ERDNS: Ensemble of Random Forest, Decision Tree, and Naive Bayes Kernel Through Stacking for Efficient Cross Site Scripting Attack Classification

A. Niranjan[1(✉)], K. M. Akshobhya[2], Arun Singh Chouhan[3], and Praveen Tumuluru[4]

[1] Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad 500090, India
[2] SAP Labs Pvt Ltd., Bangalore, India
[3] Malla Reddy University, Hyderabad, India
[4] Koneru Lakshmaiah Education Foundation, Vijayawada, India

Abstract. Cross-Site Scripting (XSS) is a form of client-side code injection attack in which an attacker attempts to execute malicious scripts in the victim's web browser by embedding dangerous code in a legitimate web page or application. During such an attack, the attacker impersonates a victim user and performs any behavior that the user is capable of, as well as accessing any of his data. If the victim user has privileged access to the application, the attacker can take complete control of the app's features and data. XSS attacks are most common on message boards, forums, and websites that accept comments. According to a study from CDN and cloud security provider CDNetworks, attacks on web applications increased by 800% in the first six months of 2020 compared to the same time frame last year. Our approach involves the application of Machine Learning Algorithms for efficient classification of XSS attacks. The MMR and SDMR Algorithms aid in the selection of significant features from a data set without compromising classification results. The proposed ERDNS framework ensures better Detection Accuracy and Precision with least Classification Error values among all the available models.

Keywords: Cross-site scripting attack classification · Ensemble learning · Hybrid feature selection · Managing XSS attacks

1 Introduction

As per the recent Research Reports, XSS constitutes close to 40% of the total cyber attacks. According to a study from CDN and cloud security provider CDNetworks, attacks on web applications increased by 800% in the first six months of 2020 compared to the same time frame last year. Cross-site Scripting

K. R. Venugopal et al. (Eds.): ICInPro 2021, CCIS 1483, pp. 353–365, 2021.
https://doi.org/10.1007/978-3-030-91244-4_28

is a form of client-side code injection attack in which an attacker attempts to execute malicious scripts in the victim's web browser by embedding malicious code in a legitimate web page or application (XSS). During such an attack, the attacker impersonates a victim user and performs any behavior that the user is capable of, as well as accessing any of his data. If the victim user has privileged access to the application, the attacker can take complete control of the app's features and data. XSS attacks are most common on message boards, forums, and websites that accept comments. Based on the origin of malicious script that is responsible for the attack, we have three XSS attack types namely, a) Reflected XSS, b) Stored XSS, and c) DOM-based XSS.

Effective XSS Attack Classification at an earlier stage only can facilitate the XSS Detection process. That is, when a sample could be accurately classified as attack or benign, the XSS Detection Mechanism could stop an XSS attack on its very onset. The application of Machine Learning techniques has been proven to be an effective tool for any classification problem. Use of Hybrid models involving Multi-layered classification schemes is not a new idea in research. The current study looks at and explores a few hybrid models that use multi-layered classification to predict XSS samples effectively. We investigate two Ensemble Schemes, Stacking and Voting, for implementing the Ensemble model, and the most efficient among them is proposed. Until classifying the data set, we investigate various Feature Selection Algorithms to compute the Rank of each feature present in it. Mean and Standard Deviation values for the Ranks generated by various Feature Selection Algorithms are then computed. Finally, a Mean of Mean of Ranks (MMR) and Standard Deviation of Mean of Ranks (SDMR) are calculated for each feature. All features with a Mean and Standard Deviation less than the final MMR and SDMR are removed, leaving behind two Feature subsets with most significant and meaningful features. In addition, these subsets of Features are intersected to determine significant features for classification. As a result, the Time Complexity for classification is greatly reduced. Various built in classifiers are run on the feature subsets obtained after MMR and SDMR and their Intersection. Top three classifiers in terms of chosen performance metrics are identified.

