Ensemble Methods to Detect XSS Attacks

PMD Nagarjun[1], Shaik Shakeel Ahamad[2]*
Department of CSE, K L University, Vijayawada, India[1]
College of Computer and Information Sciences (CCIS)
Majmaah University, Al Majmaah, Kingdom of Saudi Arabia[2]

***Abstract*—Machine learning techniques are gaining popularity and giving better results in detecting Web application attacks. Cross-site scripting is an injection attack widespread in web applications. The existing solutions like filter-based, dynamic analysis, and static analysis are not effective in detecting unknown XSS attacks, and machine learning methods can detect unknown XSS attacks. Existing research to detect XSS attacks by using machine learning methods have issues like single base classifiers, small datasets, and unbalanced datasets. In this paper, supervised ensemble learning techniques trained on a large labeled and balanced dataset to detect XSS attacks. The ensemble methods used in this research are random forest classification, AdaBoost, bagging with SVM, Extra-Trees, gradient boosting, and histogram-based gradient boosting. Analyzed and compared the performance of ensemble learning algorithms by using the confusion matrix.**

***Keywords*—Cross-site scripting; machine learning; ensemble learning; random forest; bagging; boosting**

I. Introduction

Machine learning algorithms are useful in detecting unknown and new XSS attacks in Web Applications. Ensemble methods are a combination of different base models, and the ensemble learning models can give optimal results compared to base models [1]. In XSS attacks, the attacker can steal victim's session cookie, sensitive data of victim, implement keyloggers at browser, and damage the reputation of a trusted Website.

A common problem in existing XSS prevention techniques are the incapability of detecting unknown or new XSS attacks [2]. Highly effective XSS detection models can be built by using ensemble learning techniques. AdaBoost, bagging, Extra-Trees, gradient boosting, random forest, histogram-based gradient boosting are ensemble methods, which uses base models like decision trees, etc.

Cross-site scripting injection attacks are categorized into three types, and they are persistent (stored), non-persistent (reflected), and DOM-based attacks. Many existing solutions primarily focused on preventing only one type of XSS attack, and there are only a few solutions to avoid all types of attacks [3]. The proposed ensemble learning models can detect all types of attacks by proper implementation at the server and client-side.

Ensemble methods use different algorithms to achieve better prediction rate. Usually, ensemble learning involves the same base learning algorithm. The limitation in ensemble methods is that these require more computations compared to a single model. In ensemble learning base models are combined in three ways.

Bagging: In bagging (bootstrap aggregation) weak learning algorithms applies on a small sample dataset and takes an average of all learners prediction. Bagging will decrease the variance.

Boosting: It is an iterative method, in boosting sample weights are adjusted based on the previous classification. Boosting will decrease bias error.

Stacking: In this output of one model is given as input to another model. Stacking will decrease variance or bias based on models used.

The purpose of this paper is to investigate and compare the prediction accuracy of machine learning ensemble methods in detecting Cross-site scripting attacks in Web Applications.

The paper is organized as follows: Section 2 contains related work. We prepared XSS data for training and testing in Section 3. We implemented the ensemble learning models in Section 4. We analyzed the performance of proposed ensemble models in Section 5. Sections 6 and 7 contains conclusion and future work.

II. Related Work

Rodriguez et al. [4] analyzed 67 documents related to XSS attacks. According to their research, most of the researches use browser tools or web page analysis methods to prevent XSS attacks, very few researches on machine learning algorithms to prevent these attacks. Based on their research most common issues in existing researches are detecting only one type of XSS attacks, low attacks data, only restricted to one programming environment like PHP, same data for different researches, methods not scalable, high false positives, methods work on only one browser, few methods proposed to use artificial intelligence, etc.

S. Gupta and B. B. Gupta [5] did a study on defense mechanisms of XSS attacks, and they stated that safe input handling is one of the essential techniques to mitigate XSS attacks. A good XSS defensive technique needs to differentiate malicious code and legitimate JavaScript code automatically.

