Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 9, August 2021: 6014-6027

Supervised Learning Approach to Detect and Evaluate Malicious Inside the Cloud Environment

Pranay Jha[1*], Dr. Ashok Sharma[2], Dr. Mithilesh Kumar Dubey[1]

[1]Lovely Professional University

[2]University of Jammu

*Corresponding author: pranay1988jha@gmail.com (Pranay Jha)

Emails: drashoksharma@hotmail.co.in (Dr. Ashok Sharma), drmithilesh.dwivedi@gmail.com (Dr. Mithilesh Kumar Dubey)

Abstract

SaaS, PaaS, and IaaS platforms are available to meet any organization’s IT needs. As a result, all industries are moving their infrastructure to the cloud. To combat this, attackers frequently attack the environment by exploiting a vulnerability in the infrastructure due to its distributed nature and dynamic configuration. Various security frameworks are widely used on customer and cloud provider premises. The attacks, however, are becoming more frequent. To improve the security on traditional frameworks, this paper introduced a security framework for evaluating the network packet behavior. It provides a solution for a variety of rapidly growing network attacks, and it will help to detect harmful network activities. Several classification algorithms of Machine Learning are used in this paper, including Random Forest (RF), Decision Tree (DT), Naive Bayes (NB), Support Vector Machine (SVM), and K-nearest neighbor (KNN). We tested our method on the UNSW-NB15 standard dataset. The results show that our method provides more accuracy and better result.

Keywords: Cloud Computing, Intrusion Detection System, Machine Learning, Malicious Behaviour, Supervised Learning, Cloud Security, Security Threats

1. Introduction

Throughout the last decade, cloud computing has gained popularity due to its numerous IT capabilities, which include the provision of SaaS, PaaS, and IaaS platforms that can be customized to meet the specific IT needs of any organization [1]. To improve availability, manageability, and cost-effectiveness, organizations across all industries are moving their infrastruc- ture to the cloud [2]. However, because of the distributed nature of infrastructure and the dynamic nature of configuration, security has always been a significant concern, as attackers frequently attempt to infiltrate an environment by exploiting a weakness in the infrastructure.

Customer and cloud provider premises have already seen a large number of security frameworks and intrusion detection systems of various types installed and in use. The attacks, on the other hand, are becoming more frequent on a day-to-day basis [3]. While typical security systems are trustworthy when it comes to identifying users' credentials, but there are some intentional attacks either from insiders or outsiders. In that situation, the behavior needs to detect the network packet [4].

It has been discussed in this paper how a security framework is used to determine the security of different types of network packets based on the behavior. We hope that by preventing behavior-based activities from occurring within the network system, we will be able to provide solutions for a variety of rapidly growing network attacks that have been observed recently. The malicious and normal behaviors of users are defined and illustrated in this study using packets gathered through the network. The analysis is responsible for classifying samples as malicious (0) or normal (1). We believe that by integrating established security methods with machine learning techniques, we may develop a new framework for securitydetection that is both resilient and secure for all users. Extracting relevant features for learning has been accomplished through data preparation. In the second module, we used several classification methods to train the model. This research examined the accuracy and effectiveness of several classification algorithms of machine learning, including Random Forest (RF), Decision Tree (DT), Naive Bayes (NB), Support Vector Machine (SVM), and K-nearest neighbor (KNN), to determine the best algorithm for proficiently learning the pattern of malicious attacks. Using the UNSW-NB15 standard dataset, we evaluated the performance of our model and found it to be effective [5]. The experimental results prove that our projected method is more accurate than the existing method. Following actions have been assumed in this work.

1. The most recent dataset UNSW-NB15 has been used to train the model.
2. A pre-processing step has been performed on the dataset to prepare it for classification training.
3. Various Machine Learning classification models have been compared to find the best performance of the model.
4. Model is evaluated based on Accuracy, Precision, Recall, F1 Score, and Confusion Matrix.
5. The resultant model will predict the malicious and normal behavior

This paper is organized into different sections. The next sections are showing Related work, Data Description, Data Pre-Processing, Experiment, and Results, and the last section is showing in the Conclusion.

2. Related Work

As security is one of the major concerns in Cloud Computing, many researchers have worked towards providing a robust security framework for the Infrastructure. In this section, we have reviewed the existing works and provided a brief description of the outcome.