Such classifiers are then subjected to ensemble models such as Stacking and Voting. The performance metrics are again recorded for the two ensemble models. The top performing model between the two, involving the top three Classifiers is finally chosen as our proposed model for efficient classification of XSS attack samples. When a hybrid model combining Random Forest, Decision Tree, and Naive Bayes Kernel is ensembled using Stacking, we expect results that are better in terms of the chosen metrics such as Prediction Accuracy, Classification Error, and Precision. We name the technique as ERDNS (Ensemble of Random Forest, Decision Tree, and Naive Bayes Kernel through Stacking). To demonstrate the efficacy of the proposed ERDNS process, extensive tests were performed on publicly accessible XSS data sets by Fawaz Mokbal et al. [4], namely XSSdataset1engineered.csv and XSSdataset2engineered.csv, which con-

tain 101000 and 140660 instances, respectively. ERDNS outperforms the existing Models, according to the experimental results obtained on the data sets.

A type of Decision Trees that is based on a randomly selected subset of attributes is referred to as a Random Tree. A Decision Tree basically is a set of nodes and branches. A test on an attribute is represented by a node, while the result of a test is represented by a branch. The external nodes (leaves) reflect the final decision. The route from the root to the leaf creates a classification law. A Random Forest is a classifier that is made up of many such individual Random trees. Each Random Tree in a Random Forest is constructed by using Bagging and Feature Randomness to ensure that there is no connection between the trees. Since the Random Forest bases its final prediction on the highest number of votes received for a particular prediction, it is more accurate than any Random Tree.

Naive Bayes is a supervised learning algorithm for solving classification problems that is based on the Bayes theorem. It is a simple and effective classification algorithm that aids in the development of fast machine learning models capable of making quick predictions. It's a probabilistic classifier, which means it makes predictions based on an object's likelihood. The Naive Bayes classifier has the advantage of providing very few training data to estimate the mean and variances of the variables used for classification. Since independent variables are assumed, only the variances of the variables for each label, not the entire covariance matrix, must be calculated. The Naive Bayes Kernel operator, unlike the Naive Bayes operator, can be applied to numerical attributes. In non-parametric estimation techniques, a kernel is a weighting function. The density functions of random variables, as well as the conditional expectation of a random variable, are estimated using kernels. Kernel density estimators belong to the non-parametric class of estimators. Such estimators have no fixed structure and depend on all data points to achieve an approximation, unlike parametric estimators, which have a fixed functional type (structure) and the parameters of this function are the only information we need to store.

Few of the popularly used Ensemble Techniques are Voting, Stacking, StackingC, Bagging, Boosting and Grading. Voting entails creating a variety of submodels and considering the predictions of each of the submodels to decide what the final outcome should be. Stacking involves separate and independent training of heterogeneous learning algorithms on the data, and using their results as additional inputs to the combiner algorithm for the final training. StackingC, a variant of Stacking on the other hand, uses Linear Regression as the Meta Classifier. Linear regression is a method for converting a set of numeric values (x) into an estimated output value (y). Grading is one of the meta classification methods, and it involves finding and correcting any incorrect predictions. Instead of using the predictions of base classifiers as metalevel attributes as in Stacking, Grading uses graded predictions (correct or incorrect).

This article's contributions can be summarised as follows:

- The evaluation of a novel general-purpose classifier architecture based on the Hybrid approach on two data sets is presented.
- MMR and SDMR Algorithms are proposed for efficient Feature Selection.
- Performance Metrics considered and the results obtained for the individual Classifiers and also the Ensemble Models are presented.

The following is how the rest of the article is structured. Section 2 discusses related work in this area, while Sect. 3 introduces the proposed ERDNS structure. The investigative approach is detailed in Sect. 4, and the experimental findings are presented in Sect. 5, along with reviews and discussions. The last section of this paper is the conclusion.

2 Related Work

The focus of the case study presented in [1] is on current methodologies and methods for detecting and mitigating XSS attacks and vulnerabilities. It discusses both client and server-side detection methods, as well as static and dynamic approaches, that have been proposed to detect the attack thus far. It also covers the XSS Defense methods and techniques that are used to secure both the Client and Server. The aim of this paper is to take a close look at how to mitigate XSS attacks using various detection and defence techniques.

