Performance Evaluation for Anomaly Intrusion Detection in Data Networks

Sajepan Gnanasivam*, Daniel Tveter*, Nga Dinh*
* Department of Computer Science and Communications.
Faculty of Computer Science, Engineering and Economics.
Østfold University College (HiOF), Norway
{sajepan.gnanasivam, daniel.tveter, thi.t.dinh}@hiof.no

***Abstract*—The number of computers connected to the Internet and hostile attacks on data networks increases daily. Therefore, anomaly detection, which finds network abnormalities using supervised classification approaches, is essential. Thus, this paper focuses on anomaly detection for data networks and uses the UNSW-NB15 dataset, including over 82,000 samples and 49 features. Due to their various origins, most real-world datasets are prone to missing, inconsistent, and noisy data. Implementing data mining algorithms to this noisy data would produce poor results. Consequently, data preprocessing is critical for improving overall performance. Our preprocessing method includes several necessary procedures: data- cleaning, -transformation, and feature selection (dimensionality reduction and feature engineering). This paper conducts performance evaluation by extensive experimentation on several data mining techniques and presents a novel preprocessing method. Data mining techniques such as Artificial Neural Network (ANN), Support Vector Machines (SVM), Logistic Regression (LR), and Decision Tree (DT) are implemented, performing binary classification. The experiments maximize accuracy to discover the right set of hyper-parameters. The results from the experiments indicate that ANN excels with a training accuracy of 98.58%, DT with 96.02%, SVM with 95.1%, and LR with 93.76%. The achieved results are comparable and, in many cases, better than existing works due to our preprocessing framework.**

***Index Terms*—Anomaly Detection, Machine Learning, Supervised Methods, ANN, SVM, LR, DT, Data Network**

I. Introduction

The massive growth in cyber threats, coupled with modern organizations' reliance on the stability and effectiveness of their IT infrastructure, has prompted a shift in mentality. As a result, priorities are altering as the downtime increases for systems within information technology. Intrusions, such as brute force, denial of service, or even penetration within a network, are the most common threat to a network's security. Furthermore, cyber criminals are violently attempting to interrupt network connections, get illegal access to essential data, and then steal, distort, or damage essential data structures. Hence, a dynamic approach to detect and prevent such intrusions is required. It is a critical challenge in the field of computer network security that must be addressed. In light of this, anomaly detection has been hugely popular due to these issues [1]. The goal of network anomaly detection is to find network abnormalities, where supervised classification approaches have yielded several successes through many types of research.

The difficulty of discovering unusual patterns in network traffic that do not conform to predicted typical behavior is referred to as anomaly-based intrusion detection in networks. In many application fields, these nonconforming patterns are referred to as anomalies, aberrations, outliers, or surprises [2]–[4]. Efforts have been undertaken over the last three decades to develop automated systems for detecting network irregularities [5], [6].

Anomaly detection in data networks is a vital and rapidly evolving field of study since our everyday is becoming more and more data-driven. With more data and information available than ever before, it is critical to analyze and evaluate it properly [7].

In data network, anomaly detection has a wide range of applications, including credit card fraud detection, cyber security intrusion detection, and military observation of adversary activity. An unusual traffic pattern in a computer network, for example, could indicate that a hacked computer is transferring sensitive data to an unauthorized site [2]. According to Cisco's Annual Internet Report, the frequency of breaches and the number of records exposed to each breach are increasing. Between 2018 and 2019, attacks between 100 Gbps and 400 Gbps increased by 776%, and the total number of DDoS attacks will double from 7.9 million in 2018 to 15.4 million by 2023 [8]. Due to the importance of anomaly detection, this paper focuses on detecting aberrations in computer networking.