The author recommends comparing J48, Naive Bayes, and Random Forest to determine which is the most efficient. The purpose of this research was to determine how to improve the detection rate and accuracy of the detection model. The comparative analysis enabled the development of new patterns and processes to deal with the massive volume of audit data. [6]

The author examines network traffic characteristics to enhance threat detection. For an efficient study, an enhanced model must be built to store the data. A ratio of 80 and 20 is used for training and testing in this study. Using network transaction data for training, an enhanced hybrid model was created to predict threat level thresholds According to the findings, the hybrid technique significantly reduces the supervised learning approach to detect and evaluate malicious inside the cloud environment computational and temporal complexity. For binary class and multi-label class datasets, the hybrid model was 99.81 percent accurate. The difficulties were solved utilizing the data purification by-election method with information gain. The hybrid method uses J48, Random Tree, Naive Bayes, and others. [7]

Machine learning algorithms have been shown in research to be capable of detecting harmful behavior. Machine learning methods are crucial for automated behavior analysis, given the enormous number of available data of various types of dangerous behavior. Machine learning techniques can be used to construct recognition systems based on the characteristics of network packets. [3]

The author suggested a method for finding anomalous patterns using Trapezoidal Area Estimation (TAE) and Geomet- ric Area Analysis (GAA). This method was applied to the UNSW-NB15 dataset's features, and the Beta Mixture Model (BMM) was utilized to construct all parameters of the network and distances between observations. The authors constructed a normal profile using observations with deviations and normal observations were identified as aberrant patterns. Addition- ally, they employed principal component analysis (PCA) to minimize the dimension of the underlying data in network connections. [8]

The author suggests a cloud-based They divided security into three groups. Memory utilization, peak memory consump- tion, threads, and handles are all included in the first level. In the second level, packets, and bytes for each feature's address are used. Meta features were created by employing mean, variance, and standard deviation. They use two engines include a system analysis engine (SAE) and a network analysis engine (NAE) to evaluate malicious detection during DoS attacks. The work lacks a robust enhanced detection system to monitor user behavior in Cloud-based systems. [9]

3. Classification Techniques

3.1. Support Vector Machine (SVM)

When it comes to classification and regression, the SVM method is a supervised machine learning methodology that can be applied. It is mostly employed in the solution of classification problems. Since it allows for quick and simple prediction procedures, Support Vector Machine is the best reliable method for classification in machine learning. Support vectors are used to divide class labels into related classes in a data repository, which allows for easier searching. It divides support vectors into groups based on their gamma coefficient. When gamma is equal to 0, the SVM predicts a curvature. The hyperplane is predicted by SVM based on the data that is provided [10].

3.2. K-Nearest Neighbour (k-NN)

The K-Nearest Neighbour (KNN) technique is another reliable classification algorithm that is mostly used for categoriz- ing data into groups. It is also known as the nearest neighbor algorithm. One of its most appealing aspects is that it can be used for both classification and regression tasks, which is one of its most appealing characteristics [11].

To evaluate any technique using k-NN, we normally consider three critical factors:

1. Predictive Power
2. Calculation Time
3. Simple output interpretation

KNN classifies data points using the idea of "several neighbors." The letter "K" in KNN denotes the number of neighbors that must be identified.

3.3. Decision Tree

The decision tree is perhaps the most widely used tool available for classification and prediction. A decision tree is a tree structure similar to a flowchart in which each internal node represents an attribute test, each branch represents the conclusion of the test, and each leaf node holds a class label. Classification by decision trees is accomplished by arranging instances along the tree from the root to the classification leaf node. Classification of an instance begins with the root node, then moves along the tree branch based on its significance. Subtree for the new node is then created [12].

3.4. Random Forest

Random forest is a robust, easy, and familiar to use the technique in machine learning that regularly provides exceptional results even when no hyperparameter tuning is performed. Additionally, it is one of the most often utilized algorithms due to its simplicity and diversity. One significant benefit of random forest is that it can be used to solve classification and regression issues. Consider random forest classification, as classification is frequently regarded as the fundamental building block of machine learning [13].

3.5. Naïve Bayes

Another Bayesian categorization technique is the Naive Bayes algorithm. The classifier predicts that the predictors are not independent of one another, which is correct. In other words, the classifier assumes that no characteristic in a class is connected to another feature in another class [14].

4. Dataset Description

To construct a combination of real-time modern regular activities and artificial modern attack behaviors in the UNSW-NB 15 dataset [15]. Several tools such as IXIA PerfectStorm and Bro-IDS tool were used at UNSW Canberra’s Cyber Range Lab to generate the network packets. 100 GB of raw traffic was captured using the tcpdump program which includes PCAP files. This dataset contains different nine types of attacks which are named reconnaissance, denial of service (DoS), generic analysis, exploits, backdoors, fuzzers, shellcode, and worms. There is a total of 49 features with 2 class labels [16]. A list of these properties is available in the UNSW-NB15 features.csv file.

