2020 Global Smart Industry Conference (GloSIC)

The Detecting Cross-Site Scripting (XSS) Using Machine Learning Methods

Stanislav Kascheev
South Ural State University (national research university)
Chelyabinsk, Russia
kashcheevs@susu.ru

Tatyana Olenchikova
South Ural State University (national research university)
Chelyabinsk, Russia
olenchikovati@susu.ru

Abstract—This article discusses the problem of detecting cross-site scripting (XSS) using machine learning methods. XSS is an attack in which malicious code is embedded on a page to interact with an attacker's web server. The XSS attack ranks third in the ranking of key web application risks according to Open Source Foundation for Application Security (OWASP). This attack has not been studied for a long time. It was considered harmless. However, this is fallacious: the page or HTTP Cookie may contain very vulnerable data, such as payment document numbers or the administrator session token. Machine learning is a tool that can be used to detect XSS attacks. This article describes an experiment. As a result the model for detecting XSS attacks was created. Following machine learning algorithms are considered: the support vector method, the decision tree, the Naive Bayes classifier, and Logistic Regression. The accuracy of the presented methods is made a comparison.

Keywords—cross-site scripting, machine learning, XSS attack

I. INTRODUCTION

Web applications are widely used and include sensitive and personal data. This makes them a target for malware that uses vulnerabilities to get unauthorized data stored on the computer. Such attacks include cross-site scripting (XSS).

Cross-Site scripting (XSS) [1] is a type of attack on web systems that involves inserting malicious code into a page issued by a web system (which will be executed on the user's computer when they open this page) and interacting this code with the attacker's web server. This is a type of "code Injection" attack.

The specific feature of such attacks is that malicious code can use the user's authorization in the web system to get extended access to it or to get the user's authorization data. Malicious code can be inserted into a page either through a vulnerability in the web server or through a vulnerability on the user's computer.

The term is abbreviated "XSS" to avoid confusion with cascading style sheets that use the abbreviation "CSS".

XSS can affect the victim by stealing cookies, changing the web page, capturing clipboard content, scanning ports, or dynamically loading. Therefore, the security of web applications is a very important task for developers. The lack of client input verification is the most common security flaw in web applications. These shortcomings are repeatedly detected and exploited both on the client side and on the server side. XSS attacks remain in the top ten vulnerabilities listed in the Open Web Application Security Project (OWASP) [2].

XSS attacks have not been studied for a long time. Such attacks were considered not dangerous. However, this opinion is considered erroneous now. On a site where there is no protection against cross-site request forgery (CSRF), an attacker can perform any actions available to the user. But even with this protection, the page or HTTP Cookie may contain very vulnerable data (for example, the administrator session ID or payment document numbers). Cross-site scripting can be used to conduct a DoS attack.

The article explores the use of machine learning methods for building classifiers that allow detecting XSS in JavaScript. Currently, the focus is on research on passive or active XSS, where a malicious script is entered into a web application and stored in a database. Then every time you visit this page, the script will be executed in the user's browser. This type of attack can target blogs, forums, comments, or profiles.

The work of detecting and protecting against XSS attacks can be broadly divided into three types:

- static analysis. In this analysis the source code is examined. This type of analysis may provide formal guarantees that certain vulnerabilities do not occur, but it may also be slow or not produce results;
- dynamic analysis. This analysis attempts to determine the actions of the script during its execution. The strategy for this analysis is to change the interpreter or check the syntactic structure;
- machine learning. Machine learning uses knowledge about available scripts to build classifiers and predicts the behavior of new scripts.

There are three advantages of using machine learning. First, the classifier can quickly predict whether a scenario is malicious. Second, the classifier does not need an Autonomous environment designed for scenario analysis. Third, the classifier can predict the appearance of new JavaScript malware.

The work of detecting XSS attacks using machine learning methods is not new. Thus, in [3], the authors investigated the accuracy of the naive Bayesian classifier, the support vector method, the k-nearest neighbor algorithm, and the decision tree. In this work, 3 experiments were performed. The first experiment used the entire data set for training. In the second case, the data set is divided into training and test samples. The training sample contains 80% of the data set, and the test sample contains 20%. In the third experiment, the authors used cross-validation to separate the data. The training sample contained

978-1-7281-8075-5/20/$31.00 ©2020 IEEE

90% of the data set, and the test sample contained 10%. The K-nearest neighbor algorithm showed better results in all experiments compared to other methods under study. Thus, the accuracy rate for this method reached 99%.