The paper proposed by X Zhang et al. [2] discusses about MCTS-T adversarial example generation algorithm that enables the generation model to provide a reward value. The value thus obtained reflects the probability of generative examples bypassing the detector. The authors further optimize their XSS detection model with GAN(Generative Adversarial Network) to enhance its ability to defend against adversarial examples. The disadvantage of MCTS-T algorithm is that it can only generate adversarial examples of XSS traffic at present.

Various Supervised Machine Learning Algorithms such as Support Vector Machine, Decision Tree, Naive Bayes and Un-Supervised Algorithms such as K-Means and Association Rule algorithms along with a Deep Learning Technique called Long Short-Term Memory Algorithms are explored by XiaoLong Chen et al. [3] who have listed out the advantages and disadvantages of these techniques in their work.

The authors in their survey [6] explore the background of XSS attacks, classify the derived types of XSS attacks, the role of cookies and a bunch of tools and methods that can be used for the detection and mitigation of XSS Attacks in detail.

To improve the XSS detection in a low-resource data setting, Mokbal FMM et al. [7] propose a conditional Wasserstein Generative Adversarial Network with a gradient penalty. Their method creates synthetic minority class samples with the same distribution as actual XSS attack scenarios. They use augmented data to train a new boosting model, which is then tested on a real-world data set. Their model produces consistent and accurate samples.

In the work discussed in [8] two approaches are proposed. The first approach employs insecure flow monitoring to filter malicious scripting code inserted into dynamic web pages while the second creates and validates trusted remark for detecting any suspicious activity in static web pages. Finally, the user is presented with the filtered and updated webpage. They evaluate their prototype by testing with a collection of real-world web applications to see whether it could detect and mitigate XSS attacks.

The aim of study carried out in the PG thesis by [9] is to see whether an XSS vulnerability scanner can be rendered more versatile than what is currently available. The aim is to see how reflected parameters can be identified and whether a different approach can be used to enhance XSS vulnerability detection. Mantis methodology basically focuses on the set of characters that could lead to XSS manipulation.

3 ERDNS: Efficient XSS Attack Classification Framework

The Preprocessing and Classification phases will be the focus of the majority of Machine Learning applications. It is a proven fact that a classifier doesn't need all features to make the final prediction. So picking up of only significant features from the data set becomes an important step in the preprocessing phase. We suggest MMR and SDMR algorithms for the selection of Features by leveraging the various advantages of built in Ranking Algorithms that are based on weights. The classification process entails applying the proposed ERDNS framework to the resultant Feature subset through tenfold cross validation using Random Forest, Decision Tree, and Naive Bayes Kernel ensembled via Stacking and various other techniques. The ERDNS framework is depicted in Fig. 1. The data sets XSSdatasetengineered1.csv and XSSdatasetengineered2.csv from Fawaz Mokbal et al. [4] that contain publicly accessible XSS samples collections are used. The experimentation is carried out on these files containing 101000 and 140660 instances, respectively.

The proposed MMR (Mean of Mean of Ranks) Feature Selection algorithm [5] as stated in Algorithm 1 and SDMR (Standard Deviation of Mean of Ranks) as listed in Algorithm 2 employ many existing Feature Selection Algorithms for the determination of Ranks of every Feature. The Mean and Standard Deviations are calculated using the rank values provided by various Algorithms for a Feature. All features with Mean of Ranks less than the Overall Mean value and less than the final Standard Deviation of Mean of Ranks are discarded, leaving only valid and important feature subsets for classification. For the determination of the Ranks R_i, Algorithms such as InfoGain, InfoGainRatio, Rule method, Deviation method, Correlation method, Chi-squared statistics, Gini-Index and Principal Component Analysis are employed. The MMR for XSSdatasetengineered1 is 0.05 and that of XSSdatasetengineered2 is 0.12. The number of Features selected from XSSdatasetengineered1 using MMR is 15 out of 67 while features selected from XSSdatasetengineered2 is 28 out of 77 features. The SDMR for XSSdatasetengineered1 is 0.12 and that of XSSdatasetengineered2 is

0.10. The number of features selected from XSSdatasetengineered1 using SDMR is 14 out of 67 while features selected from XSSdatasetengineered2 is 30 out of 77 features. To further determine the most significant features selected from both the subsets of features obtained from MMR and SDMR, it was decided to perform an intersection operation on these subsets. After this operation, 14 Features were picked from XSSdatasetengineered1 and 26 from XSSdatasetengineered2 as listed in Table 1.

