Düzce University
Journal of Science & Technology

Research Article

A Stacking Ensemble Learning Approach for Intrusion Detection System

Murat UÇAR [a], Emine UÇAR [a,*] Mürsel Ozan İNCETAŞ[b]

[a] *Department of Management Information Systems, Faculty of Business and Management Sciences, İskenderun Technical University, Hatay, TURKEY*
[b] *Department of Computer Technologies, ALTSO Vocational School, Alanya Alaaddin Keykubat University, Antalya, TURKEY*
** Corresponding author's e-mail address: emine.ucar@iste.edu.tr*
DOI:10.29130/dubited.737211

ABSTRACT

Intrusion detection systems (IDSs) have received great interest in computer science, along with increased network productivity and security threats. The purpose of this study is to determine whether the incoming network traffic is normal or an attack based on 41 features in the NSL-KDD dataset. In this paper, the performance of a stacking technique for network intrusion detection was analysed. Stacking technique is an ensemble approach which is used for combining various classification methods to produce a preferable classifier. Stacking models were trained on the NSLKDD training dataset and evaluated on the NSLKDDTest+ and NSLKDDTest21 test datasets. In the stacking technique, four different algorithms were used as base learners and an algorithm was used as a stacking meta learner. Logistic Regression (LR), Decision Trees (DT), Artificial Neural Networks (ANN), and K Nearest Neighbor (KNN) are the base learner models and Support Vector Machine (SVM) model is the meta learner. The proposed models were evaluated using accuracy rate and other performance metrics of classification. Experimental results showed that stacking significantly improved the performance of intrusion detection systems. The ensemble classifier (DT-LR-ANN + SVM) model achieved the best accuracy results with 90.57% in the NSLKDDTest + dataset and 84.32% in the NSLKDDTest21 dataset.

Keywords:*Intrusion detection, Stacking, Security, Ensemble learning*

Saldırı Tespit Sistemi İçin İstifleme Topluluk Öğrenme Yaklaşımı

ÖZ

Saldırı tespit sistemleri (STS'ler), artan ağ verimliliği ve güvenlik tehditlerinin yanı sıra bilgisayar bilimlerinde de büyük ilgi görmüştür. Bu çalışmanın amacı, NSL-KDD veri kümesindeki 41 özelliğe bağlı olarak gelen ağ trafiğinin, normal veya saldırı olup olmadığını belirlemektir. Bu yazıda, ağ izinsiz giriş tespiti için bir istifleme tekniğinin performansı analiz edilmiştir. İstifleme tekniği, tercih edilebilir bir sınıflandırıcı üretmek için çeşitli sınıflandırma yöntemlerini birleştirerek kullanılan bir topluluk yaklaşımıdır. İstifleme modelleri NSLKDD eğitim veri seti üzerinde eğitilmiş ve NSLKDDTest+ ve NSLKDDTest21 test veri setleri üzerinde test edilmiştir. İstifleme tekniğinde temel öğrenenler olarak dört farklı algoritma ve istifleme meta öğrenicisi olarak bir algoritma kullanılmıştır. Lojistik Regresyon (LR), Karar Ağaçları (KA), Yapay Sinir Ağları (YSA) ve K En Yakın Komşu (KEYK) temel öğrenici modelleridir ve Destek Vektör Makinesi (DVM) modeli meta öğrenicidir. Önerilen modeller, doğruluk oranı ve sınıflandırmanın diğer performans metrikleri kullanılarak değerlendirilmiştir. Deney sonuçları istiflemenin saldırı tespit sisteminin performansını önemli ölçüde artırdığını göstermiştir. Topluluk sınıflandırıcısı (KA-LR-YSA + DVM) modeli, NSLKDDTest+ veri kümesinde %90.57 ve NSLKDDTest21 veri kümesinde %84.32 ile en iyi sonuçlara ulaşmıştır.

Received: 14/05/2020, Revised: 30/04/2021, Accepted: 06/05/2021

Anahtar Kelimeler: *Saldırı tespiti, İstifleme, Güvenlik, Topluluk öğrenimi*

I. INTRODUCTION

Over the past decades, the Internet has become an indispensable part of our daily lives and day-to-day growth of the Internet has made it an impeccable space for attackers [1]. Because of the faintness of computer systems' security, many of the computer networks have been exposed to hackers' attacks. According to the 2017 Cybercrime Report, cybercrime activity is one of the largest challenges over the next twenty years and Cybersecurity Ventures projected that cybercrime will cost the world $6 trillion yearly by 2021, up from $3 trillion in 2015 [2]. Although using firewalls and some access control mechanisms are the most preferred techniques for intrusion detection/prevention systems, they fail when the attacker is within the network or when using unknown attack types [3]. Therefore, providing network security is very important to ensure the confidentiality and integrity of data.