Several machine learning algorithms were applied in performance analysis at the University of Johannesburg [9]. The proposed method included a feature selection technique utilizing XGboost on the UNSW-NB15 dataset. As a final result with binary classification, ANN received a training accuracy of 94.49%, LR with 93.76, SVM with 70.98%, and DT with 93.65% In another study [1], network abnormality in the UNSW-NB15 dataset is detected using a time-based statistical analysis. According to the findings, the labeled dataset may be utilized to accurately train both supervised and unsupervised machine learning systems. In addition, the authors discovered that the suggested model might be applied to detect network anomalies that have not been addressed previously in network traffic by extracting features from the network packet using unsupervised learning. Furthermore, in [10], the authors present a framework to detect malicious behavior in cloud environments also based on UNSW-NB15 dataset. Decision Tree, Random

Forest, Naive Bayes, KNN, Linear Regression, and SVM are among the machine learning algorithms used by the authors. The results reveal that with feature selection, decision tree received an accuracy of 96.77%, LR with 74.73% and SVM with 68.05%.

In the experimental phase, we focused on five different supervised machine learning algorithms: Artificial Neural Network (ANN), Decision Tree (DT), Logistic Regression (LR), Random forest (RF), and Support vector machine (SVM). As a result, the implemented machine learning algorithms performed better or were comparable to other research done on the same dataset, the UNSW-NB15([9]). However, prior to any experiments or hyperparameter tuning, the dataset had to undergo a preprocessing method to transform the dataset into a readable format. Preprocessing is also essential to reduce training time and for accuracy improvements. Preprocessing consists of data cleaning, data reduction, and data transformation. With the preprocessed dataset, we could do hyperparameter tuning. Each algorithm went through two tuning methods, either manual tuning or an algorithm that goes through a set parameter grid to find the optimal parameters for each algorithm. Then, the algorithms are evaluated based on the four metrics: accuracy, recall, precision, and F1-score, where accuracy is the primary metric for our evaluation compared to other researchers.

The rest of the paper is organized as follows. Section II is the introduction of the UNSW-NB15 dataset which is selected to conduct the analysis and experiments. Section III presents our comprehensive works on data preprocessing including data cleaning, data transformation, and data reduction. The different machine learning algorithms used in this paper is described in section IV. In section **??**, We conducted a performance evaluation of the implemented machine learning algorithms on our preprocessed UNSW-NB15 dataset. Finally, section VI concludes this paper.

II. Dataset

Dataset is the most crucial aspect of a network security analysis. The amount of the data also impacts the performance of Machine Learning algorithms. Selecting the most significant features from the input data can simplify the modeling process and result in faster and more accurate detection rates. Frequently, datasets contain multiple useless, redundant features, which is disadvantageous to the accuracy of the results. In light of this, the UNSW-NB15 dataset is selected. The dataset has also been widely used in many researches [11]–[13].

The UNSW-NB15 dataset was created by researchers from the Australian Center for Cyber Security (ACCS) in 2015 [13], [14] with 49 features. The UNSW-NB15 dataset is more sophisticated and reflective of current attack and normal network traffic, making it suitable for evaluating network intrusion techniques. The number of records in the UNSW-NB15 dataset is 2,540,044 which is stored in four .csv files represented in Table I. Each of the dataset contains attack and normal records. A training set and a testing set were created from the data collection, named UNSW-NB15_training-set.csv and UNSW-NB15_testing-set.csv, respectively. The training set has 82,332 records while the testing set contains 175,341 records including attack and normal records. Do note that the training and testing sets are reversed.

TABLE I
Dataset overview

File name	File Size	Records	Features
UNSWNB15_1.csv	161.2 MB	700000	49
UNSWNB15_2.csv	157.6 MB	700000	49
UNSWNB15_3.csv	147.4 MB	700000	49
UNSWNB15_4.csv	91.3 MB	440044	49
UNSW_NB15_testing-set.csv	31.5 MB	175341	45
UNSW_NB15_training-set.csv	15.0 MB	82332	45

III. Data Preprocessing

Before any Machine Learning algorithm is applied to a dataset, the data has to be preprocessed so that the processed data is in the format that the model can comprehend (fig 1). The UNSW-NB15 contains categorical and non-similar scale features that need to be preprocessed in order to make the dataset compliant with a Machine Learning algorithm [1]. To ensure that the selected dataset is converted into a suitable one, we perform data preprocessing with *data cleaning*, *data transformation*, and *data reduction*.