- Dataset has 540,044 records which contain four CSV files as UNSW-NB15 1.csv, UNSW-NB15 2.csv, UNSW-NB15 3.csv, and UNSW-NB15 4.csv, respectively.

- These CSV files have a Testing and Training set which is named UNSW_NB15_testing-set.csv and UNSW_NB15_training-set.csv.

- The testing set has 82,332 records whereas the Training set has 175,341 records.

- We have used UNSW NB 15 Testing set and further worked with our experimental work. Types of Attacks in Dataset

Attack No. 1 (Analysis) An analysis-based attack is the first line of defense against port scanners, which include HTML file penetrations and spam among other things [17].

Attack No. 2 (Worms) When using the second method (Worms), a hacker duplicates himself and then distributes the duplicated code around a network of computers. A network environment is typically used to spread the code around the world. When the security of the target computer is breached, this procedure is initiated [18].

Attack No. 3 (Backdoors): It is a type of anomaly threat that gives unauthorised access to the system [19].

Attack No. 4 (Fuzzers): It is a type of attack that attempts to disable an application or network by using randomly generated data [20].

Attack No. 5 (Denial-of-service (DoS)): Malware attempts to create a machine or OS inaccessible to its active users by stopping the services temporarily that are connected to the Internet [21].

Attack No. 6 (Generic): It is a type of attack which collision against ciphers. The most understandable example is a cipher that requires a key known as N-Bit; the general assault takes a cipher and tries to decrypt the N-Bit using 2N keys [22].

Attack No. 7 (Shellcode): It is sometimes referred to as Bash, takes use of flaws in the command-line shells of different operating systems. As a result of being infected via remote code execution, a huge number of devices and appliances became vulnerable when Shell-shock was first used in September 2014 and allowed attackers to gain complete access and control over the workstations and appliances [23].

Attack No. 8 (Exploits): Exploit is a type of attack which can take the form of a series of small software which involves the discovery of a security hole in an operating system and the subsequent exploitation of that flaw to gain complete control over the system [24].

Attack No. 9 (Reconnaissance): Negative/theft methods are used to gather information from networks and services in this type of assault. Acquiring knowledge about the target network and then using that information to undertake illicit search and tagging of existing VoIP systems, vulnerabilities, and services are examples of reconnaissance attacks [25].

5. Data Pre-Processing

Data pre-processing is always an important process in the development of any machine learning approach. The gathered information is organized into raw data that contain relevant and irrelevant values. Pre-processing is a method of filtering raw data by removing unnecessary or useless information that can harm the effectiveness of the decision engine in detecting harmful behavior [26]. Following that, the features that have been obtained are relevant inputs for the experiment. There are several phases in Data Pre-Processing which include Missing Value Imputation, Feature Encoding, Feature Selection, Data Splitting, and Feature Normalization process. UNSW NB-15 dataset has two class levels and 47 features. Pre-processing was carried out on the dataset due to the presence of continuous, discrete, and symbolic features across a wide range of time scales and ranges. During the experiment, all nominal features were converted into integers to ensure that the results were as accurate as possible. Numerical qualities with a wide range of values can be difficult to deal with because of their complexity. Logarithmic scaling was employed to narrow the range of possible values for them as a result, which helped to reduce the number of possible values. The Boolean properties did not necessitate the use of scaling. A normalization method called min-max normalization was used to determine the smallest and largest values of each feature within a range of values.

5.1. Missing Value Imputation

The UNSW-NB15 data set has many missing values. Missing data complicates data and can bring bias while also reduces accuracy. Table 1 lists the missing values and the feature name. is_ftp_login, Ct_flw_http_mthd, and ct_ftp_cmd is the most common feature to have missing values. A record with a high percentage of missing values for one feature also has many missing values for other features. We had two options for resolving this issue. We could either remove these samples from the dataset by filtering them from blank or perform imputations to correct the missing values. We used missing value imputation because removing the irrelevant features would reduce also reduce the accuracy of the model. In imputation, these substituted values can be derived using a variety of techniques.

Missing values in a dataset are replaced with the mean of all available samples' values. This method maintains the data set's size and is simple to use; however, the data's unevenness is reduced.