Intrusion Detection Systems (IDS) are defined as tools, processes and applications that include monitoring and analyzing of network traffic for the detection of unauthorized attacks, violations and threats[4]. Basically, it is a service used to protect the network and traffic from malicious activity and to prevent the network from attackers' unauthorized access. Most of the Intrusion Detection Systems use one of two detection methods: signature based intrusion detection systems and anomaly detection based intrusion detection systems [5]. Basically, signature based detection is utilized to search harmful package strings in network traffic and compare them with previously identified attacks while anomaly detection is more appropriate to detect unknown attacks based on the profiles recognizing normal (and anomalous) behaviors. Obviously, the main purpose of these techniques is to detect the intrusions for better preparation against future attacks [6].

Ensemble learning is the machine learning technique which combines a set of classifiers to improve accuracy performance in many classification problems. It is accepted that the ensemble classifier can perform better than a single classifier in terms of accuracy and robustness [7]. Therefore, in this study, a new stacking-based ensemble method approach, which provides both high accuracy and good interpretability, is proposed for network intrusion detection.

This manuscript is arranged as follows. Section II gives an extensive review of ensemble learning approaches for intrusion detection. Section III explains the ensemble learning methods and assessment metrics in detail. Section IV explains the data and presents the experimental studies. Section V presents the stacking models' results and comparison analyzes. At last, the paper is concluded by Section VI.

II. RELATED WORK

Various ensemble learning models were suggested in the literature to improve the detection of anomaly in the computer network traffic. Gyanchandani et al. utilized the decision tree algorithm in boosting, bagging and stacking ensemble techniques. They used NSL-KDD dataset and compared their model for classification but did not test it on unseen attacks [8]. Bahri et al. used a hybrid model based upon a Greedy-Boost ensemble technique. They utilized KDD 99 dataset in their study and used the AdaBoost, C4.5, and Greedy-Boost algorithms. They measured the precision and recall metrics for the performance of models. Their findings stated that their strategy was great in identifying rare attacks, however was not tried on unknown attacks [9]. In another study, to solve the problem of intrusion detection, Syarif et al. utilized four different classification algorithms for boosting, bagging, and stacking ensemble methods. Their methodology accomplished more than 99% accuracy rate at known intrusions detection. But, the accuracy rate was just 60% at detecting new types of intrusions [10]. Shrivas et al. developed an ensemble technique for detecting intrusion. They utilized gain ratio for feature selection and a Bayesian network and artificial neural network algorithms as base learners

of classification [11]. In their study Gaikwad and Thool utilized a bagging ensemble method for intrusion detection problems. They used genetic algorithms for feature selection and used partial decision tree-based classifiers for the ensemble technique. Although the ensemble approach utilized in their study reduced the model-building time, the findings showed that the C4.5 algorithm had better results than the bagged PART ensemble technique [12]. Tama and Rhee studied an ensemble technique on the NSL-KDD dataset. For binary classification they used the majority voting method and the averaging of posterior probabilities. Their findings stated that the averaging of posterior probabilities method accomplished the highest performance. But they did not test the model on unknown attacks [13]. In the other study, Choudry and Bhowal developed an ensemble approach based on bagging, boosting and stacking. They utilized NSL-KDD dataset for the binary classification. They also compared machine learning algorithms with ensemble methods and the findings stated that the Boosting algorithm had the best performance [14]. Thaseen and Kumar utilized principal component analysis (PCA) and support vector machines (SVM) with RBF kernel for intrusion detection. They noted that the proposed model could better detect minor U2R and R2L attacks [15].