Hydara et al. [6] studied 115 research papers on XSS attacks. Based on their study, non-persistence XSS attacks are popular, and there is a need for solutions to remove XSS vulnerabilities from the source code.

Shanmugasundaram et al. [7] stated that developers lack knowledge on implementing existing XSS solutions in their web applications.

*Corresponding Author

Aliga et al. [8] study showed that most of the XSS prevention solutions are client-side, and they are unable to detect new XSS attacks, and these solutions lack self-learning capabilities. They reviewed 15 XSS prevention techniques, and out of 15, only two techniques have self-learning capabilities.

Nunan et al. [9] used supervised ML methods like Naive Bayes and SVM to detect XSS attacks. Their total data set size 216054, and among them, 15366 are XSS attacks. They evaluate the algorithms based on accuracy, detection, and false alarm rates, etc. Their results show that compared to Naive Bayes, the SVM achieved the best performance. They selected the following features for classification of XSS attacks Obfuscation of code, the number of domains, URL Length, duplicate special characters, Schemes, etc.

Mereani and Howe [10] developed Random Forest, kNN, and SVM models to detect XSS malicious code, and they used labeled data in training. They trained using 2000 samples and used 13000 for testing. In their experiments, they reached accuracy up to 99.75%. They extracted Structural features contain a set of special characters in malicious JavaScript, and Behavioural Features includes function and commands used in malicious JavaScript code, a total of 59 features from both categories.

Rathore et al. [11] developed an ML method for Social networking services (SNSs) to detect XSS attacks. In their method, extracted Webpage features, URL features, and SNSs features from web pages and used this data to train models. Some of the features include domains in a URL, URL length, Iframes count, external link counts, and malicious JavaScript codes in SNSs webpage, etc. 1000 SNSs pages used to build a dataset for testing and used different classifiers in their testing. They achieved 97.2% accuracy in their tests.

Akaishi and Uda [12] used a combination of classifiers to detect XSS attacks in their research. Their data set contain balanced 10000 samples where attack data in URL format. They divided the attack sentence into words, co-occurrence, and frequency of words used in their classification. They used word2vec based model in their research to transform words into vectors, and used those vectors is classification algorithms. According to them, CNN and SVM are the best filters for real-world problems.

Mokbal et al. [13] proposed a Multilayer perceptron based model to detect XSS attacks. Their model achieved an accuracy of 99.32% in detecting attacks. Their dataset contains a total of 138569 samples, and among them, 38569 are attack samples. They extracted URL based, HTML based, and JavaScript-based features form content and used these features in training proposed models. Some of the features like URL length and special characters in URL, HTML tags, JavaScript events, etc.

Wang, Cai, and Wei [14] proposed a deep learning-based framework to detect malicious JavaScript. The structure contains logistic regression, deep learning method, and sparse random projection. They extracted features from JavaScript code by using Stacked denoising autoencoders (SdA). These features used to train SVM or logistic regression models. Classification of malicious code done by logistic regression. Their labeled dataset contains 14783 malicious JavaScript codes and 12320 benign samples. Their model achieved 94.9% accuracy.

III. Data Collection and Data Preprocessing

For this research, collected XSS vectors by using popular XSS tools like XSStrike, XSSER [15] and from different sources collected thousands of attack vectors. The dataset contains 154626 unique samples with labels. Half of this dataset is XSS attack vectors, and another half (77313) of the dataset is safe vectors. XSS attack vectors and Safe vectors are maintained at 128 characters, and longer sequences are split into 128 character chunks. Fig. 1 shows safe vector generator, by using this, generated safe vector samples. These randomly generated safe vectors are three types with length ranges from 40 to 126 those are, string with only uppercase or lowercase alphabets, strings with all alphabets and digits, and strings with all alphabets, digits and special characters. The below examples show different types of safe vectors:

1. kikDfuPLasVpSDqfKLMUTbyDAssjedEhphsOSPUnxO OHwDUkdHxLyJGPoMRIVERzJwuTVmbCwwYjVTtQ TfApxparHUUEEiidfUWBfJNUnVovFYNlBTJJ
2. aLcmHRaDMXwMmOmzQDhbEfeSYcZTRsPNkbjcoCa YauezgpthiPEvrUGfOXHGljqgZSDiArGKshBDvmcYm OdOYIpDsfbfGoPrwQXIkjltIIqImReZGbEVFwABEJZg Sn
3. BqAoxOrvaovydRv8QuQmQvoAk6hUbTaUFx18al7jYZ XBWvf1GWHIlbwgYd1qR2mx
4. x54fQrSJicA8f2KInEibadR3NrAVwkTgKdFn8WqBpqB KcufKJZ1zPpqybBPPQCu0LcWHjkRqvEgnJHUolgRLiZ ebe13wt7b6S1uY23cWkbleU7dzbKyQMysra18u
5. Y0P/U#Y_Dk#NNZ?p>B]6Ndb[&,:^iMI=~ts8Depf*C`aQ >!d[;p02LzJ,`5"hVCqAPXonVtrQ]L9`JBD=8L<c"Tl-?PASb7bs|/[.IXXMyQ:7av`q?m-@XV7"xm(
6. 2\{k@1\WMNXMi/3[1=mo#UHv5Da@-PzvG%*t(h-f\L25+{IU3#2Y_[msZ8h_^QP$@E4quPS~.~JddH"G3.+2)1~+svNQ.HPuCT5eKZVV*[Ej]*x5

The number of safe vectors generated depends on XSS attack vectors, to maintain the balance between XSS and safe samples of the dataset. This balanced dataset used to train and test the models. The below examples shows sample XSS attack vectors.

1. <img src="http://www.example.org/theerrornoimg.file" onerror=alert(" hi, here XSS Problem");>
2. <script\x20type="text/javascript">javascript:alert(19);</script>

To prepare input for models, converted the character sequence of XSS attacks, and Safe vectors into Unicode integer format, Fig. 2 shows sample data in Unicode format. The dataset is standardized by using sklearn's [16] StandardScaler function, Standardization of a dataset will improve the performance and accuracy of machine learning algorithms. Fig. 3 shows a sample data after standardization without the output column. The preparing process of dataset for model training shown in Fig. 4.

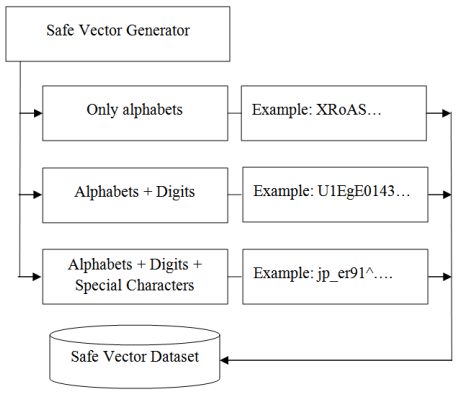

Fig. 1. Safe Vector Generator.

	0	1	2	3	4	5	6	...	122	123	124	125	126	127	128
86564	106	42	50	47	46	67	43	...	0	0	0	0	0	0	0
141242	51	36	91	126	65	59	55	...	0	0	0	0	0	0	0
74704	37	50	66	65	67	73	65	...	0	0	0	0	0	0	1
4185	60	65	47	43	47	79	110	...	0	0	0	0	0	0	1
47456	60	47	83	84	121	76	101	...	0	0	0	0	0	0	1
95944	101	67	118	77	90	99	117	...	0	0	0	0	0	0	0
61	60	83	84	89	76	69	62	...	0	0	0	0	0	0	1
147029	48	101	113	102	50	115	83	...	0	0	0	0	0	0	0
138884	71	72	107	119	85	66	89	...	0	0	0	0	0	0	0
81449	122	101	83	87	106	67	81	...	0	0	0	0	0	0	0

Fig. 2. Sample Data from the Dataset in Unicode Format.