Table 1. Missing Values in different features

Feature Name	**Missing Values**
ct_ftp_cmd	7685
ct_flw_http_mthd	8759
is_ftp_login	12998

5.2. Feature Encoding

The dataset contains both categorical and numerical features [27]. Table 2 is showing the type of features and feature numbers in the dataset. Many of the Machine Learning classifiers do not support categorical data, so we need to convert the categorical data to numerical data. To convert all categorical features into the numerical feature, one hot encoding has been used. Below Figure 1 is showing categorical features converted to numerical features.

Table 2. Type of features and existence in dataset

Type of Feature	**Feature Number**
Integer	2, 4, 8, 9, 10, 11, 12, 13, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47
Nominal	1, 3, 5, 6, 14
Timestamp	29, 30
Float	7, 15, 16, 27, 28, 31, 32, 33, 34, 35
Binary	36, 39

supervised learning approach to detect and evaluate malicious inside the cloud environment

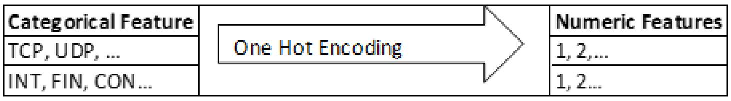


Figure 1. Feature Encoding to Convert Categorical to Numeric Feature

5.3. Feature Selection

This section discusses various methods for selecting features. The relevant features are selected using feature selection methods [28]. These techniques, which select a subset of compatible features for model creation, are classified as a filter, wrapper, and hybrid techniques. Filter methods quantify each feature independently of the classifier, rank them, and select the best. The chi-square test is an example of a filter-based method that was used in this work. It will select the most pertinent features, which will result in increased accuracy.

5.4. Data Splitting

In this stage, the dataset has been divided into two sets. One is in the Training set and another one is for the Testing set. The ratio of dataset split is 80:20. We have used 80% data for training purposes and 20% data for testing purposes.

Training of model will be done using the training set whereas test set will be used for evaluating the model. This is one of the important tasks in Machine Learning. Some researchers also use the validate set of the data, and some of uses test set from different dataset. In this work, we have used UNSW-NB15 dataset for both training and testing phase.

5.5. Feature Normalization

Feature scaling is one of the phases that is quite important in data preparation. There are multiple types of feature scaling such as Feature Standardization and Feature Normalization. In this work, we have used Feature Normalization. The There are various sorts of features in a dataset, which affect the analysis outcome. As a result, dimensionality must be reduced. Moreover, each sample feature must have a uniform distribution of values. Normalization is the best way to solve these problems [29]. Once the data has been normalized, the unified data scale allows for a thorough comparative study of the unique indicators of the original data. Using a min-max normalization strategy, we altered the initial value to ensure that it is mapped between [0, 1].

$$X\ normalization = \frac{X - min(X)}{max(X) - min(X)}$$

In the above equation, when X is the feature value, max (X) is the highest value, and min (X) is the minimum value in a pattern.

6. Experiment and Result

6.1. Experimental Setup

Setting up the environment is important for every phase of the experiment. We need a robust underlying hardware which can support Machine Learning libraries and processing. We have set up the environment using the computer used in the experiment is an Intel NUC box equipped with an Intel Core i7 processor and 64 GB of memory. It has a 1 TB SSD drive and runs the Windows 10 operating system. We used certain Python scripts using PyCharm IDE and Google Colab. Several data science libraries such as Scikit-learn have been used to implement the experiment work and to evaluate the results.

6.2. Evaluation Matrix

UNSW-NB15 dataset has been used for the experimental work. Several evaluation metrics such as Accuracy, Recall, Precision, F1 score, ROC Curve, and Confusion Matrix have been used to evaluate the performance of the model. These matrices are described below.

Accuracy

Accuracy is the sum of True Positive and True Negative which are further divided by True Positive, True Negative, False Positive, and False Negative [30]. To calculate this, total correct predictions are divided by total observations in datasets. The lowest level can be 0.0 and the highest can be 1.0. Below is the formula of Accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

Precision is calculated by dividing the total True Positive with the total True Positive and False Positive. It is always within 0.0 to 1.0 where 0.0 results for no precision and 1.0 results a perfect precision. The formula for Precision is given below.

$$Precision = \frac{TP}{TP + FP}$$

Recall

A recall is calculated by the total number of True Positive and False Negative. It is always outcome within 0.0 to 1.0 where 0.0 results as no recall and 1.0 results and perfect recall. The formula is given below.