In addition, there were studies using deep learning algorithms to detect anomalies in computer network traffic in recent years. Naseer et al., have done a comprehensive study. They evolved different intrusion detection models including autoencoders, convolutional neural networks (CNN), and recurrent neural networks (RNN). They used the NSL-KDD training data set for training and the NSLKDDTest + and NSLKDDTest21 test datasets for evaluating the models. Experimental results of deep IDS models showed that Long Short Term Memory (LSTM) model achieved the best accuracy results with 89% in the NSLKDDTest+ dataset and 80% in the NSLKDDTest21 dataset [16]. Aygun and Yavuz used the deterministic autoencoders (AE) and the stochastic denoising auto encoders (DAE) models for IDS and compared the achievements of the models. Their results noted that the suggested AE and DAE models perform almost the same accuracy rate of 88.28% and 88.65% respectively [17]. Yin et al., developed a deep learning method for intrusion detection including RNN. They compared RNN model with traditional machine learning classification methods. The experimental results showed that RNN works with a good detection rate, 83.28% for NSLKDDTest+ dataset and 68.55% for the KDDTest21 dataset [18].Shone et al, proposed a new deep learning model based on nonsymmetric deep autocoder and tested the performance of this model using KDD Cup '99 and NSL-KDD datasets. According to the test results obtained in the study, they reported that the proposed method achieved promising success when compared with the existing approaches [19].Tang et al, proposed a new deep learning model for network intrusion detection based on Software Defined Networking (SDN), which is called DeepIDS. They tested the performance of the proposed model on the NSL-KDD data set with two different deep learning algorithms as anomaly detector. The authors stated that the performance of the model was acceptable according to the results obtained in experimental studies [20].

The findings from the previous studies mentioned above show that many studies have been carried out on network intrusion detection, but there are still gaps for improvement with new studies. For this reason, we believe that the stacking ensemble learning approach proposed in this article will provide a valid contribution to the existing literature with the high accuracy rate it achieves.

III. MATERIALS AND METHODS

A. CLASSIFIER COMBINATION TECHNIQUE

An ensemble of classifiers is a group of methods that combines individual determinations for classifying new instances. The aim of combining classifiers is to increase a single classifier's accuracy [21]. Stacking is the acronym for Stacked Generalization [22]. It utilizes various learning techniques to produce the ensemble of classifiers, unlike bagging and boosting. The primary theme of stacking is to compound classifiers from various learners like support vector machines, decision trees, instance-based learners, etc. Stacking does not use a voting process when combining the classifier, because if

most of the classifiers make poor predictions, that will make a poor classification. To tackle this issue, stacking utilizes the notion of meta classifier. The meta learner (or level-1 model), attempts to learn, using a learning algorithm, how to combine the predictions of the base classifiers (or level-0 models) [21].

In this study five types of classifiers such as Logistic Regression, Artificial Neural Networks, Decision Trees, K Nearest Neighbors and Support Vector Machines were used. The details of these classifiers were as follows:

Logistic regression (LR) is a generalized theory of linear regression. LR is principally utilized to predict binary or multi-class dependent variables. LR investigates the relationship between the input variables and the output by predicting probabilities. As shown in Equation (1), LR measures the probability of Y by given X.

$$P\ (Y = 1|X = x) \tag{1}$$

The main purpose of LR is to utilize the linear regression, where the equation P of linear regression is given in Equation (2).

$$w_1x_1 + w_2x_2 + \cdots = \sum w_ix_i \tag{2}$$

In equation (3) logit transformation is used to transform linear regression to LR and equation (4) gives the solving of P [23].

$$\log \frac{P(x)}{1 - P(x)} \tag{3}$$

$$P = \frac{e^{\sum w_ix_i}}{1 + e^{\sum w_ix_i}} = \frac{1}{1 + e^{-\sum w_ix_i}} \tag{4}$$

Artificial neural networks (ANNs) are mathematical methods that can model highly complicated non-linear functions inspired by biological systems [24]. Artificial neural networks suggest that the nonlinear relationship between dependent variable and independent variables is entirely based upon data without statistical hypothesis. The ANN model is composed of layers with interconnected neurons.The multi-layer perceptron model has an input layer, one or more hidden layers, and an output layer. Neurons in the input layer distribute the input signal to the neurons in the hidden layer. In the hidden layer, the information from the input layer is processed and transmitted to the output layer. In the output layer, the information coming from the hidden layer is processed and the output produced by the network for the input provided to the network from the input layer is presented to the outside world. [25].

Decision tree (DT) is a type of supervised learning algorithm that is mostly used in classification problems. Basically a DT consists of a set of nodes and each node involves a rule for each attribute. An instance that is to be classified is tested according to the node rules by moving down the tree branch, beginning from the root node. The process is reiterated until the leaf nodes for all subtrees. Then each leaf is marked as the best class or it could involve the probability of the target class [26]. In order to obtain the highest possible estimation accuracy, decision tree algorithm separates observations in branches recursively for constructing a tree. To do this, various algorithms such as information gain, Gini index, Chi-square statistics, etc. are used to define the threshold value to divide the observation pool into two or more subgroups.