The results of research on cross-site scripting detection are given in articles [4–6]. The work of such methods as the support vector method, the k-nearest neighbor method, and the random forest method was compared. These methods use syntactic and behavioral features of JavaScript. As a result, the methods showed good outcomes. The accuracy reached 99.75%.

The authors of the above articles use functions, structural or behavioral features from JavaScript as attributes.

This paper describes an experiment that results in a model for detecting XSS attacks. The most popular parameters found in JavaScript are used as attributes. Machine learning algorithms are considered: the method of support vectors, the decision tree, the naive Bayesian classifier, and logistic regression. The accuracy of these machine learning methods is compared.

II. Statement of the Problem

The task of detecting cross-site scripting using machine learning methods is the task of binary classification of incoming requests. Fig. 1 shows the algorithm for classifying incoming requests. Incoming requests may relate to malicious code or benign code. Before training, you must present each query as a vector of token words. These tokens are request parameters in JavaScript. A set of tokens makes up the model dictionary.

Cross-validation is used to generate training and test samples. The training sample takes part in building the model. The test sample is used at the stage of checking the result of the algorithm using metrics.

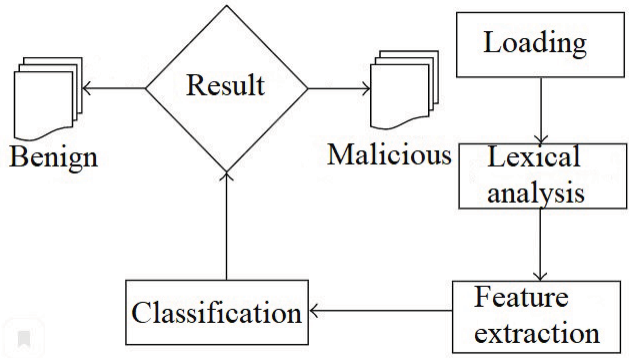

Fig. 1. Scheme of the algorithm for classifying JavaScript code

The model is checked for the percentage of correct responses, accuracy, completeness, and F-measure.

For the experiment, we selected algorithms for supervised learning, since the task of detecting XSS attacks is a classification task.

III. Building a Model for Classification

A. Dataset

To get a model with good accuracy, you need to assemble and prepare the source dataset correctly. The data set must contain malicious and benign JavaScript code. The data set for the experiment consists of 240,000 instances of malicious and benign code. Malicious code is 40,000 instance [7], and benign code is 200,000 instance [8].

B. Preprocessing of data

The source data set is represented as queries with a different number of parameters. Each request is decoded into Unicode characters. After decoding, parameters of this query are extracted using regular expressions. Abnormal queries containing a large number of parameters are removed from the dataset.

C. Building a learning model

At the stage of building the model, each request is presented as a list of parameters. A dictionary of tokens is being compiled. In this model, the dictionary size is 3000 words, which are often found in the data set. After compiling the dictionary, the source data set is represented as a set of word vectors using word2vec [9]. Word2vec is a tool for calculating vector representations of words.

Fig. 2 shows a conditional similarity for the 100 most popular tokens. These tokens are represented in a low-dimensional space according to joint probabilities with other tokens. This means that the closer a pair of tokens is located on the chart, the more often they are found together in the same query. A three-dimensional display space is used, and the third dimension is represented by color. Words can be considered close if they are located next to each other and have the same hue.