	0	1	2	...	125	126	127
0	0.180603	-0.114946	-1.220119	...	-0.086933	-0.04473	-0.043924
1	-0.146358	-0.443607	-0.561638	...	-0.086933	-0.04473	-0.043924
2	0.088645	-0.402525	0.058109	...	-0.086933	-0.04473	-0.043924
3	0.241908	1.405115	1.336337	...	-0.086933	-0.04473	-0.043924
4	-0.146358	-1.881503	-1.491258	...	-0.086933	-0.04473	-0.043924
5	-0.146358	-1.306345	-0.019359	...	-0.086933	-0.04473	-0.043924
6	-0.146358	-0.443607	0.677856	...	-0.086933	-0.04473	-0.043924
7	0.201038	-1.347427	0.019375	...	-0.086933	-0.04473	-0.043924
8	-0.146358	-1.306345	1.220134	...	-0.086933	-0.04473	-0.043924
9	-0.146358	1.035371	1.258868	...	-0.086933	-0.04473	-0.043924

Fig. 3. Sample Data from the Dataset after Standardization.

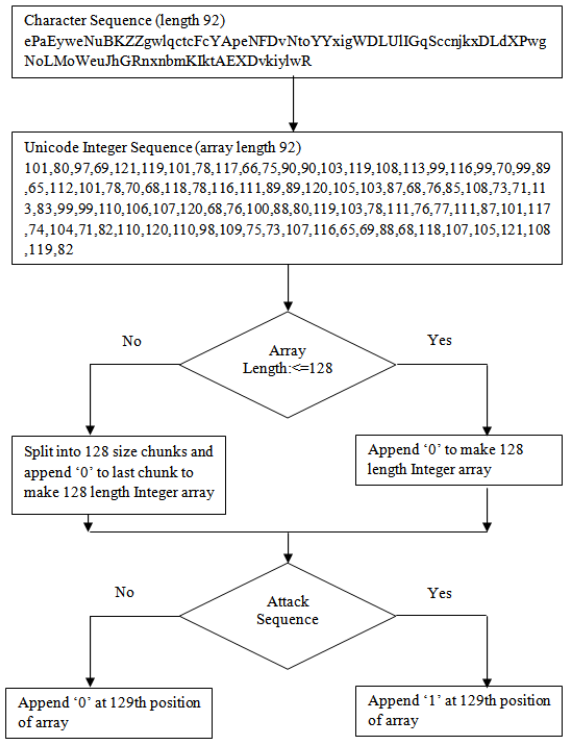

Fig. 4. Process of Preparing the Dataset.

IV. Implementation of Ensemble Methods

In this research supervised ensemble machine learning methods are used to detect XSS attacks. The ensemble learning methods [17] used are random forest classification, AdaBoost, bagging with SVM, Extra-Trees, gradient boosting, and histogram-based gradient boosting. These ensemble classification methods are effective in detecting XSS attacks compared to base models.

Google Colab [18] is used to build and test these models. The working environment includes Python 3.6.9, scikit-learn, TensorFlow 2.1.0 (includes Keras) [19], etc.

The dataset contains balanced unique 154626 samples, 77313 are Safe vectors and 77313 are XSS attacks. Total samples divided into 8:2 ratio for training (123700) and testing (30926) samples.

Fig. 5 shows the confusion matrix. The confusion matrix values are used to compare and evaluate the models.

Confusion matrix [20] used to calculate performance metrics of a model, by using the confusion matrix one can calculate the following values.