The k-nearest neighbor (k-NN) is a straightforward and efficient method for classifying objects based on the nearest instances of training in the feature space [27]. For classification, k-NN algorithm evaluates the class of the nearest neighbor according to the given K value. In this algorithm, the

classification of a vector is found by using known vectors. When a sample is selected for the test, to determine the class of this sample k samples are selected that are closest to the sample. Which class has the most examples in the selected set of samples, the tested sample is included in this class. In k-NN, the Euclidean distance is often utilized to measure the distances of two samples:

$$d^2(x_i, x_j) = \|x_i - x_j\|^2 = \sum_{k=1}^{d} (x_{ik} - x_{jk})^2 \quad (5)$$

where $(x_i, x_j) \in R^d, x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$.In this study the k value was determined by an iterative experimentation process.

*Support vector machines (SVMs)*are a member of the generalized linear models' family and they aim to obtain an estimation with a linear combination of variable features [28]. SVM uses nonlinear kernel functions to convert input data into a high-dimensional property field that becomes more manageable than the original input field [29]. Basically, the objective of SVM is to discover a formulation of a hyperplane which distinguishesthe training data. It tries to find the optimal hyperplane by maximizing the distance between the two closest samples in two-class problems to leave data points of the same class on the same side of the hyperplane. While this hyperplane is called optimal hyperplane, the closest vectors on both sides of this hyperplane are called support vectors. After the formulation is determined for the optimal hyperplane, this formulation can be used as a model for assigning new data into a class.

B. PERFORMANCE METRICS

The performances of stacking model discussed in this study were measured using different metrics such as Overall Accuracy (ACC), True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR). Overall Accuracy estimates the ratio of the correctly recognized records to the whole test dataset. The performance metrics were calculated using following formulas:

$$Accuracy\ rate = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

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

$$TNR = \frac{TN}{TN + FP} \quad (8)$$

$$FPR = \frac{FP}{FP + TN} \quad (9)$$

$$FNR = \frac{FN}{FN + TP} \quad (10)$$

True Positive (TP), if the model classifies an attack as an attack, the result is recognized as TP.False Negative (FN), if the model classifies an attack as a normal instance, the result is recognized as FN.False Positive (FP), if the model classifies a normal instance is as an attack, the result is recognized as FP. True Negative (TN), if the model classifies a normal instance as a normal instance, the result is recognized as TN.The values of correctly or incorrectly predicted samples such as TP, FN, FP, and TN were presented in Table 1 in the form of a confusion matrix.

Table 1. *Confusion matrix*

		Predicted Class	
		Attack	Normal
Actual Class	Attack	TP	FN
	Normal	FP	TN

IV. EXPERIMENTS

A. DATASET

KDDCUP'99 dataset is the most widely used dataset to test the models developed for detecting abnormalities in computer network traffic. However, in the studies carried out on this dataset, researchers have determined that some conditions negatively affect the success of the models. In order to solve these problems, some records in the KDDCUP'99 dataset were eliminated and a new dataset called NSL-KDD was created for testing the proposed systems [30]. For this reason, the NSL-KDD dataset was used to evaluate the accuracy performances of proposed ensemble methods. The NSL-KDDtrain dataset contains a total of 125973 records of different attack types and normal data traffic [31]. Each record of the network traffic is represented by 41 features [30]. Details of training and testing records of the NSL-KDD dataset and feature descriptions are indicated in Table2 and Table3, respectively [32].

Table 2. *Training and testing records of the NSL-KDD dataset*

	Description	**NSLKDDTrain+**	**NSLKDDTest+**	**NSLKDDTest21**
Normal	Connections are normal	67343	9711	2152
Attack	DoS, Probe, R2L, U2R	58630	12833	9698
Total		125973	22544	11850

Table 3. *Features of NSL-KDD dataset*

Number	**Features**	**Types**	**Number**	**Features**	**Types**
1	Duration	Numeric	22	is_guest_login	Binary
2	Protocol_type	Nominal	23	count	Numeric
3	Service	Nominal	24	srv_count	Numeric
4	Flag	Nominal	25	serror_rate	Numeric
5	src_bytes		26	srv_serror_rate	Numeric
6	dst_bytes	Numeric	27	rerror_rate	Numeric
7	Land	Binary	28	srv_rerror_rate	Numeric
8	wrong_fragment	Numeric	29	same_srv_rate	Numeric
9	urgent	Numeric	30	diff_srv_rate	Numeric
10	hot	Numeric	31	srv_diff_host_rate	Numeric
11	num_failed_logins	Numeric	32	dst_host_count	Numeric
12	logged_in	Binary	33	dst_host_srv_count	Numeric
13	num_compromised	Numeric	34	dst_host_same_srv_rate	Numeric
14	root_shell	Binary	35	dst_host_diff_srv_rate	Numeric
15	su_attempted	Binary	36	dst_host_same_src_port_rate	Numeric
16	num_root	Numeric	37	dst_host_srv_diff_host_rate	Numeric
17	num_file_creations	Numeric	38	dst_host_serror_rate	Numeric