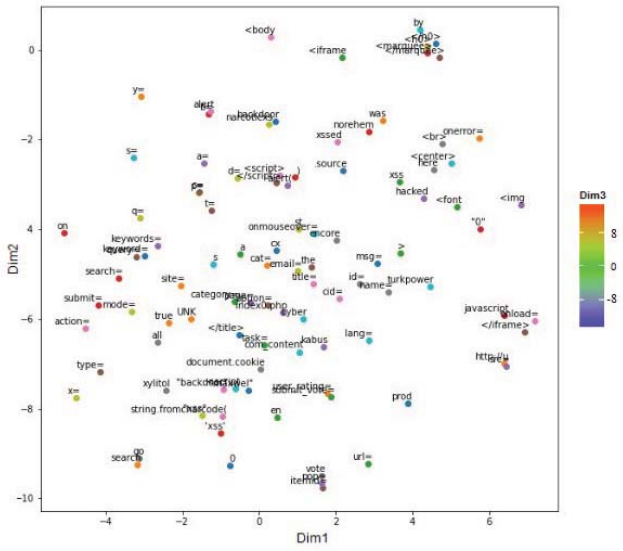

Fig. 2. Conditional similarity of tokens

Joint probabilities are obtained using the method of T-distributed Stochastic Neighbor Embedding (t-SNE) [10]. This method allows you to reduce the data dimension and restore the data nesting structure. The t-SNE method first calculates the probabilities p_{ij} that are proportional to the similarity of objects x_i and x_j as follows (1):

$$p_{j|i} = \frac{\exp(-|x_i - x_j|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-|x_i - x_k|^2 / 2\sigma_i^2)}, \quad (1)$$

where:

x_i, x_j, x_k – high-dimensional objects from N;
$p_{j|i}$– conditional probability x_i and x_j;
σ_i– dispersion for x_i.

The similarity between a pair of data points is defined as a symmetric variant of the conditional similarity by (2):

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \quad (2)$$

where:

$p_{j|i}$ – conditional probability x_i and x_j;
$p_{i|j}$– conditional probability x_j and x_i;
N– number of high-dimensional objects.

The similarity of the token varies over a wide range and tokens do not form distinct clusters. Regression methods will not work in this case. Therefore, machine learning methods are required for data analysis.

D. Machine learning methods

For the experiment, we selected machine learning algorithms for teaching with a teacher, since the task is a binary classification of incoming requests. The following is a description of each method.

E. Decision tree

The decision tree [11] is a decision support tool used in machine learning, data analysis, and statistics. The tree structure consists of leaves and nodes. Each leaf represents the value of a target variable that has changed as it moves from the root to the leaf. Each internal node corresponds to one of the input variables.

The tree is trained by dividing the source sets of variables into subsets based on testing attribute values. This is a process that is repeated on each of the resulting subsets. The recursion is completed when the subset at a node has the same values of the target variable. So it doesn't add value to predictions.

F. Naive Bayes classifier

The method of the naive Bayes classifier [12] is based on the Bayes theorem with an independent assumption between predictors. A naive Bayesian model is easy to construct when the dimension of the data set is very large. In the context of the learning process, the classifier tags training data for certain tokens and counts the number of occurrences in each class. Based on this process, the probability of each class is calculated with some test data and classifies this test data for the class that has the highest probability. This probability is calculated by (3).

Despite its simplicity, the naive Bayesian classifier is widely used because it is superior to more complex classification methods.

$$p(\mathrm{c}|d) = \frac{p(d|\mathrm{c}) * p(\mathrm{c})}{p(d)}, \quad (3)$$

where:

$p(\mathrm{c}|d)$– the probability that the parameter d belongs to class c;
$p(d|c)$– probability of meeting parameter d among all parameters of class c;
$p(\mathrm{c})$– the absolute probability of meeting an instance of class c in the dataset.
$p(d)$– absolute probability of finding parameter d in the dataset;

G. Logistic regression

Logistic regression [13] is used to predict the probability of occurrence of a certain event based on the values of a set of features. To do this, enter the so-called dependent variable y. This variable takes only one of two values – usually the numbers 0 (the event did not occur) and 1 (the event occurred). We also introduce a set of independent variables (also called signs, predictors, or regressors) – real $x_1, x_2, \ldots, x_n$. Based on the input values, you need to calculate the probability of accepting a particular value of the dependent variable by (4).

$$P = \frac{1}{1 + exp^{-y(x_1, x_2, \ldots, x_n)}}, \quad (4)$$

where:

P – probability that the event will occur;
$y\ (x_1, x_2, \ldots, x_n)$ – the standard regression equation.

H. Support vector machine

The support vector method [14] is a set of similar learning algorithms used for classification and regression analysis tasks. A special property of the support vector method is a continuous reduction in the empirical classification error and an increase in the gap. This is why the method is also known as the maximum gap classifier method.