Recall = (TRUE POSITIVE) / (TRUE POSITIVE + FALSE NEGATIVE)

Precision = (TRUE POSITIVE) / (TRUE POSITIVE + FALSE POSITIVE)

F-measure = (2 x Recall x Precision) / (Recall + Precision)

Accuracy = (TP+TN)/(TP+TN+FP+FN)

A. Random Forest Classifier (RF)

Random forest classifier contains a collection of decision trees, and each decision tree fits on a subset of the dataset. Based on the output of all decision trees, the random forest classifier decides the final class of an input object. Table I shows the confusion matrix of the random forest model, and Table II shows the recall, precision, F-measure, and accuracy of the random forest model in detecting XSS attacks and safe vectors. Cross-validation scores of random forest classifier model are 0.99803, 0.99774, 0.99787, 0.99796, 0.99822 and the mean is 0.99796. In the random forest model, reached accuracy up to 0.99822.

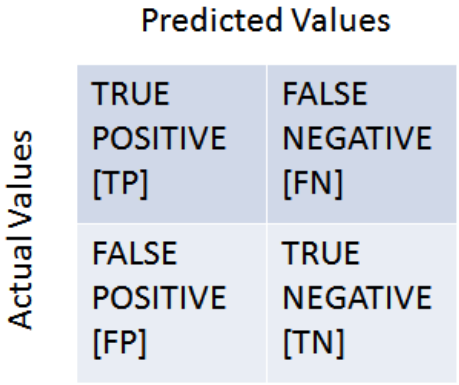

Fig. 5. Confusion Matrix.

TABLE I. RANDOM FOREST MODEL CONFUSION MATRIX

	Safe samples (Predicted)	XSS samples (Predicted)
Safe samples (Actual)	15463	0
XSS samples (Actual)	62	15401

TABLE II. RECALL, PRECISION, F-MEASURE, AND ACCURACY OF RANDOM FOREST MODEL

	Recall	Precision	F-measure
Safe samples	1.00000	0.99601	0.99800
XSS samples	0.99599	1.00000	0.99799
Accuracy = 0.99800			

B. AdaBoost Classifier (AB)

Boosting algorithms are used to reach high accuracy, AdaBoost (Adaptive Boosting) is a popular ensemble boosting algorithm works on decision trees. AdaBoost combines multiple low performing classifiers to get high performing classifier. In AdaBoost's every iteration, weak classifiers are tweaked (weighted data) based on the accuracy of previous training. The confusion matrix of the AdaBoost classifier model is shown in Table III, and Table IV shows the recall, precision, F-measure, and accuracy of the AdaBoost classifier model in detecting XSS attacks and safe vectors. Cross-validation scores of AdaBoost classifier model are 0.9977, 0.99793, 0.99735, 0.99764, 0.99832 and the mean is 0.99779. In the AdaBoost model, reached accuracy up to 0.99832.

C. Bagging Classifier with SVM (BC)

Bootstrap aggregating (or Bagging) is an ensemble method in machine learning. SVM is used as a base classifier for the bagging model. In bagging, the base classifiers are trained (fits) on a randomly selected subset data of the original dataset, and the final prediction depends on individual base classifiers predictions. The confusion matrix of the bagging model is shown in Table V, and Table VI shows the recall, precision, F-measure, and accuracy of the bagging model in detecting XSS attacks and safe vectors. Cross-validation scores of bagging classifier model are 0.98192, 0.98228, 0.98186, 0.98264, 0.98276 and the mean is 0.98229. In the bagging model, reached accuracy up to 0.98276.

D. Extra-Trees Classifier (ET)

Extra-Trees (Extremely Randomized Trees) method is an ensemble method similar to the random forest classifier. In Extra-Trees classifier, decision trees are constructed randomly in the forest, and these decision trees trained (fits) on subsets of data. The final prediction depends on all decision trees predictions. The confusion matrix values of the Extra-Trees classifier model is shown in Table VII, and Table VIII shows the recall, precision, F-measure, and the accuracy of the Extra-Trees classifier model in detecting XSS attacks and safe vectors. Cross-validation scores of Extra-Trees classifier model are 0.99049, 0.99088, 0.99069, 0.99192, 0.99175 and the mean is 0.99115. In the Extra-Trees classifier model, reached accuracy up to 0.99192.