18	num_shells	Numeric	39	dst_host_srv_serror_rate	Numeric
19	num_access_files	Numeric	40	dst_host_rerror_rate	Numeric
20	num_outbound_cmds	Numeric	41	dst_host_srv_rerror_rate	Numeric
21	is_host_login	Binary			

B. TRAINING

In this stacking model, which is proposed using the supervised learning approach,first the base learners were trained from the initial (base-level) training sets. Then the predictions by the learned classifiers were generated on a distinct validation dataset. And then a meta-level training set was composed from the validation set and the predictions produced by the classifiers on the validation set. Finally, the meta classifier was trained from the meta-level training set. The stacking architecture used to detect anomalies in this study is presented in Fig. 1.

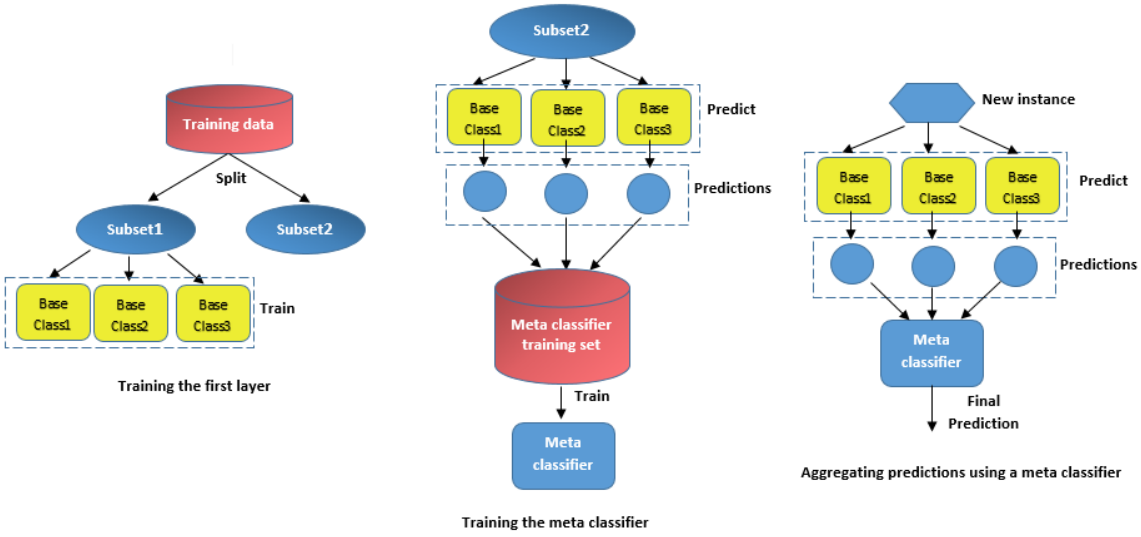

Figure 1.*The stacking architecture used to detect anomalies*

In the stacking technique, four different algorithms were used as base learners and an algorithm was used as a stacking meta learner. Different combinations of LR, DT, ANN, and KNN models were applied. Table 4 presents the stacking model combinations.

Table4.*Stacking model combinations*

MODEL	BASE LEARNERS	STACKING MODEL LEARNER
Stacking Model 1	Decision Tree K-Nearest Neighbor Logistic Regression	Support Vector Machines
Stacking Model 2	Decision Tree Artificial Neural Network Logistic Regression	Support Vector Machines
Stacking Model 3	K-Nearest Neighbor Logistic Regression Artificial Neural Network	Support Vector Machines
Stacking Model 4	Decision Tree Artificial Neural Network K-Nearest Neighbor	Support Vector Machines

As a result of the experiments, the ANN architecture consists of one input layer, two hidden layers and one output layer. The neuron numbers of the input layer relies on the type and amount of variables in the dataset (41 for this research). The neuron numbers of the output layer relies on the count of class labels connected with the researched classification case (2 for this research). To find the optimum number of neurons for each hidden layer, different numbers of neurons were tested. The best results were obtained when 50 neurons were used in the first hidden layer and 20 in the second one.ReLU activation function was used in hidden layers.Adam was used as an optimizer and Keras binary cross-entropy used as a loss function.Learning rate value was set to 0.001.The output layer activation function was softmax.To prevent overfitting, a dropout layer was added after each hidden layer and the dropout value was set to 0.3. Batch size was set to 32 and the training was completed in 14 epochs.In applying the DT method, the Gini index algorithm was used to measure the quality of a division, and the maximum tree depth was not limited. In applying the KNN method, it was found that the best result was obtained when the k value was selected as 12.Polynomial kernel was used in SVM and it was set as degree 15.