$$Recall = \frac{TP}{TP + FN}$$

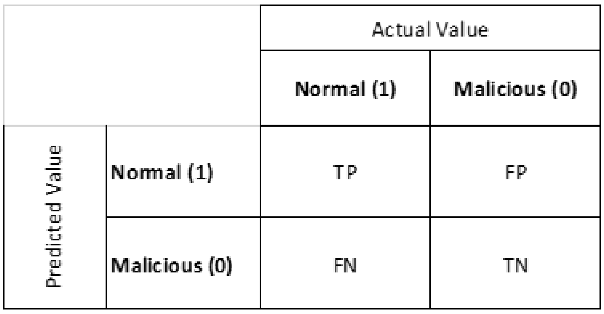
F1-Score

F1 Score is calculated to be twice of Precision into Recall which is divided by the total number of Precision and Recall. F1-Score is a function for calculating the correctness of a model by using Precision and Recall. The formula is given below.

$$F1\ Score = \frac{2\ X\ Precision\ X\ Recall}{Precision + Recall}$$

Confusion Matrix

The confusion matrix is a table to illustrates the model's performance on test data with the actual value. It has True Positive, True Negative, False Positive, and False Negative values within the table as showing in Figure 2. This given table is for binary classifiers.



		Actual Value	
		Normal (1)	**Malicious (0)**
Predicted Value	**Normal (1)**	TP	FP
	Malicious (0)	FN	TN

Figure 2. Confusion Matrix Table

After categorization, we may see the anticipated and actual values for the results. A confusion matrix follows four rules:

True Positive (TP) - In this case, both the expected and actual results turn positive.

True Negative (TN) - In this case, both the projected and actual results become negative.

False Positive (F P) - The value that was projected was incorrect. Even though the model predicted a positive result, the actual result was negative. This is also known as Type 1 error.

False Negative (FN) - The expected value turned out to be incorrectly predicted. In this case, the actual value was positive, but the model was predicted as negative. This is also known as Type 2 error.

6.3. Evaluation Result

This section will evaluate these models and present the results. Various evaluation metrics such as Accuracy, Precision, Recall, F1 scores, ROC, and Confusion Matrix have been used. F1-Score has been calculated using twice of Precision into Recall which is divided by the total number of Precision and Recall. Precision is the proportion of actual identified positive values to all correctly predicted positive values, whereas recall is the proportion of correctly identified positive values to all correctly predicted positive values. As showing in Table 3 and Figure 5, the Accuracy, Precision, Recall, and F1 scores are visualizing the model's performance of each algorithm. Python sklearn metrics libraries were used during the implementation of this work to calculate evaluation metrics. The performance of the model is illustrated for each possible classification threshold value. The ROC curve represents the model's dynamic evaluation. ROC curve is showing in Figure

6. The confusion matrix was represented as showing in Figure 7.

Table 3. Performance of the proposed model

Classifier	**Accuracy**	**Precision**	**Recall**	**F-Score**
Decision Tree	0.966	0.982	0.975	0.978
SVM	0.784	0.753	0.903	0.821
Logistic Regression	0.857	0.855	0.891	0.873
Random Forest	0.974	0.985	0.968	0.976
Random Forest with Tune Model Hyperparameters	0.959	0.965	0.959	0.962

s showing the types of classifiers that have been used in this work, and comparative analysis using the different evaluation matrices. We have used Decision tree, SVM, Logistic Regression, and Random Forest techniques in this work. Accuracy is showing best result for Random Forest. Figure 3 is showing the visualized analysis of all the classifiers.

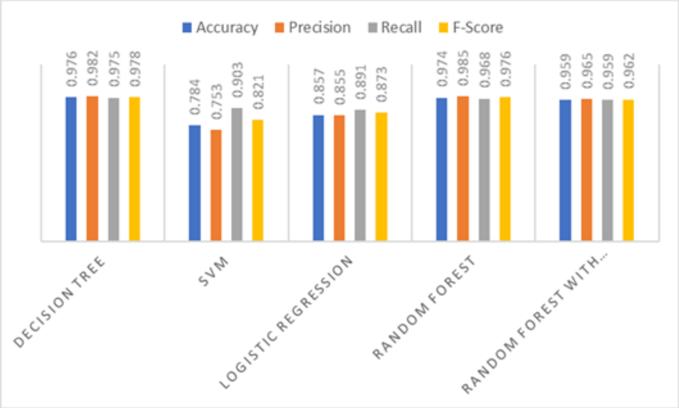


Figure 3. Comparison Result of different classifiers

supervised learning approach to detect and evaluate malicious inside the cloud environment Figure 4 is showing Accuracy result using all the classifiers.