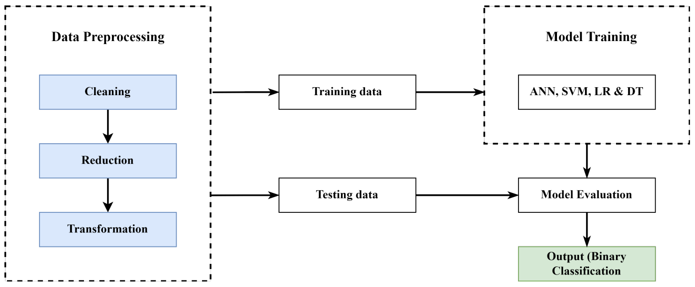

Fig. 1. The complete framework

A. Data Cleaning

Data Cleaning aims to fix the problem with incomplete, inconsistent, and noisy data. But first, the data that is missing or irrelevant needs to be processed.

Missing data points is a common attribute that many datasets have. There are several approaches to solve this problem:

1) Duplicating values rows or columns that occur more than twice in the same dataset. Dropping a duplicate row or column and keeping the first instance is one of the solutions to this problem. The reason for doing this is not to make the specific data point take advantage of or be biased.
2) Dropping rows and columns that includes a not a number (NaN) field or an empty object (NULL) values.
3) Estimating missing values is used if a small percentage of the data is lost. Data imputation techniques such as

mean, median, or mode values are the most common approach to apprehend this problem.

After a thorough analysis, the UNSW-NB15 does not exhibit any missing values to remove or perform data imputation.

B. Data Transformation

The second phase of data preprocessing is Data Transformation which aims to convert the data into a known format readable to a machine learning model. Data normalization transforms numerical columns into a standard scale (-1.0 to 1.0 or 0.0 to 1.0). This process transforms the data columns without distorting the differences in the data range, meaning it will maintain the dynamic range of the dataset.

1) Reversed Datasets: The UNSW-NB15 dataset has the training and testing set samples reversed, caused by an error from the publishers. This means that the testing set samples are double the training set size. We promptly fix the issue by running an if statement on both sets and checking whether the training set has fewer or larger samples than the testing set. The testing set contains 175,341 rows and 45 columns opposed to the training set, corresponding to 82,332 rows and 45 columns.

2) Feature Engineering: The process of changing the raw data into features that better describe the underlying problem to predictive models, resulting in enhanced model accuracy on unseen data, is known as feature engineering. Feature engineering depends on domain knowledge in a specific field to select and transform features from a dataset. It is evident that many features can share requirements characteristics. The procedure entails a combination of data analysis, rule-of-thumb use, and judgment.

The feature engineering process starts with analyzing the various values for each feature for the dataset. Then, by running `train['state'].value_counts` from the pandas library, we output the row count for each distinct value in a feature, in this case, 'state'. Observing the table shows that the row counts for the states 'FIN', 'INT', 'CON' and 'REQ' are more significant than the others. Therefore, the states which have lower row counts are renamed to 'Others'.

The '-' value is present under the 'service' feature, which equals 47,153. This value is converted to 'Others' since the dataset description informs us that the value is a service that is not frequently used. The same method is replicated for the 'service' and 'proto' features. We also observe the row counts for those features and renamed the values to 'Others'. The purpose of this procedure is to reduce the complexity of the model.

3) Categorical Columns: Our machine learning algorithm cannot process Categorical data points, meaning the transformation of the data points into a readable format is essential for the model to run without errors. Therefore, we drop the 'label' feature since our model performs binary classification. Further, to make the categorical features readable by the model, one-hot encoding needed to be implemented. Therefore, we transform each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns using one-hot encoding. A binary vector is used to represent each integer value. The `pd_dummies()` function from the Pandas library accomplished exactly this assignment.