V. RESULTS AND EVALUATIONS

In this study, a stacking-based approach was presented to determine whether the incoming network traffic is normal or an attack. The focus of the study was to demonstrate the performance of stacking technique for network intrusion detection. All the codes prepared for the proposed approach were implemented with Scikit-learn library [33] written in Python language. All experimental studies were performed on both NSLKDDTest + and NSLKDDTest21 data sets in order to measure the performance of stacking models. In this context, the general accuracy, TPR, TNR, FPR and FNR values obtained from all models in the test data set for the NSLKDDTest + and NSLKDDTest21 dataset were given in Table 5 and Table 6, respectively. Then the performance of the proposed approach was compared with other studies in this field.

Table 5.*Prediction Results of Stacking Models*

	NSLKDDTest+					**NSLKDDTest21**				
	Overall Accuracy	**TNR**	**TPR**	**FNR**	**FPR**	**Overall Accuracy**	**TNR**	**TPR**	**FNR**	**FPR**
Stacking Model 1 (DT-KNN-LR + SVM)	0,8334	0,9609	0,7369	0,2630	0,0390	0,6845	0,8317	0,6518	0,3481	0,1682
Stacking Model 2 (DT-ANN-LR + SVM)	**0,9057**	0,9049	0,9063	0,0936	0,0950	**0,8432**	0,6951	0,8760	0,1239	0,3048
Stacking Model 3 (ANN-KNN-LR + SVM)	0,8871	0,9048	0,8737	0,1262	0,0951	0,8082	0,6970	0,8329	0,1670	0,3029
Stacking Model 4 (DT-ANN-KNN + SVM)	0,8399	0,9596	0,7493	0,2506	0,0403	0,6968	0,8257	0,6682	0,3317	0,1742

Table 6.*Predicted Evaluation Metrics of Stacking Models*

STACKING MODEL 1 (DT-KNN-LR + SVM)					
	NSLKDDTest+			NSLKDDTest21	
	Attack	Normal		Attack	Normal
Attack	9457	3376	Attack	6322	3376
Normal	379	9332	Normal	362	1790
STACKING MODEL 2 (DT-ANN-LR + SVM)					
NSLKDDTest+			NSLKDDTest21		
	Attack	Normal		Attack	Normal
Attack	11631	1202	Attack	8496	1202
Normal	923	8788	Normal	656	1496
STACKING MODEL 3 (ANN-KNN-LR + SVM)					
	NSLKDDTest+			NSLKDDTest21	
	Attack	Normal		Attack	Normal
Attack	11213	1620	Attack	8078	1620
Normal	924	8787	Normal	652	1500
STACKING MODEL 4 (DT-ANN-KNN + SVM)					
	NSLKDDTest+			NSLKDDTest21	
	Attack	Normal		Attack	Normal
Attack	9616	3217	Attack	6481	3217
Normal	392	9319	Normal	375	1777

Accuracy results of stacking models were given in Figure 2.

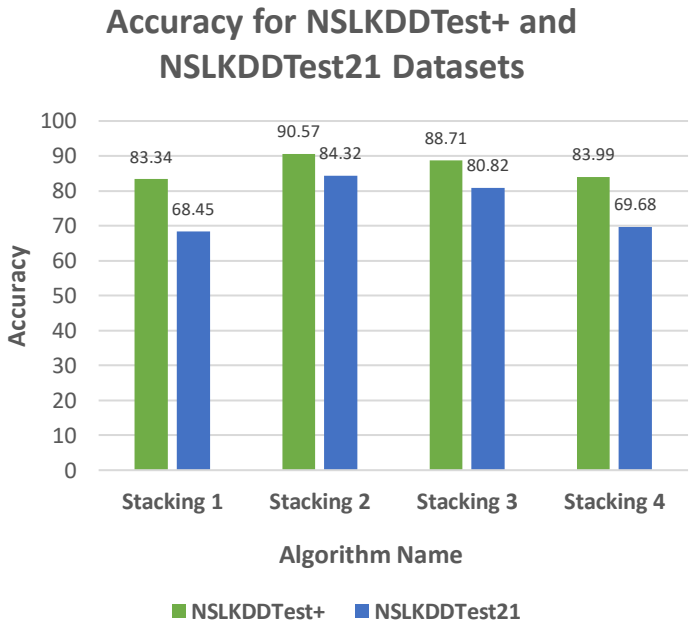

Figure 2.*Comparison of stacking model accuracies applied on NSLKDDTest+ and NSLKDDTest21 datasets.*