Algorithm 1: MMR (Mean of Mean of Ranks) Feature Selection

1: Input: Data set D having n Features
2: Output: Subset of Most Significant Features (F_1) Present in D
3: **for** every feature $f_i \in D$ **do**
4: Calculate Ranks R_i using various Feature Weight Calculation Techniques
5: **end for**
6: **for** each feature $f_i \in D$ **do**
7: Determine Sum ($\sum R_i$) and Mean (m R_i) of Ranks
8: $\sum R_i = R_1 + R_2 + ... + R_n$ and $mR_i = \sum R_i / n$
9: **end for**
10: Calculate the Mean of Mean of Ranks $M(mR_i)$
11: Discard all $f_i \in D < M(mR_i)$
12: return F_1

Algorithm 2: SDMR (Standard Deviation of Mean of Ranks) Feature Selection

1: Input: Data set D having n Features
2: Output:The subset of Most Significant Features (F_2) Present in D
3: **for** each feature $f_i \in D$ **do**
4: Calculate Ranks R_i using various Rank Based Feature Selection Techniques
5: **end for**
6: **for** each feature $f_i \in D$ **do**
7: Determine Mean (mR_i) of the feature
8: **end for**
9: Determine Standard Deviation of Mean of Ranks σ_f (mR_i)
10: Discard all $f_i \in D < \sigma_f(mR_i)$
11: return F_2

Algorithm 3: Ensemble of Random Forest, Decision Tree, and Naive Bayes through Voting for Efficient Cross Site Scripting Attack Classification (ERDNS)

1: Input: Features subsets F_1 and F_2 obtained from Algorithm 1 and 2 respectively
2: Output: Classification Results
3. Perform the intersection ($\cap$) of the feature subsets F_1 and F_2
3: Subject the result to an Ensemble of Random Forest, Decision Tree, and Naive Bayes through Stacking
4: The model is subjected for a ten-fold cross validation to estimate the performance metrics of classification

The proposed MMR and SDMR for Feature Selection are presented as Algorithm 1, and Algorithm 2 respectively. Algorithm 3 presents the ERDNS framewoek for efficient XSS Attack Classification. Table 1 lists the Final feature subset obtained after the Intersection of Feature subsets generated by MMR and SDMR.

Figure 1 sketches the System Model of the proposed ERDNS for XSS Attack Classification.

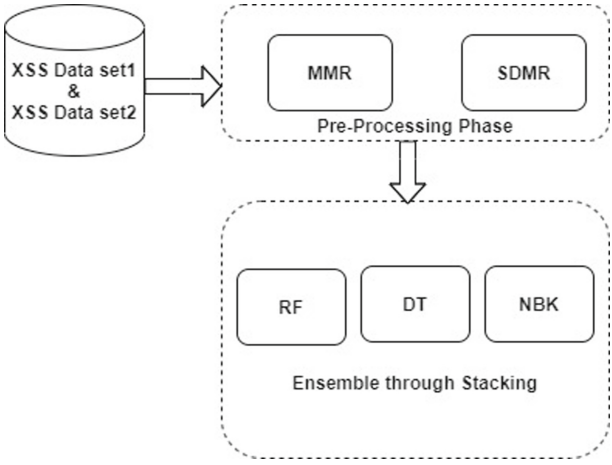


Fig. 1. XSS Attack Classification Model

4 Investigation Methodology

The data sets from publicly available XSS samples collections by Fawaz Mokbal et al. [4] namely XSSdataset1engineered.csv and XSSdataset2engineered.csv consisting of 101000 and 140660 instances respectively are used for the experimentation purpose. The proposed MMR (Mean of Mean of Ranks) as listed in Algorithm 1 and SDMR (Standard Deviation of Mean of Ranks) as listed in