4) Normalization: The UNSW-NB15 dataset has a broad dynamic range of values, indicating that normalization needs to be applied on the numerical columns. The core concept of standardizing/normalizing is to individually delegate $\mu = 0$, and $\sigma = 1$ for the features of X. `Standardscaler()` will independently normalize the features so that each column/feature will produce $\mu = 0$, and $\sigma = 1$. x equals to the observation, μ is the mean and σ is the standard deviation. The equation for standard deviation (3) is achievable by taking the equation for standardization (1) and combining it with the mean for each sample (2).

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} (x_i) \tag{2}$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2} \tag{3}$$

C. Data Reduction

Dealing with a large amount of data for analysis comes with difficulty. Data reduction aims to improve storage efficiency and lower data storage and analysis expenses. The concept of *Dimensionality Reduction* is to reduce the number of features or dimensionality. Data reduction is, in our case, done in the *Feature Selection*. But firstly, we analyze the dataset by looking at the correlation matrix.

1) Correlation: Correlation is the degree to which two variables are linearly related is expressed by correlation. If there is an extensive dataset with many columns, displaying the correlation matrix as a heatmap is a rapid approach to assess relationships between columns. When a dependent and independent variable has a high correlation value, the independent variable affects the output.

The Pearson r correlation is the most extensively utilized statistical approach for determining the degree of linearly related variables. The Pearson correlation between any two variables x, y can be calculated by using Equation (4), where n is number of observations, and i-denotes the i-th observation.

$$r_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}} \tag{4}$$

Fig. 2 shows a graphical representation of the correlation matrix for several features with coefficient values given by a color scale.

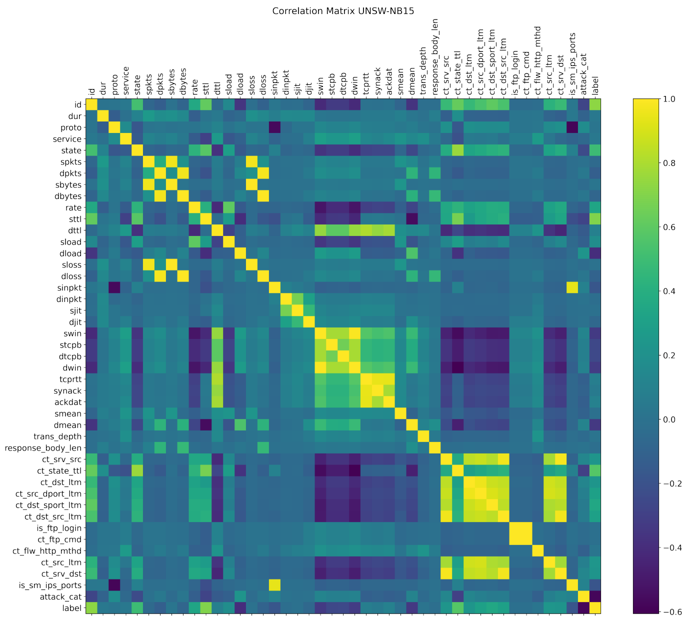

Fig. 2. Correlation plot for UNSW-NB15 Training dataset

2) Feature Selection: Feature selection is one of the core concepts in Machine Learning. This process hugely impacts the performance of any Machine Learning model. When developing a predictive model, the primary objective is to reduce the number of input variables. The number of input variables included in a model's training stage dramatically influences the performance, meaning feature selection and data cleaning should be the first and most crucial step of the developing phase.

Reducing the number of input variables lessens the computational cost and, in many cases, enhances the performance of a given model. The process has two given approaches:

1) Automatically determine input variables using various algorithms
2) Manually choose input variables to remove from the model.

The benefits of reducing the number of input variables are (1) reduce overfitting i.e, the dataset will become less redundant, which in hand will make fewer opportunities to make a decision based on noise, (2) accuracy improvement i.e, training data will be less misleading, which will improve accuracy, and (3) training time reduction i.e, the complexity of the algorithm decreases.