E. Gradient Boosting Classifier (GB)

Gradient boosting classifier is an ensemble boosting algorithm, where a weak classifier is modified into a strong classifier. In the gradient boosting classifier, decision trees are base classifiers, and loss function is optimized while adding a new tree. Table IX shows the confusion matrix of the gradient boosting model, and Table X shows the recall, precision, F-measure, and the accuracy of the gradient boosting model in detecting XSS attacks and safe vectors. Cross-validation scores of gradient boosting classifier model are 0.99618, 0.99573, 0.99644, 0.99609, 0.99648 and the mean is 0.99618. In the gradient boosting model, reached accuracy up to 0.99648.

TABLE III. ADABOOST MODEL CONFUSION MATRIX

	Safe samples (Predicted)	XSS samples (Predicted)
Safe samples (Actual)	15440	23
XSS samples (Actual)	46	15417

TABLE IV. RECALL, PRECISION, F-MEASURE, AND ACCURACY OF ADABOOST MODEL

	Recall	Precision	F-measure
Safe samples	0.99851	0.99703	0.99777
XSS samples	0.99703	0.99851	0.99777
Accuracy = 0.99800			

TABLE V. BAGGING MODEL CONFUSION MATRIX

	Safe samples (Predicted)	XSS samples (Predicted)
Safe samples (Actual)	15243	220
XSS samples (Actual)	354	15109

TABLE VI. RECALL, PRECISION, F-MEASURE, AND ACCURACY OF BAGGING MODEL

	Recall	Precision	F-measure
Safe samples	0.98577	0.97730	0.98152
XSS samples	0.97711	0.98565	0.98136
Accuracy = 0.99800			

TABLE VII. EXTRA-TREES CLASSIFIER MODEL CONFUSION MATRIX

	Safe samples (Predicted)	XSS samples (Predicted)
Safe samples (Actual)	15463	0
XSS samples (Actual)	272	15191

TABLE VIII. RECALL, PRECISION, F-MEASURE, AND ACCURACY OF EXTRA-TREES CLASSIFIER MODEL

	Recall	Precision	F-measure
Safe samples	1.00000	0.98271	0.99128
XSS samples	0.98241	1.00000	0.99113
Accuracy = 0.99800			

TABLE IX. GRADIENT BOOSTING MODEL CONFUSION MATRIX

	Safe samples (Predicted)	XSS samples (Predicted)
Safe samples (Actual)	15406	57
XSS samples (Actual)	79	15384

TABLE X. RECALL, PRECISION, F-MEASURE, AND ACCURACY OF GRADIENT BOOSTING MODEL

	Recall	Precision	F-measure
Safe samples	0.99631	0.99490	0.99561
XSS samples	0.99489	0.99631	0.99560
Accuracy = 0.99800			

F. Histogram-based Gradient Boosting Classification (HGBC)

Histogram-based gradient boosting classification is an ensemble boosting algorithm, which is better compared to gradient boosting for large datasets. HGBC can handle missing values. In HGBC, decision trees are base classifiers. Table XI shows the confusion matrix of the HGBC model, and Table XII shows recall, precision, F-measure, and the accuracy of the HGBC model in detecting XSS attacks and safe vectors. Cross-validation scores of HGBC model are 0.99874, 0.99877, 0.99851, 0.99871, 0.9989 and the mean is 0.99873. In the HGBC model, reached accuracy up to 0.9989.

