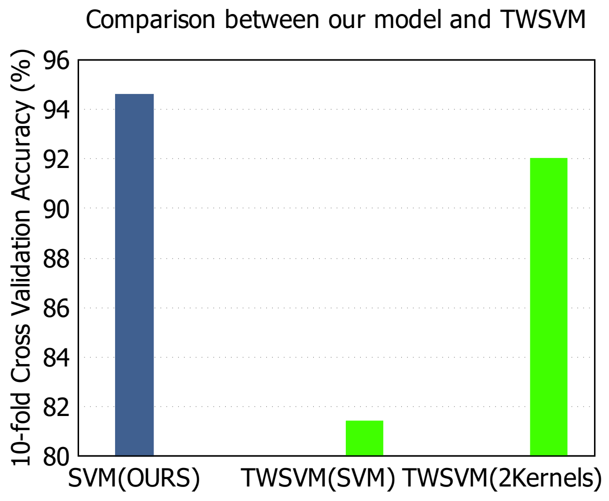


Fig. 5: Performance Gain in SVM Model for Content Feature Set

VI. Conclusion

In this paper, we have developed a machine learning classifier to detect web spam from webgraph. For our labelled dataset, WEBSPAM-UK2007, SVM provides 100% accuracy for all of the Feature Sets. Along with Random Forest and KNN are best-suited with 100% accuracy for Obvious Feature Set. For Link Feature Set, Random Forest also provides 100% accuracy. Naïve Bayes shows 94.17%-99.63% accuracy for all of the feature sets except Content Feature Set. We generate features for test data which can be used for any webgraph for feature generation. Our model shows 90-94% accuracy for our test data with the graph-based features generated from uk-2014 dataset, the most recent available webgraph in uk domain. Using our predictive model classifier, we can detect web spam with graph-based features for any webgraph provided as input. We have found that Weka gives better accuracy compared to Scikit-learn for these feature sets in default parameter settings in most cases. In future, we will generate feature sets based on the values of graph metrics other than in-degree, out-degree and PageRank and check how other metrics perform to detect spam and non-spam websites. We plan to compute the graph metrics with Apache-Hadoop-Spark using Graphx because some webgraphs are very large to handle. For scalable computing, we will use the generated feature sets for our classifier built using Apache MLib to predict webspam with high accuracy. We plan to use our classifier in an existing web archive to check if webspam has been archived there and how much space we can save by removing web spam from the archive.

ACKNOWLEDGMENT

This work has been partially supported by Louisiana Board of Regents RCS Grant LEQSF(2017-20)-RDA-25 and University of New Orleans ORSP SCORE award 2019. We also thank the anonymous reviewers for the helpful comments and suggestions to improve this paper.

References

[1] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 16–24.

[2] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates, "Using rank propagation and probabilistic counting for link-based spam detection," in *Proc. of WebKDD*, vol. 6, 2006.

[3] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna, "A reference collection for web spam," in *ACM Sigir Forum*, vol. 40, no. 2. ACM, 2006, pp. 11–24.

[4] D. Donato, L. Laura, S. Leonardi, and S. Millozzi, "The web as a graph: How far we are," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 1, p. 4, 2007.

[5] Davison and B. D, "Recognizing nepotistic links on the web," *Artificial Intelligence for Web Search*, pp. 23–28, 2000.

[6] K. Chellapilla and D. M. Chickering, "Improving cloaking detection using search query popularity and monetizability." in *AIRWeb*, 2006, pp. 17–23.

[7] A. A. Benczúr, M. Erdélyi, J. Masanés, and D. Siklósi, "Web spam challenge proposal for filtering in archives," in *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*. ACM, 2009, pp. 61–62.

[8] M. Erdélyi, A. A. Benczúr, J. Masanés, and D. Siklósi, "Web spam filtering in internet archives," in *Proceedings of the 5th international workshop on Adversarial information retrieval on the web*. ACM, 2009, pp. 17–20.

[9] A. Singhal, "Challenges in running a commercial search engine," 2005.

[10] M. R. Henzinger, R. Motwani, and C. Silverstein, "Challenges in web search engines," in *ACM SIGIR Forum*, vol. 36, no. 2. ACM, 2002, pp. 11–22.

[11] Z. Gyöngyi and H. Garcia-Molina, "Spam: It's not just for inboxes anymore," *IEEE Computer*, vol. 38, no. 10, pp. 28–34, 2005.

[12] Z. Gyongyi and H. Garcia-Molina, "Web spam taxonomy," in *First international workshop on adversarial information retrieval on the web (AIRWeb 2005)*, 2005.

[13] A. H. Wang, "Detecting spam bots in online social networking sites: a machine learning approach," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2010, pp. 335–342.

[14] X. Niu, G. Liu, and Q. Yang, "Trustworthy website detection based on social hyperlink network analysis," *IEEE Transactions on Network Science and Engineering*, 2018.