3) Method 1 - Recursive Feature Elimination (RFE): RFE is a well-known feature selection algorithm used to identify which features in a training dataset are more significant in predicting the target variable. RFE works by searching for a subset of features in the training dataset, starting with all features and successfully eliminating them until the targeted number remains. The method can be accomplished by re-fitting the model using the provided machine learning technique, ranking features by relevance, excluding the least important features, and fitting the model again. This procedure is executed until only a certain amount of features are left. Features are evaluated by using either a machine learning model or a statistical technique. In our work, RFE has concluded with the top features and has left us with the following features to drop: `attack_cat`, `id`, `response_body_len`, `spkts`, `ct_flw_http_mthd`, `trans_depth`, `dwin`, `ct_ftp_cmd` and `is_ftp_login`, which we tend to drop.

IV. Machine Learning Algorithms

The following section provides an overview of the used algorithms for our preprocessed dataset.
comment

A. Artificial Neural Network (ANN)

Artificial Neural Networks (ANN), often known as neural networks, are a subset of machine learning algorithms that are at the heart of deep learning algorithms. A neural network's core concept is to imitate a large number of tightly interconnected brain cells inside a computer so that it can learn new things, identify patterns, and make decisions in a human-like approach. A neural network does not need to be explicitly designed; it learns on its own, precisely like a brain.

ANNs include an input layer, one or more hidden layers, and an output layer, each of which links to the layers on either side. They range from a few dozen to hundreds, thousands, or even millions of artificial neurons known as units. The connections between two units are represented by a weight, which might be positive (if one unit excites another) or negative (if one unit repels another). The greater the impact of one unit on another, the higher the weight. (This is similar to how synapses in the brain allow brain cells to communicate with one another)

If a node's output exceeds a threshold value, the node is activated and sends data to the next layer of the network. Otherwise, no data is sent to the network's next layer.

Neural networks use training data to learn and increase their accuracy over time. However, once these learning algorithms have been fine-tuned for accuracy, they become effective tools in computer science and artificial intelligence, allowing us to categorize high-density data quickly. For example, human specialists' manual identification, speech recognition, or picture recognition tasks can take minutes rather than hours. Google's search algorithm is one of the most well-known neural networks.

B. Support Vector Machines (SVM)

Support vector machine is one of the most robust machine learning algorithms out there. Its also highly flexible and can work with both regression and classification problems. SVM's main objective is to separate any given dataset into different classes. The way it separates the dataset is by finding the optimal hyperplane with the use of the support vectors. Some pros about SVM is that it got multiple kernels which got different use for different problems. SVM is also quite effective for data with high-dimensional input features. One downside to SVM is that it requires a lot of tuning to the parameters, especially when the input dimension is larger than the number of samples the dataset has, this can easily lead to overfitting.

C. Logistic Regression (LR)

Logistic regression (LR) machine learning algorithm can do binary classification, and multiclass classification which is the primary use for this machine learning algorithm. Even though it got regression in its name, it's rarely used for this. When it's used for multiclass classification, it uses the one vs all concept to learn. Logistic regression is using either the sigmoid function(7) or something variation. This is applied to the linear model (5) and gives an output between 0 and 1. The output is the probability of belonging to the class labeled 1. 0.5 is the soft boundary of the algorithm, and as the output goes towards 1 or 0 goes further away from this boundary. This is the equation for logistic regression, y is a process by σ(6)

$$y = b + w_1x_1 + w_2x_2 + ... + w_nx_n \tag{5}$$

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{6}$$

$$\sigma(K) = \frac{1}{1 + e^{-k}}c \tag{7}$$

D. Decision tree (DT)

Decision trees are a supervised learning algorithm method used for both regression and classification problems. The objective is to make a model that can predict a value based on a target variable. To make the model, it first needs to make rules based on the training features. The model is easy to understand and interpret, and multiple tools can be used to visualize it.

V. Performance Evaluation

The preprocessing and the build of the machine learning models is executed on a computer with the following specifications: operating system is macOS Catalina(10.15.4) with Intel Core i9 9880H, @2.3 GHz CPU, AMD Radeon Pro 5500M with 4GB GDDR6 graphics card, and 32GB DDR4 memory @2667 MHz. Jupyter lab and Visual Studio code have been our development environment with anaconda distribution. Python version 3.8.8 has been used as the only programming language to analyze and develop our project.