Table 1. Final Feature subset after Intersection operation of MMR and SDMR

Data set-1	Data set-2
url_length	DestinationPort
url_special_characters	FlowDuration
url_tag_script	TotalLengthofFwdPackets
url_tag_iframe	FwdPacketLengthMax
url_attr_src	FwdPacketLengthMean
url_event_onload	BwdPacketLengthStd
url_event_onmouseover	FlowIATMean
url_cookie	FlowIATStd
url_number_domain	FlowIATMax
html_event_onblur	FwdIATTotal
js_file	FwdIATMean
js_pseudo_protocol	FwdIATStd
js_method_alert	FwdIATMax
js_method_eval	BwdIATTotal
	MaxPacketLength
	PacketLengthMean
	FINFlagCount
	PSHFlagCount
	ACKFlagCount
	URGFlagCount
	AveragePacketSize
	AvgFwdSegmentSize
	FwdAvgPackets_Bulk
	SubflowFwdBytes
	Init_Win_bytes_forward
	Init_Win_bytes_backward

Algorithm 2 employ several existing Feature Selection Algorithms for computing the ranks of all the features that are present in the data set. Finally, an overall Mean and Standard Deviation for the Mean of Ranks for all features are computed. All features with Mean of Ranks less than the Overall Mean value and less than the final Standard Deviation of Mean of Ranks are discarded, leaving only valid and important feature subsets for classification. For the determination of the Ranks Ri Algorithms such as InfoGain, InfoGainRatio, Rule method, Deviation method, Correlation method, Chi-squared statistics, GiniIndex and Principal Component Analysis are employed. The MMR for XSSdatasetengineered1 is 0.05 and that of XSSdatasetengineered2 is 0.12. The number of Fea-

tures selected from XSSdatasetengineered1 using MMR is 15 out of 67 (77.6% Reduction)while features selected from XSSdatasetengineered2 is 28 out of 77 features(63.63% Reduction). The SDMR for XSSdatasetengineered1 is 0.12 and that of XSSdatasetengineered2 is 0.10. The number of features selected from XSSdatasetengineered1 using SDMR is 14 out of 67 (79.1% Reduction)while features selected from XSSdatasetengineered2 is 30 out of 77 features(61% Reduction). To further determine the most significant features selected from both the subsets of features obtained from MMR and SDMR, it was decided to perform an intersection operation on these subsets. After this operation, 14 Features were picked from XSSdatasetengineered1(79.1% Reduction) and 26 from XSSdatasetengineered2 (66.23% Reduction) as listed in Table 1. The performance metrics were recorded after a thorough ten-fold cross validation. A ten-fold cross validation technique typically involves dividing the data set into ten sections, training the model with the nine parts of the data while using the excluded section as the test set, and repeating the process for ten iterations, using each unused test set during each round. Prediction Accuracy, Classification Error, Recall and Precision values are used as the Performance Metrics for determining the efficiency of the classifiers. Two Ensemble approaches namely Voting and Stacking are tried to enhance the Performance of Classification on the top performing classifiers.

If a classifier has a higher true positive rate and a lower false positive rate, it is considered efficient. There are seven efficiency criteria in a traditional classification methodology, which we will discuss below. N_{Ben} is the number of benign samples in the XSS data set, while N_{XSS} is the number of XSS samples. $N_{Ben}rightarrow_{Ben}$ is the number of benign samples correctly identified as benign (TP). True Negative (TN) is the number of XSS samples correctly defined as XSS. $N_{XSS}\rightarrow_{XSS}$ is the symbol for it. False Positive (FP) is a metric for Benign samples that have been mislabeled as XSS. $N_{Ben}\rightarrow_{XSS}$ is the abbreviation, and False Negative (FN) is a metric for XSS incidents mis-classified as Benign. $N_{XSS}\rightarrow_{Ben}$ is the symbol for it. The Detection Rate (DR) refers to the percentage of XSS samples that are detected and correctly identified as XSS.