[15] M. Callón, J. Fdez-Glez, D. Ruano-Ordás, R. Laza, R. Pavón, F. Fdez-Riverola, and J. Méndez, "Warcprocessor: An integrative tool for building and management of web spam corpora," *Sensors*, vol. 18, no. 1, p. 16, 2018.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[18] M. Erdélyi, A. Garzó, and A. A. Benczúr, "Web spam classification: a few features worth more," in *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality*. ACM, 2011, pp. 27–34.

[19] B. Zhou and J. Pei, "Link spam target detection using page farms," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 3, p. 13, 2009.

[20] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 83–92.

[21] G. Mishne, D. Carmel, R. Lempel *et al.*, "Blocking blog spam with language model disagreement." in *AIRWeb*, vol. 5, 2005, pp. 1–6.

[22] L. Becchetti, C. Castillo, D. Donato, R. Baeza-Yates, and S. Leonardi, "Link analysis for web spam detection," *ACM Transactions on the Web (TWEB)*, vol. 2, no. 1, p. 2, 2008.

[23] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri, "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 423–430.

[24] S. H. R. Mohammadi and M. A. Z. Chahooki, "Web spam detection using multiple kernels in twin support vector machine," *arXiv preprint arXiv:1605.02917*, 2016.

[25] M. Luckner, "Practical web spam lifelong machine learning system with automatic adjustment to current lifecycle phase," *Security and Communication Networks*, vol. 2019, 2019.

[26] M. Iqbal, M. M. Abid, U. Waheed, and S. H. Alam Kazmi, "Classification of malicious web pages through a j48 decision tree, a naïve bayes, a rbf network and a random forest classifier for webspam detection," 2017.

[27] A. Makkar, M. S. Obaidat, and N. Kumar, "Fs2rnn: Feature selection scheme for web spam detection using recurrent neural networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[28] K. L. Goh and A. K. Singh, "Comprehensive literature review on machine learning structures for web spam classification," *Procedia Computer Science*, vol. 70, pp. 434–441, 2015.

[29] M. Awad and R. Khanna, "Support vector machines for classification," in *Efficient Learning Machines*. Springer, 2015, pp. 39–66.

[30] J. Brownlee. How to use classification machine learning algorithms in weka. [online] available at:. [Online]. Available: https://machinelearningmastery.com/use-classification-machine-learning-algorithms-weka

[31] A. Ng, "Cs229 lecture notes," *CS229 Lecture notes*, vol. 1, no. 1, pp. 1–3, 2000.

[32] D. Team. Kernel functions-introduction to svm kernel & examples. [online] available at:. [Online]. Available: https://data-flair.training/blogs/svm-kernel-functions

[33] D. Community. Support vector machines in scikit-learn. [online] available at:. [Online]. Available: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python#kernels

[34] S. Ray. Understanding support vector machine algorithm from examples (along with code). [online] available at:. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code

[35] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2003, pp. 986–996.

[36] K-nearest neighbors algorithm. [online] available at:. [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[37] I. Kurt, M. Ture, and A. T. Kurum, "Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease," *Expert systems with applications*, vol. 34, no. 1, pp. 366–374, 2008.

[38] R. Jain. Introduction to naive bayes classification algorithm in python and r. [online] available at:. [Online]. Available: https://www.hackerearth.com/blog/machine-learning/introduction-naive-bayes-algorithm-codes-python-r

[39] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.

[40] H. Lan. Decision trees and random forests for classification and regression pt.2. [online] available at:. [Online]. Available: https://towardsdatascience.com/decision-trees-and-random-forests-for-classification-and-regression-pt-2-2b1fcd03e342

[41] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[42] A. Bronshtein. Train/test split and cross validation in python towards data science. [online] available at:. [Online]. Available: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

[43] J. Brownlee. A gentle introduction to k-fold cross-validation. [online] available at:. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation

[44] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: with Applications in R (Springer Texts in Statistics)*, 1st ed. Springer, 2013, vol. 112.

[45] U. M. Braga-Neto and E. R. Dougherty, "Is cross-validation valid for small-sample microarray classification?" *Bioinformatics*, vol. 20, no. 3, pp. 374–380, 2004.

[46] C. Castillo. Webspam-uk2007 (current dataset). [online] available at:. [Online]. Available: http://chato.cl/webspam/datasets/uk2007

[47] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*. Manhattan, USA: ACM Press, 2004, pp. 595–601.

[48] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," in *Proceedings of the 20th international conference on World Wide Web*, S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, and R. Kumar, Eds. ACM Press, 2011, pp. 587–596.

[49] P. Boldi, A. Marino, M. Santini, and S. Vigna, "BUbiNG: Massive crawling for the masses," in *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2014, pp. 227–228.

[50] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler *et al.*, "Api design for machine learning software: experiences from the scikit-learn project," *arXiv preprint arXiv:1309.0238*, 2013.

[51] F. Eibe, M. Hall, and I. Witten, "The weka workbench. online appendix for" data mining: Practical machine learning tools and techniques," *Morgan Kaufmann*, 2016.