There were research papers in the literature using the same training and testing strategy with our proposed study. In Table 7, we compared obtained results of our model with the results of these papers.

Table 7.*Predicted Evaluation Metrics of Stacking Models*

COMPARISON MODEL	ACCURACY (%)	
	NSLKDDTest+	**NSLKDDTest21**
NBTree[30]	82.02	66.16
Decision Tree[34]	80.14	80.14
Fuzzy Classifier[35]	82.74	-
Random Tree+ NBTree[36]	89.24	80.0
Long Short Term Memory LSTM[16]	89.0	83.0
Denoising Autoencoder[17]	88.65	-
Recurrent Neural Networks[18]	83.28	68.55
StackingModel2 (Our approach)	90.57	84.32

As can be seen in Table 7, some researchers used traditional methods for network intrusion detection [30, 34-36], some used deep learning methods [16-18].Tavallaee et al., introduced a new data set, called NSL-KDD. This data set is composed of selected records of the KDD data set and is not affected by any deficiencies [30]. They employed various classifiers for intrusion detection. They reported that NBTree classifier achieved the best accuracy results with 82.02% in the NSLKDDTest+ dataset and 66.16%in the NSLKDDTest21 dataset. Mohammadi et al., used three different classifiers such as neural network-based, distance-based and DT based for detecting network intrusion.Interestingly, they reported the same overall accuracy for both test sets. In addition, no clear information was given about values of the parameters they used in the decision tree classifier proposed in the study [34].Krömer et al., proposed a genetic programming (GP) to evolve fuzzy classifier. They used NSLKDDTrain+ training dataset for training the fuzzy classifier and NSLKDDTest+ data set for testing the classifier. They reported that their approach had an 82.74% accuracy score. In this study, it was not clearly stated whether the parameters used in the proposed model were default or not [35].Kevric et al., evolved a combined classifier model based on tree algorithms. Their combining classifier (Random Tree+ NBTree) achieved the best accuracy results with 89.24% in the NSLKDDTest+ dataset and 80% in the NSLKDDTest21 dataset [36].On the other hand, in some studies [35,17], it was seen that the performance of the proposed models was tested only on the NSLKDDTest + data set. These models were not tested on the NSLKDDTest21 dataset with unknown attacks.Considering the performance of the models in terms of overall accuracy, it is seen that the proposed solution approach in this study has achieved the best results on both test data sets.

VI. CONCLUSIONS

In this paper a stacking ensemble technique was used to develop a network intrusion detection system using the NSL-KDD dataset. To evaluate the effectiveness of proposed approach accuracy results and performance metrics of test datasets were used. The experimental results showed that the stacking method is very useful for detecting intrusions. The ensemble classifier (DT-LR-ANN + SVM) model achieved the best accuracy results with 90.57% in the NSLKDDTest+ dataset and 84.32% in the NSLKDDTest21 dataset.

VII. REFERENCES

[1] C. Tsai, Y. Hsu, C. Lin, and W. Lin, "Expert Systems with Applications Intrusion detection by machine learning : A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.

[2] S. Morgan, (2019, Jun 15). *"2017 Cyber Crime Report."* [Online]. Available: https://cybersecurityventures.com/2015-wp/wp-content/uploads/2017/10/2017-CybercrimeReport.pdf.

[3] G. Karataş and O. Şahingöz, "Neural network based ıntrusion detection systems with different training functions," *IEEE,* pp. 1–6, 2018.

[4] D. P. Vinchurkar and A. Reshamwala, "A review of ıntrusion detection system using neural network and machine learning technique*," International Journal of Engineering Science and Innovative Technology (IJESIT),*vol. 1, no. 2, pp. 54–63, 2012.

[5] K. Leung and C. Leckie, "Unsupervised anomaly detection in network ıntrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian Conference on Computer Science – Volume 38*, 2005, pp. 333–342.