$$TP = \frac{N_{\text{Ben}\rightarrow\text{Ben}}}{(N_{\text{Ben}\rightarrow\text{Ben}} + N_{\text{XSS}\rightarrow\text{Ben}})} \tag{1}$$

The percentage of benign samples wrongly classified as XSS is known as the False Positive Rate (FPR).

$$FPR = \frac{N_{\text{Ben}\rightarrow\text{XSS}}}{(N_{\text{XSS}\rightarrow\text{XSS}} + N_{\text{Ben}\rightarrow\text{Ben}})} \tag{2}$$

The False Negative Rate (FNR) is the percentage of XSS samples that are wrongly classified as benign when they are not.

$$FNR = \frac{N_{\text{XSS}\rightarrow\text{Ben}}}{(N_{\text{XSS}\rightarrow\text{XSS}} + N_{\text{XSS}\rightarrow\text{Ben}})} \tag{3}$$

The True Negative Rate (TNR) is the percentage of Benign samples that are correctly classified as Benign out of all available Benign samples.

$$TNR = \frac{N_{\text{Ben}\rightarrow\text{Ben}}}{(N_{\text{Ben}\rightarrow\text{Ben}} + N_{\text{Ben}\rightarrow\text{XSS}})} \tag{4}$$

The total number of XSS and Benign samples correctly identified in comparison to the total number of all available instances is referred to as Prediction Accuracy (PA).

$$PA = \frac{(N_{\text{Ben}\rightarrow\text{Ben}} + N_{\text{XSS}\rightarrow\text{XSS}})}{(N_{\text{Ben}\rightarrow\text{Ben}} + N_{\text{XSS}\rightarrow\text{XSS}} + N_{\text{Ben}\rightarrow\text{XSS}} + N_{\text{XSS}\rightarrow\text{Ben}})} \tag{5}$$

The number of true positives divided by the total number of instances listed as positive is Precision.

$$PREC = \frac{N_{\text{XSS}\rightarrow\text{XSS}}}{(N_{\text{XSS}\rightarrow\text{XSS}} + N_{\text{Ben}\rightarrow\text{XSS}})} \tag{6}$$

Recall is defined as the number of true positives divided by the total number of instances that truly belong to the positive class.

$$REC = \frac{N_{\text{XSS}\rightarrow\text{XSS}}}{(N_{\text{XSS}\rightarrow\text{XSS}} + N_{\text{XSS}\rightarrow\text{Ben}})} \tag{7}$$

Figure 5 depicts the performance comparison of various Classifiers in terms of Precision while the other metrics such as Classification Error, Recall and Prediction Accuracy are depicted in Figs. 2, 3, and 4 respectively. From Fig. 2 we can make out that the Classification Error rate is the least in case of Random Forest and Decision Tree while slightly larger in Naive Bayes Kernel. However, these Classifiers record better Recall (as seen in Fig. 3), Prediction Accuracy