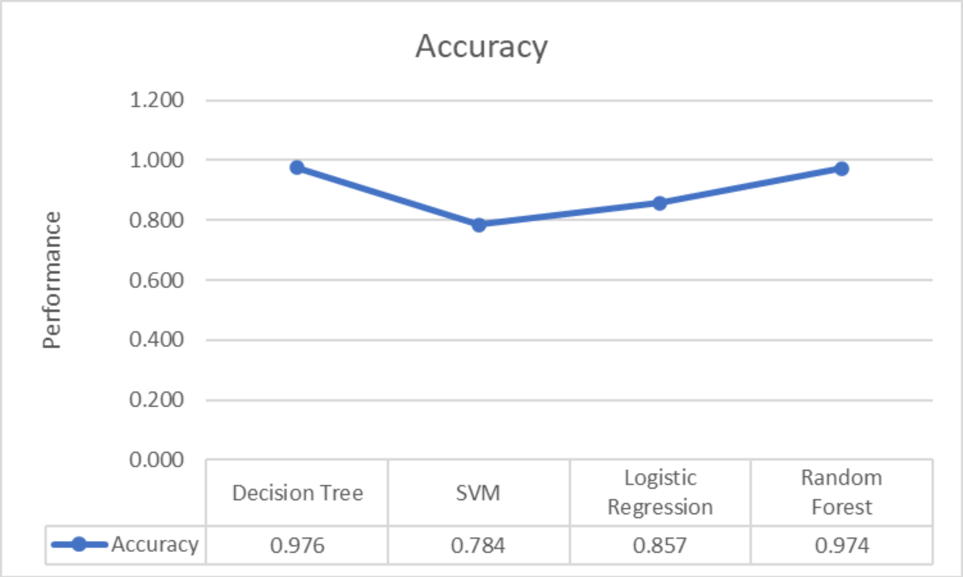


Figure 4. Accuracy Comparison

Figure 5 is showing the result for F1 score for all the classifiers. Figure 6 is showing ROC curve for all the classifiers.

Figure 7 is showing Confusion Matrix for all the classifiers.

We used Support Vector Machine, Random Forest, Decision Tree, and Logistic Regression. The accuracy results are depicted in Figure 3. The best accuracy score was obtained by Random Forest, resulting in a score of 97.4 percent. Support Vector Machine’s best accuracy score is 78.4 percent, which is the lowest one. The confusion matrix is depicted in Figure

7. In general, the accuracy obtained using various imputation strategies does not vary significantly. The confusion matrices display a heat map of the accuracy achieved by each classifier. According to Figure 4, Random Forest has the highest accuracy in terms of true positives and negatives.

7. Conclusion

Recent events demonstrate that existing frameworks are insufficient to combat environmental security threats. Each day, attackers introduce a new type of attack that results in significant losses for organizations. Identity and access management systems must monitor and evaluate users’ activities within the environment. In this study, we proposed a model that