The main idea of the method is to translate the source vectors into a higher-dimensional space and search for a dividing hyperplane with a maximum gap in this space. Two parallel hyperplanes are constructed on both sides of the hyperplane that separates the classes. The separating hyperplane is a hyperplane that maximizes the distance to two parallel hyperplanes. The algorithm works under the assumption that the greater the difference or distance between these parallel hyperplanes is, the smaller the average classifier error will be.

I. Metrics

Metrics is one of the key indicators. It is used to evaluate the result of the algorithm.

To evaluate the accuracy of the implemented algorithms we used metrics such as: accuracy, precision, recall and F-measure [15].

Accuracy – the percentage of correct responses of the algorithm is calculated by (5). If the problem uses unequal classes, as in this case, this metric becomes useless. Therefore, several other metrics are selected for the issue along with this metric.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (5)$$

where:

TP (True Positives) – correctly classified positive examples;
TN (True Negatives) – correctly classified negative examples;
FN (False Negatives) – positive examples classified as negative (type I error);
FP (False Positives) – negative examples classified as positive (type II error).

Precision [16] may be interpreted as the percentage of objects that are called malicious by the classifier and are actually malicious. This value is calculated by (6).

$$precision = \frac{TP}{TP+FP}, \quad (6)$$

where:

TP (True Positives) – correctly classified positive examples;
FP(False Positives) – negative examples classified as positive (type II error).

Recall [17] shows what percentage of malicious instance objects out of all malicious instances in the test sample the algorithm found. This value is calculated by (7).

$$recall = \frac{TP}{TP+FN} \quad (7)$$

where:

TP (True Positives) – correctly classified positive examples;
FN (False Negatives) – positive examples classified as negative (type I error).

There are several different ways to combine precision and recall into an aggregated quality criterion. F-measure (in general) - average harmonic precision and recall:

$$F_\beta = (1+\beta^2) * \frac{precision*recall}{(\beta^2*precision)+recall}, \quad (8)$$

where:

β– weight of accuracy in the metric;
$precision$ – the percentage of objects that the classifier calls malicious, but is actually malicious;
$recall$ – the percentage of malicious instance objects that the algorithm found.

The F-measure reaches a maximum when the completeness and accuracy are equal to one, and is close to zero if one of the arguments is close to zero.

IV. Experimental Results and Analysis

The prepared dataset is divided by cross-validation into training and test samples. The percentage of the training sample is 70%. The percentage of the test sample is 30%. Cross-validation [18] is a procedure for empirically evaluating the generalizing ability of algorithms. Cross-validation emulates the presence of a test sample that does not participate in training, but for which the correct answers are known.

Each instance is represented as a token vector. Tokens are replaced with the corresponding item number of the parameter in the dictionary. Abnormal instances were deleted during the dataset preparation stage. Such data may reduce the accuracy of the built model. Each vector is represented by 256 values. This dimension for the vector is not chosen by chance. The dimension of each vector was set according to the vector with the maximum number of non-zero values. This means that the longest query used for building the model contains 256 parameters. Thus, the training matrix is a sparse matrix of dimension 168532x256.

The test sample contains 72229 instances, of which 12242 instances are examples of malicious code.

Fig. 3 shows the value of the F-measure for each of the methods. The best results for this metric are shown by the decision tree, after the Bayesian classifier, the support vector method. In last place is the logistic regression, for which the score (8) for malicious code is 0%.

Fig. 4 shows the full accuracy values for the algorithms. As described above, it is not practical to use only this metric. So, for the decision tree, the percentage of correct answers was 98.81%. Then the logistic regression is 83.03%, the support vector method is 71.37%, and the Bayesian classifier is 65.27%.

Fig. 5 shows the percentages of detected malicious code objects for machine learning algorithms. The best result was shown by the decision tree – 99.19%, the worst result was shown by the support vector method – 27.82%.