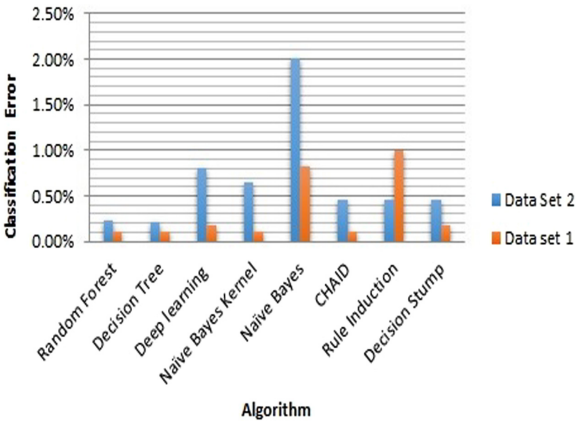


Fig. 2. Classification error comparison of classifiers

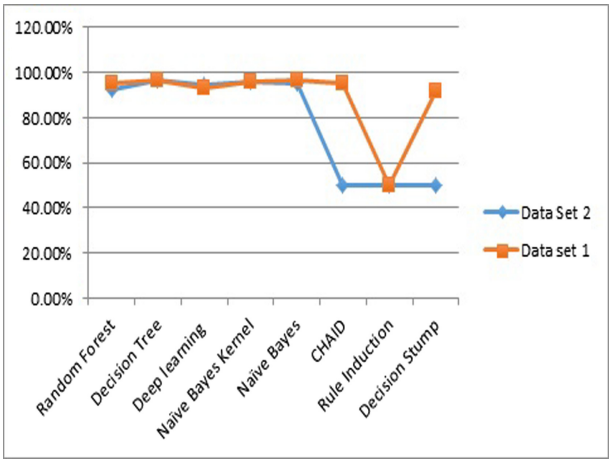


Fig. 3. Recall comparison of classifiers

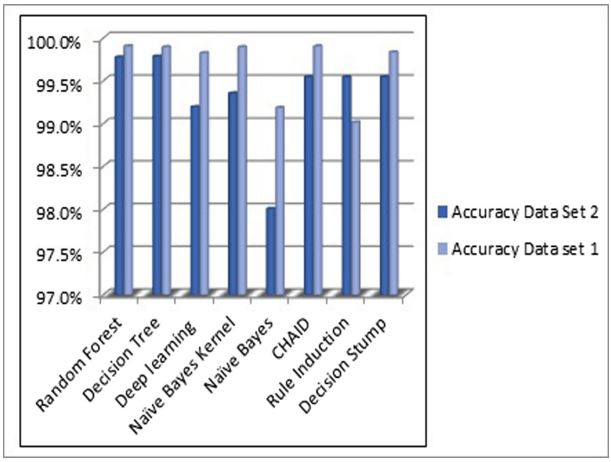
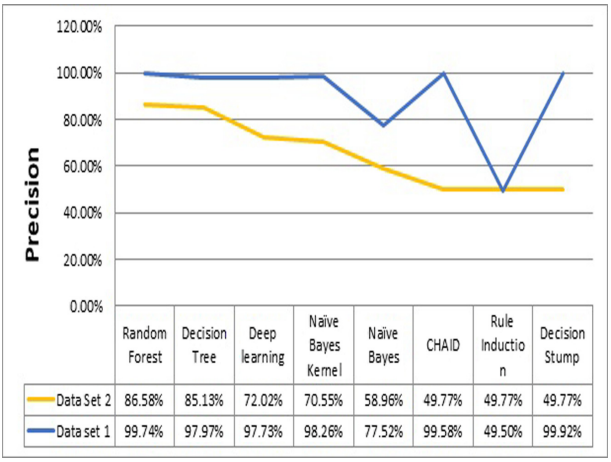


Fig. 4. Prediction accuracy comparison of classifiers

(Fig. 4), and Precision (Fig. 5). Expecting still better results it was decided to combine the three Classifiers using two ensemble models namely Stacking and Voting. Figure 6 indicates the Comparative results of the two Ensemble Models. It can be observed from Fig. 6 that Stacking records lesser Classification Error, with slightly better Accuracy and Precision than the Voting Model. The top performing Ensemble model that is Stacking, involving top three Classifiers Random Forest, Decision Tree, and Naive Bayes Kernel is therefore chosen as our proposed model for efficient classification of XSS attack samples.