[6] P. Ning and S. Jajodia, (2020, Feb 10). "*Intrusion detection techniques*." [Online]. Available: https://doi.org/10.1002/047148296X.tie097.

[7] L. I. Kuncheva, J. C. Bezdek and R. P. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, vol. 34, no. 2, pp. 299–314, 2001.

[8] M. Gyanchandani, R. N. Yadav and J. L. Rana, "Intrusion Detection using C4 . 5 : Performance Enhancement by Classifier Combination," *ACEEE Int. J. on Signal & Image Processing*, vol. 01, no. 03. pp. 46–49, 2010.

[9] E. Bahri, N. Harbi, and H. N. Huu, "Approach Based Ensemble Methods for Better and Faster Intrusion Detection," in *Proceedings of the 4th International Conference on Computational Intelligence in Security for Information Systems*, 2011, pp. 17–24.

[10] I. Syarif, E. Zaluska, A. Prugel-Bennett, and G. Wills, "Application of bagging, boosting and stacking to ıntrusion detection," in *Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2012, pp. 593–602.

[11] A. K. Shrivas and A. K. Dewangan, "Article: An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD Data Set," *Int. J. Comput. Appl.*, vol. 99, no. 15, pp. 8–13, 2014.

[12] D. P. Gaikwad and R. C. Thool, "Intrusion detection system using bagging with partial decision treebase classifier," *Procedia Comput. Sci.*, vol. 49, pp. 92–98, 2015.

[13] B. A. Tama and K. H. Rhee, "A Combination of PSO-Based Feature Selection and Tree-Based Classifiers Ensemble for Intrusion Detection Systems," in *CSA/CUTE*, 2015.

[14] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," *2015 Int. Conf. Smart Technol. Manag. Comput. Commun. Control. Energy Mater.*, 2015, pp. 89–95.

[15] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of PCA and optimized

SVM," in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, 2014, pp. 879–884.

[16] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[17] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically ımproved autoencoder based models," *2017 IEEE 4th Int. Conf. Cyber Secur. Cloud Comput.*, 2017, pp. 193 198.

[18] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for ıntrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[19] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A deep learning approach to network intrusion detection," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[20] T. A. Tang, L. Mhamdi, D. McLernon, S.A.R. Zaidi, M. Ghogho and F El Moussa, "DeepIDS: deep learning approach for intrusion detection in software defined networking," *Electronics*, vol. 9, pp. 1533, 2020.

[21] A. Ledezma, R. Aler, and D. Borrajo, "Heuristic Search-Based Stacking of Classifiers," 2002.

[22] D. H. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[23] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, 1st ed. Chapman & Hall/CRC, 2012.

[24] S. Haykin, *Neural Networks and Learning Machines*. New Jersey: Prentice Hall, 2008.

[25] E. Öztemel, *Yapay Sinir Ağları.* İstanbul, Türkiye: PapatyaYayıncılık, 2012

[26] N. Demir and G. Dalkiliç, "Modified stacking ensemble approach to detect network intrusion," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 26, pp. 418–433, 2018.

[27] W. Wang, X. Zhang, and S. Gombault, "Constructing attribute weights from computer audit data for effective intrusion detection," *J. Syst. Softw.*, vol. 82, pp. 1974–1981, 2009.

[28] M. Pontil and A. Verri, "Support vector machines for 3D object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 6, pp. 637–646, 1998.

[29] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[30] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[31] UNB. (2021, March 6). "*The NSL-KDD Data Set.*" [Online]. Available:https://github.com/ defcom17/NSL_KDD.

[32] T. Poongothai, K. Jayarajan and P.Udayakumar, "An Effective and Intelligent Intrusion DetectionSystem using Deep Auto-Encoders", *IJAST*, vol. 29, no. 9s, pp. 3139–3154, 2020.

[33] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[34] M. Mohammadi, B. Raahemi, A. Akbari, and B. Nasersharif, “New class-dependent feature transformation for intrusion detection systems,” *Secur. Commun. Networks*, vol. 5, pp. 1296–1311, 2012.

[35] P. Krömer, J. Platos, V. Snásel, and A. Abraham, “Fuzzy classification by evolutionary algorithms,” *2011 IEEE Int. Conf. Syst. Man, Cybern.*, 2011, pp. 313–318.

[36] J. Kevric, S. Jukic, and A. Subasi, “An effective combining classifier approach using tree algorithms for network intrusion detection,” *Neural Comput. Appl.*, vol. 28, pp. 1051–1058, 2016.