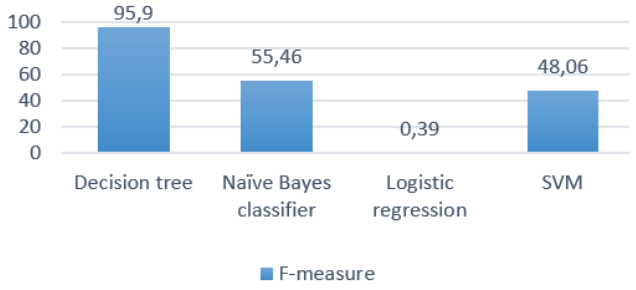

Fig. 3. Comparison of methods by F-measure

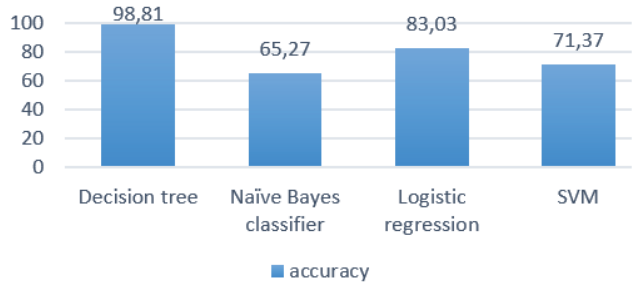

Fig. 4. Comparison of methods by accuracy

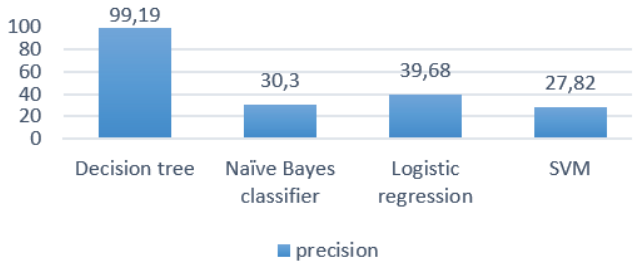

Fig. 5. Comparison of methods by precision

Fig. 6 shows the values of the recall metric. The highest result is shown by the decision tree 93.70%. The worst result is shown by logistic regression. The recall value for it was 0.2%.

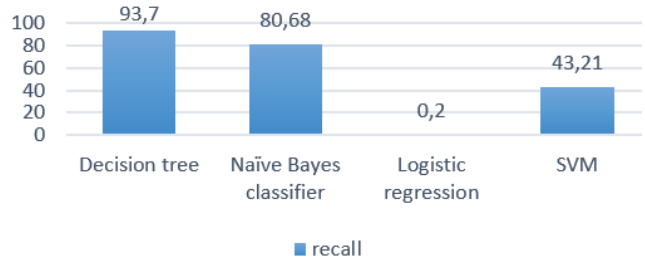

Fig. 6. Comparison of methods by recall

The results of the experiment are presented in summary table 1.

TABLE I. EXPERIMENTAL RESULT

Metric	Method evaluation %			
	Decision tree	*Naïve Bayes classifier*	*Logistic regression*	*SVM*
F-measure	95,90	55,46	0,39	48,06
accuracy	98,81	65,27	83,03	71,37
precision	99,19	30,30	39,68	27,82
recall	93,70	80,68	0,20	43,21

A. Analysis of results

The evaluation will not be impartial if you analyze the results of the algorithm for only one metric. Therefore, in this task, it is necessary to conduct research on a set of metrics. According to the conducted metrics, the best result was shown by the model built using the decision tree. For this model, the accuracy of detecting malicious code is 93.70%, full accuracy is 98.81%, and the accuracy of predicting malicious code is 99.19%.

For other algorithms, the indicators are worse. For example, the Bayesian classifier has many placemarks. As is known, the naive Bayesian classifier is based on the application of the Bayes theorem with strict independence assumptions. This means that the probability of one parameter should not affect the probability of another parameter, but there is a conditional similarity between the parameters of one query. It is shown in figure 1. Also, the presence of such a large number of tags can be explained by a slight difference between the parameters of malicious and normal requests.

Similarly to the Bayesian classifier, the support vector method has placed many labels, and the percentage of malicious instances found is lower. This means that the classifier has produced many false positives. The main idea of this method is to search for a dividing hyperplane with the maximum gap between classes. The algorithm works under the assumption that the greater the difference or distance between these parallel hyperplanes is, the smaller the average classifier error will be. One of the disadvantages of the support vector method is its instability to noise. Outliers or abnormal instances in the training sample become reference intruder objects and directly affect the construction of the dividing hyperplane. Some vectors in the training sample had many parameters, which is why the support vector method could not correctly construct the dividing hyperplane.