	Random Forest	Decision Tree	Deep learning	Naïve Bayes Kernel	Naïve Bayes	CHAID	Rule Induction	Decision Stump
Data Set 2	86.58%	85.13%	72.02%	70.55%	58.96%	49.77%	49.77%	49.77%
Data set 1	99.74%	97.97%	97.73%	98.26%	77.52%	99.58%	49.50%	99.92%

Fig. 5. Precision comparison of classifiers

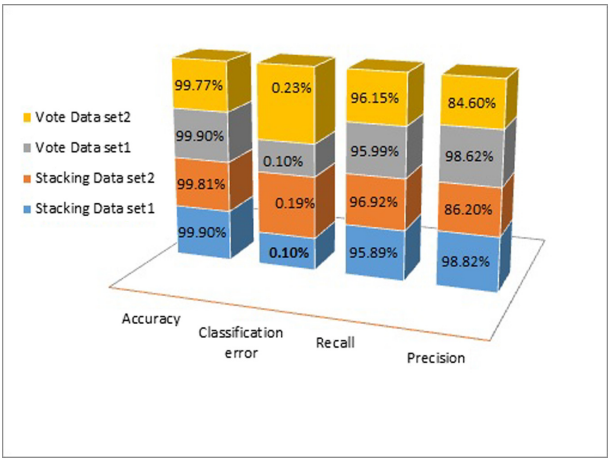


Fig. 6. Comparison of ensemble models

5 Conclusion

The proposed MMR and SDMR Feature Selection Algorithms select 14 Features from XSSdatasetengineered1 out of a total 67 and 26 from XSSdatasetengineered2 out of 77 after the Intersection operation on the Feature subsets is performed during the Pre-Processing phase. This amounts to a dimensionality reduction of 79.1% and 66.23% in the first and second data sets respectively. The resultant Feature subset is subjected to various Classier algorithms and a tenfold cross validation is performed before recording the performance metrics. Two Ensemble approaches namely Voting and Stacking are tried to determine

the best Classification Model on the top performing classifiers. The ERDNS model must be checked on real-time data sets and compared with other existing models to see if it can be improved further. The time it takes to complete the whole process must be minimized so that XSS samples can be classified as soon as they are detected.

References

1. Vijayalakshmi, K., Syed Mohamed, E.: Case study: extenuation of XSS attacks through various detecting and defending techniques. Int. J. Innov. Res. Comput. Sci. Technol. (IJIRCST) **9**(1), 1736 (2021). https://doi.org/10.1080/19361610.2020.1735283
2. Zhang, X., Zhou, Y., Pei, S., Zhuge, J., Chen, J.: Adversarial examples detection for XSS attacks based on generative adversarial networks. IEEE Access **8**, 10989–10996 (2020). https://doi.org/10.1109/ACCESS.2020.2965184
3. Chen, X.L., Li, M., Jiang, Yu., Sun, Y.: A comparison of machine learning algorithms for detecting XSS attacks. In: Sun, X., Pan, Z., Bertino, E. (eds.) ICAIS 2019. LNCS, vol. 11635, pp. 214–224. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24268-8_20
4. Mokbal, F.M.M.: Cross-Site Scripting Attack (XSS) dataset (2020). https://doi.org/10.6084/m9.figshare.13046138.v4
5. Niranjan, A., Nitish, A., Deepa Shenoy, P., Venugopal, K.R.: Security in data mining-a comprehensive survey. Glob. J. Comput. Sci. Technol. **16**(5), 51–72 (2017). https://computerresearch.org/index.php/computer/article/download/1464/1451
6. Rodríguez, G.E., Torres, J.G., Flores, P., Benavides, D.E.: Cross-site scripting (XSS) attacks and mitigation: a survey. Comput. Netw. **166**, 1–21 (2020). https://doi.org/10.1016/j.comnet.2019.106960
7. Mokbal, F.M.M., Wang, D., Wang, X., Fu, L.: Data augmentation-based conditional wasserstein generative adversarial network-gradient penalty for XSS attack detection system. PeerJ Comput. Sci. **6**, e328 (2020). https://doi.org/10.7717/peerj-cs.328
8. Gupta, B.B., Chaudhary, P., Gupta, S.: Designing a XSS defensive framework for web servers deployed in the existing smart city infrastructure. J. Organ. End User Comput. **32**(4) (2020). https://doi.org/10.4018/JOEUC.2020100105
9. Liljebjörn, J., Broman, H.: Mantis The Black-Box Scanner: Finding XSS Vulnerabilities through Parse Errors, Retrieved from Dissertation: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-19566