A. Performance Metrics

We look at four different measurements when analyzing the algorithms in this paper; Recall, Precision, F1-score, and Accuracy. Out of those 4, we aimed to maximize accuracy(8) for each algorithm. When calculating the metrics, we use true positive(TP), true negative(TN), false positive(FP), and false negative(FN).

1) *True Positive*
   The true positive(TP) is where the actual class is positive, and the predictor class is positive. So for our case, if the outcome label was 1(attacked) and the predicted value was 1. Then it is a true positive.
2) *True Negative*
   True negative(TN) is where the predicted class is negative, and the actual outcome is negative. So with our data set, it is 0(not attacked) on the predicted and 0 on the actual label class.
   False negatives and false positive is when the predicted and the actual class contradict each other. For example, the predicted class can show one, and the actual value is 0.
3) *False Positives*
   False positives are when the label output is 0 and the predicted value is 1.
4) *False Negative*
   False negatives are when the label output is 1 and the predicted value is 0.

.

$$Accuracy = \frac{TP + TN}{(TP + FP + FN + TN)} \tag{8}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{9}$$

$$F1 - measure = \frac{2 \times (Recall \times Precision)}{Precision \times Recall} \tag{10}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{11}$$

1) Accuracy: Accuracy relates to our model's performance. In other words, based on the input or training data, accuracy is used to recognize patterns and correlations among variables in a dataset. This equation is used to calculate the metric (8)

2) Recall: Recall Offers information on the percentage of the real anomalies that have been classified. Recall provides information about the percentage of the true anomalies that were classified. This equation determines this measure (9).

3) Precision: Precision is the number of correct predictions of the positive observations compared to the total amount of predicted positive observations. So in our case, it's how many attacks that were labeled as an attack are actually attacked (11).

4) F1-Score: F1-score By combining both Recall and Precision scores, the overall performance of the anomaly detection model is determined using a mean value method. The F1 metric incorporates both Recall and Precision scores and uses a mean value approach to estimate the overall performance of the anomaly detection algorithm. This is used to calculate the equation (10).

B. Artificial Neural Network (ANN)

An epoch is a complete transit of the training data through the algorithm in machine learning. The algorithm's epoch value is a significant hyperparameter. It denotes the number of epochs or complete runs through the algorithm's training or learning phase for the whole training dataset. With each epoch, the dataset's internal model parameters are changed. As

TABLE II
ANN EXPERIMENTS

layers	epoch	train acc	test acc	F1	PRC	RCL	time(s)
3	170	0.955	0.872	0.975	0.959	0.975	941.0
4	200	0.959	0.864	0.978	0.963	0.978	1232.6
4	190	0.986	0.849	0.993	0.986	0.993	2603.3
5	200	0.972	0.862	0.981	0.978	0.981	1905.8
7	200	0.984	0.857	0.991	0.985	0.991	2462.5
9	200	0.985	0.861	0.990	0.989	0.990	3041.1
11	140	0.978	0.856	0.989	0.980	0.989	3294.7

an outcome, a single batch epoch is named after the gradient learning algorithm. The batch size of an epoch is usually one and is always an integer.

The number of epochs equals the number of iterations, and the batch size equals the whole training dataset. This is rarely the case for practical reasons. More than one epoch is used in several models. In general, the relationship is $d * e = i * b$, in which d is the dataset size, e is the number of epochs, i is the number of iterations, and b is the batch size.

C. Support vector machine (SVM)

By using the preprocessed dataset with the default parameters on SVM we got an accuracy of 81.57% on binary classification. By running an extensive search on the C value which determines the trade-off between misclassification loss and maximizing the margin. At C equal to 500 and running gamma equal to scale and RBF kernel we got an accuracy of 84.03%. We can see that in the figure 3 that we got the best result at C = 500 with the other parameters mentioned.