TABLE XI. HGBC MODEL CONFUSION MATRIX

	Safe samples (Predicted)	XSS samples (Predicted)
Safe samples (Actual)	15447	16
XSS samples (Actual)	32	15431

TABLE XII. RECALL, PRECISION, F-MEASURE, AND ACCURACY OF HGBC MODEL

	Recall	Precision	F-measure
Safe samples	0.99897	0.99793	0.99845
XSS samples	0.99793	0.99896	0.99845
Accuracy = 0.99800			

V. RESULTS AND DISCUSSION

This research evaluated the XSS detection rate in ensemble learning techniques. AdaBoost, bagging with SVM, Extra-Trees, gradient boosting, random forest classification, and histogram-based gradient boosting models are trained on a large labeled dataset and evaluated these methods performance based on their accuracy, recall, precision, and the F-measure. Table XIII compares the performance metrics of all models, and Table XIV compares the cross-validation scores of all models, and Fig. 6 shows the mean score of cross-validations of models. From the results, it is concluded that all ensemble methods performed well and reached an accuracy of more than 98% in all models.

Form all tested ensemble machine learning algorithms, the histogram-based gradient boosting classification model is the best performed model with the highest possible accuracy of 0.9989.

TABLE XIII. COMPARISON OF PERFORMANCE METRICS

Model	Recall		Precision		F-measure		Accuracy
	Safe	**XSS**	**Safe**	**XSS**	**Safe**	**XSS**	
RF	1.00000	0.99599	0.99601	1.00000	0.99800	0.99799	0.99800
AB	0.99851	0.99703	0.99703	0.99851	0.99777	0.99777	0.99777
BC	0.98577	0.97711	0.97730	0.98565	0.98152	0.98136	0.98144
ET	1.00000	0.98241	0.98271	1.00000	0.99128	0.99113	0.99120
GB	0.99631	0.99489	0.99490	0.99631	0.99561	0.99560	0.99560
HGBC	0.99897	0.99793	0.99793	0.99896	0.99845	0.99845	0.99845

TABLE XIV. COMPARISON OF CROSS-VALIDATION SCORES

Model	Fold 1 score	Fold 2 score	Fold 3 score	Fold 4 score	Fold 5 score	Mean score
RF	0.99803	0.99774	0.99787	0.99796	0.99822	0.99796
AB	0.9977	0.99793	0.99735	0.99764	0.99832	0.99779
BC	0.98192	0.98228	0.98186	0.98264	0.98276	0.98229
ET	0.99049	0.99088	0.99069	0.99192	0.99175	0.99115
GB	0.99618	0.99573	0.99644	0.99609	0.99648	0.99618
HGBC	0.99874	0.99877	0.99851	0.99871	0.9989	0.99873

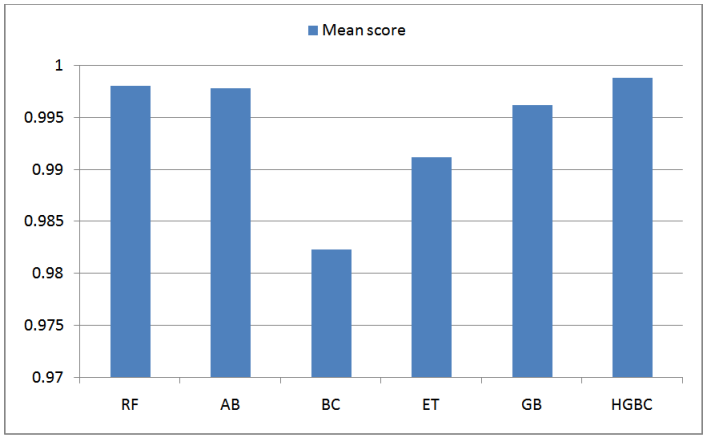

Fig. 6. Mean Score of Ensemble Learning Algorithms.