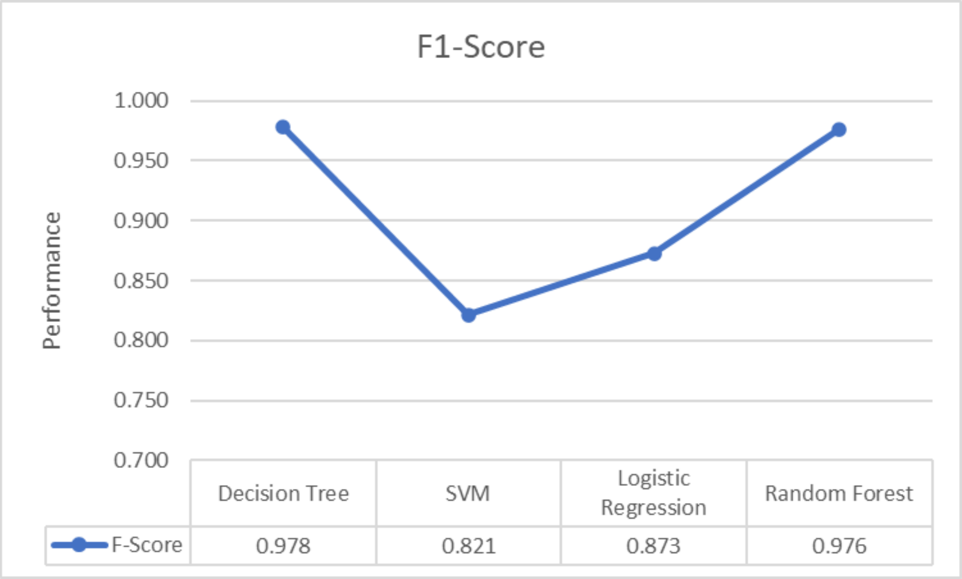


Figure 5. F1 Score comparison

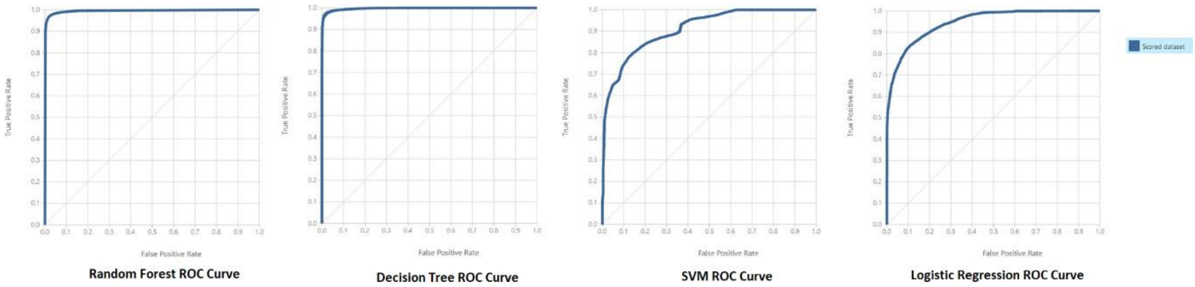


Figure 6. ROC curve for performance evaluation

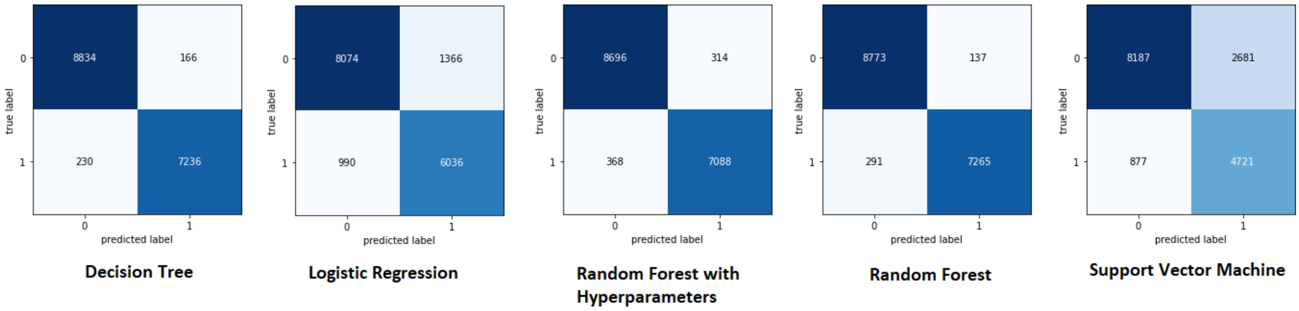


Figure 7. Confusion Matrix for all classifiers

leverages substantial activity features to recognize normal and malicious user behavior. The current study examined the potential for detecting Normal and Malicious behavior using an anomaly-based machine learning model. Experiments with the models were conducted using the UNSW-NB15 data set. It was further segregated into a training and a test set. Total 80% of the data was kept for the training set and 20% on the test set. Additionally, feature selection has been used to select the most pertinent features for experiment work that provide the highest level of accuracy. This work makes use of a variety of classification techniques, including Support Vector Machine, Random Forest, Decision Tree, and Logistic

Regression. Random Forest achieved the highest accuracy score of 97.4 percent. The best accuracy score for Support Vector Machine is 78.40 percent, which is the lowest. Performance has been analysed using a variety of evaluation matrices, including Accuracy, Precision, Recall, F1-Score, ROC, and Confusion Matrix. The confusion matrices depict a heat map of each classifier's accuracy. We can conclude that Random Forest achieves a higher level of accuracy on the dataset used, which is optimal for aligning our model. Its accuracy in terms of true positives and negatives is unmatched. The findings indicated that the methodology was sufficiently robust to ensure the study's validity. We identified and classified users' behaviours as either normal or malicious using this collaborative system.