The model built using logistic regression, on the contrary, placed too few labels, which is due to an unbalanced data. In practice, malicious code is less common than normal. In the original data set, malicious code makes up 17%. For logistic regression, the data is not proportional.

V. FUTURE WORK

Detecting XSS attacks is quite a serious task that requires further research. With each new attack, the approach to embedding malicious code on the page becomes more complex. This type of attack causes business losses, increases the number of fraudulent transactions, and leaks of confidential data.

As a further development, several approaches are proposed. One of them is to balance the data for the task [19]. Data balancing is achieved by generating new instances of malicious code. To generate new samples, you must either create a generative model, or select existing samples with replacement. The main advantage of this approach is that a single model can take into account all the data at the same time, and also helps generate new data. Another way to balance data is to equalize the number of class instances in the dataset. In this case, the dataset will include 40,000 samples of malicious code and 40,000 samples of good-quality code. However, in this case, 160,000 instances of good-quality code will not be used in the dataset, which is a disadvantage of this data balancing method.

The second approach implies the use of an ensemble of trees [20]. The ensemble consists of decision trees, the number of which is specified by creating the ensemble and aggregating module. The task of the module is to combine the conclusions of single decision trees. An ensemble of classifiers receives a training sample, which is then randomly divided into two parts for each decision tree in a given relation, the training sub-sample itself and the validation sub-sample. The training sub-sample is used to train the decision tree, and the validation sub – sample is used to evaluate the quality of classification and assign a certain weight to the tree. Thus, the decision trees in the ensemble are different, since they are trained on different subsamples.

A special feature of this work is the use of the most popular parameters found in queries as attributes. The experiment showed that for this set of features, the most optimal method of the machine learning algorithm is the decision tree. The current detection accuracy of malicious code for a model built using this method makes up 93%.

REFERENCES

[1] S. Kirsten, "Cross Site Scripting (XSS)," OWASP Foundation, 2020.

[2] B. Glas, "OWASP Top 10 Security Risks & Vulnerabilities," OWASP Foundation, 2020.

[3] A. Johari and S. K. Adnan, "Defending Malicious Script Attacks Using Machine Learning Classifiers," Wireless Communications and Mobile Computing, 2017.

[4] F. A. Mereani and J. M. Howe, "Detecting Cross-Site Scripting Attacks Using Machine Learning," Advanced Machine Learning Technologies and Applications, vol. 723, pp. 200–210, 2018.

[5] F. A. Mereani and J. M. Howe, "Preventing Cross-Site Scripting Attacks by Combining Classifiers," Proc. of the 10th Int. Joint Conf. on Computational Intelligence, vol. 1, pp. 135–143, 2018.

[6] F. A. Mereani and J. M. Howe, "Exact and approximate rule extraction from neural networks with boolean features," Proc. of the 11th Int. Joint Conf. on Computational Intelligence, vol. 1, pp 424–433, 2019.

[7] XSS Archive, In Famous and Government Websites.

[8] S. S. Shah, Cross site scripting XSS dataset for Deep learning. Kaggle, 2020.

[9] Word2vec, The Python Package Index. [Online]. Available: https://pypi.org/project/word2vec/

[10] TSNE, scikit-learn: machine learning in Python. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[11] S. Rajesh, "Decision Trees – A simple way to visualize a decision," Towards Data Science, 2018.

[12] Naïve Bayes, scikit-learn: machine learning in Python. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html

[13] Logisitic Regression, scikit-learn: machine learning in Python. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[14] C. Zoltan, "SVM and Kernel SVM," Towards Data Science, 2018.

[15] S. Ghoneim, Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on? 2019.

[16] Precision score, scikit-learn: machine learning in Python. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

[17] Recall score, scikit-learn: machine learning in Python. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

[18] C. Neale, D. Workman, and A Dommalapati, "Cross Validation: A Beginner's Guide," Towards Data Science, 2019.

[19] Working with unbalanced data sets for classification. Machine Learning-2020.

[20] A. Zimmermann, "Ensemble-Trees: Leveraging Ensemble Power Inside Decision Trees," Discovery Science: 11th Int. Conf., pp. 76–87, 2008.