VI. CONCLUSION

We developed and analyzed supervised ensemble machine learning methods to detect XSS attacks in Web applications. Ensemble learning techniques are a collection of base classifiers, and these ensemble methods perform better than single classifiers. Existing solutions to detect XSS attacks by using machine learning methods have issues like single base classifiers, small datasets, and unbalanced datasets. We trained and evaluated proposed models on a large balanced dataset, and in this research, we detect XSS attacks in data submitted by the user. In this work, we evaluated the performance of random forest classification, AdaBoost, bagging with SVM,

Extra-Trees, gradient boosting, and histogram-based gradient boosting models in detecting XSS attacks and safe vectors. We compared the performance of models by using the confusion matrix metrics. The results show that all ensemble learning models performed exceptionally well in detecting XSS attacks and safe vectors. We reached the highest accuracy of 0.9989 in the histogram-based gradient boosting classification model.

VII. Future Work

In future, the work can be extend to detect other Web application attacks like SQL injection. The models can be tested by integrated into real world applications to detect attacks.

References

[1] Y. Zhou and P. Wang, "An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence," Comput. Secur., vol. 82, pp. 261–269, 2019.

[2] U. Sarmah, D. K. Bhattacharyya, and J. K. Kalita, "A survey of detection methods for XSS attacks," J. Netw. Comput. Appl., vol. 118, pp. 113–143, 2018.

[3] L. K. Shar and H. B. K. Tan, "Defending against cross-site scripting attacks," Computer (Long. Beach. Calif)., vol. 45, no. 3, pp. 55–62, 2011.

[4] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," Comput. Networks, vol. 166, p. 106960, 2020.

[5] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," Int. J. Syst. Assur. Eng. Manag., vol. 8, no. 1, pp. 512–530, 2017.

[6] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, "Current state of research on cross-site scripting (XSS) - A systematic literature review," Inf. Softw. Technol., vol. 58, no. July 2015, pp. 170–186, 2015.

[7] G. Shanmugasundaram, S. Ravivarman, and P. Thangavellu, "A study on removal techniques of Cross-Site Scripting from web applications," in 2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC), 2015, pp. 436–442.

[8] A. P. Aliga, A. M. John-Otumu, R. E. Imhanhahimi, and A. C. Akpe, "Cross Site Scripting Attacks in Web-Based Applications," J. Adv. Sci. Eng., vol. 1, no. 2, pp. 25–35, 2018.

[9] A. E. Nunan, E. Souto, E. M. Dos Santos, and E. Feitosa, "Automatic classification of cross-site scripting in web pages using document-based and URL-based features," in 2012 IEEE symposium on computers and communications (ISCC), 2012, pp. 702–707.

[10] F. A. Mereani and J. M. Howe, "Detecting cross-site scripting attacks using machine learning," in International Conference on Advanced Machine Learning Technologies and Applications, 2018, pp. 200–210.

[11] S. Rathore, P. K. Sharma, and J. H. Park, "XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs.," JIPS, vol. 13, no. 4, pp. 1014–1028, 2017.

[12] S. Akaishi and R. Uda, "Classification of XSS Attacks by Machine Learning with Frequency of Appearance and Co-occurrence," in 2019 53rd Annual Conference on Information Sciences and Systems (CISS), 2019, pp. 1–6.

[13] F. M. M. Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar, and W. Xiaoxi, "MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique," IEEE Access, vol. 7, pp. 100567–100580, 2019.

[14] Y. Wang, W. Cai, and P. Wei, "A deep learning approach for detecting malicious JavaScript code," Secur. Commun. networks, vol. 9, no. 11, pp. 1520–1534, 2016.

[15] I. M. Babincev and D. V Vuletić, "Web application security analysis using the kali Linux operating system," Vojnoteh. Glas., vol. 64, no. 2, pp. 513–531, 2016.

[16] S. Raschka and V. Mirjalili, Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd, 2019.

[17] R. Polikar, "Ensemble learning," in Ensemble machine learning, Springer, 2012, pp. 1–34.

[18] E. Bisong, "Google Colaboratory," in Building Machine Learning and Deep Learning Models on Google Cloud Platform, Springer, 2019, pp. 59–64.

[19] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[20] V. Labatut and H. Cherifi, "Accuracy measures for the comparison of classifiers," arXiv Prepr. arXiv1207.3790, 2012.