References

1. Mazhar Ali, Samee U. Khan, and Athanasios V. Vasilakos. "Security in cloud computing: Opportunities and challenges", Information Sciences, 305, 357–383, 2015.
2. V Chang, M Ramachandran, Y Yao, Y. H Kuo, and C. S Li. "A resiliency framework for an enterprise cloud", Int. J. Inf. Manag, 36(1), 155–166, 2016.
3. P Jha and A Sharma. "Framework to Analyze Malicious Behaviour in Cloud Environment using Machine Learning Techniques", 2021 International Conference on Computer Communication and Informatics (ICCCI), pages 1–12, 2021.
4. Anjana and Ajit Singh. "Security concerns and countermeasures in cloud computing: a qualitative analysis", International Journal of Information Technology, 11(4), 683–690, 2019.
5. N Moustafa and J Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW- NB15 network data set", 2015 Military Communications and Information Systems Conference (MilCIS), pages 1–6, 2015.
6. Nabeela Ashraf, Waqar Ahmad, and Rehan Ashraf. "A Comparative Study of Data Mining Algorithms for High Detection Rate in Intrusion Detection System", Annals of Emerging Technologies in Computing, 2(1), 49–57, 2018.
7. Shadi Aljawarneh, Monther Aldwairi, and Muneer Bani Yassein. "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model", Journal of Computational Science, 25, 152–160, 2018.
8. Nour Moustafa, Jill Slay, and Gideon Creech. "Novel Geometric Area Analysis Technique for Anomaly Detection Using Trapezoidal Area Estimation on Large-Scale Networks", IEEE Transactions on Big Data, 5(4), 481–494, 2019.
9. Mohammadhadi Alaeiyan, Saeed Parsa, and Mauro Conti. "Analysis and classification of context-based malware behavior", Computer Communications, 136, 76–90, 2019. 2021.
10. J Zhang and M Zulkernine. Cheng Xiang, Png Chin Yong, and Lim Swee Meng. "Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees", Pattern Recognition Letters, 29(7), 918–924, 2008.2021.
11. J Choi, C Choi, B Ko, D Choi, and P Kim. "Detecting Web based DDoS Attack using MapReduce operations in Cloud Computing Environment", J Internet Serv Inf Secur, 3, 28–37, 2013.
12. J Feng, Y Chen, D Summerville, W Ku, and Z Su. "Enhancing cloud storage security against roll-back attacks with a new fair multi-party non-repudiation protocol", 2011 IEEE Consumer Communications and Networking Conference, pages 521–522, 2011.
13. D Chumachenko and S Yakovlev. "On Intelligent Agent-Based Simulation of Network Worms Propagation", 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), pages 11–14, 2019.2021.
14. S VivinSandar and Sudhir Shenai. "Economic Denial of Sustainability (EDoS) in Cloud Services using HTTP and XML based DDoS Attacks", International Journal of Computer Applications, 41(20), 11–16, 2012.
15. A Albugmi, M O Alassafi, R Walters, and G Wills. "Data security in cloud computing", 2016 Fifth International Conference on Future Generation Communication Technologies (FGCT), pages 55–59, 2016.
16. C. K Chen, S. C Lan, and S W Shieh. "Shellcode detector for malicious document hunting", 2017 IEEE Conference on Dependable and Secure Computing, pages 527–528, 2017.
17. M Jha, S Sharma, and P Jain. "Uncoating the Global Virus: SARS Coronavirus-2", 2020.
18. X Li, J D Smith, and M T Thai. "Adaptive Reconnaissance Attacks with Near-Optimal Parallel Batching", 2017 IEEE

37th International Conference on Distributed Computing Systems (ICDCS), pages 699–709, 2017.

19. T Ahmad and M N Aziz.
20. V Kantorov and I Laptev, 2014.
21. M. A. Adzmi, A. Abdullah, Z. Abdullah, and A. G. Mrwan. “Effect of Al2O3 and SiO2 Metal Oxide Nanoparticles Blended with POME on Combustion, Performance and Emissions Characteristics of a Diesel Engine”, 2019.
22. Lee Friedman and Oleg V. Komogortsev. “Assessment of the Effectiveness of Seven Biometric Feature Normalization Techniques”, IEEE Transactions on Information Forensics and Security, 14(10), 2528–2536, 2019.
23. Fatma Hachmi, Khadouja Boujenfa, and Mohamed Limam. “Enhancing the Accuracy of Intrusion Detection Systems by Reducing the Rates of False Positives and False Negatives Through Multi-objective Optimization”, Journal of Network and Systems Management, 27(1), 93–120, 2019.