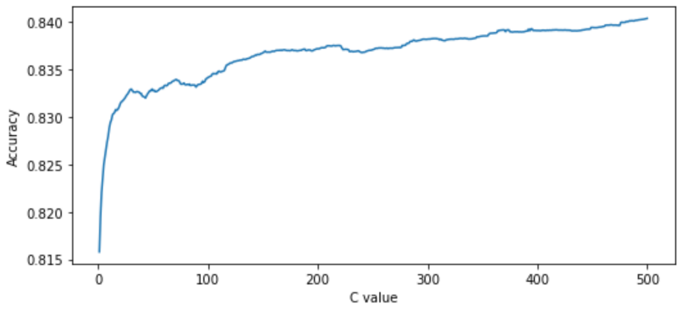

Fig. 3. Graph showing how C value impact the accuracy value from 1-500

TABLE III
SVM EXPERIMENT

kernel	rbf			
C	500			
gamma	scale			
train_acc	0.951			
test_acc	0.840			
F1	0.869			
PRC	0.790			
RCL	0.966			
time(s)	2590			

TABLE VI
BEST RESULTS FROM THE ML MODELS WITH BINARY CLASSIFICATION

method	train_acc	test_acc	F1	PRC	RCL
ANN	0.985	0.861	0.990	0.989	0.990
DT	0.960	0.867	0.890	0.818	0.976
SVM	0.951	0.840	0.869	0.790	0.966
LR	0.937	0.818	0.856	0.758	0.983

D. Logistic regression (LR)

LR had an accuracy of 81.4% on the testing set, with a max iteration of 1000 on the preprocessed dataset. To get higher accuracy on LR, we had to find optimal values for the parameters: C, max_iter, penalty, and solver. With the use of the grid search algorithm, we got the following values shown in this table (ref - table). When those parameters were applied to the algorithm, the accuracy increased to 81.8%.

TABLE IV
LR EXPERIMENT

max_iter	100			
C	100			
Penalty	l2			
Solver	newton-cg			
train_acc	0.937			
test_acc	0.818			
F1	0.856			
PRC	0.758			
RCL	0.983			
time(s)	0.1			

E. Decision tree (DT)

The DT algorithm went through several steps. The first model gave an accuracy of 86.3% on the testing set with no parameter tuning. To increase the accuracy of the DT model, post pruning techniques were applied. This gave the DT model a slightly better accuracy of 86.4%. In the last experiment, we used the grid-search algorithm. The result of this grid search is shown in this table (ref - table) and gave an accuracy of 86.75%.

TABLE V
DT EXPERIMENT

Criterion	gini			
min_samples leaf	20			
min_samples split	14			
max_features	16			
ccp_alpha	2×10^{-17}			
train_acc	0.960			
test_acc	0.867			
F1	0.980			
PRC	0.818			
RCL	0.976			
time(s)	0.1			

VI. Conclusion

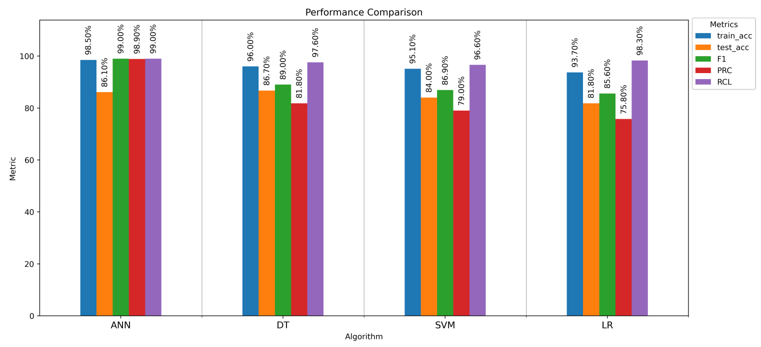

Fig. 4. Performance Comparison

References

[1] D. Sujana, S. Hegde, C. Pankaj, S. Suresh, P. Ramakrishna *et al.*, "Temporal based network packet anomaly detection using machine learning," in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2021, pp. 1–6.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2013.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[4] N. K. Ampah, C. M. Akujuobi, M. N. Sadiku, and S. Alam, "An intrusion detection technique based on continuous binary communication channels," *International Journal of Security and Networks*, vol. 6, no. 2-3, pp. 174–180, 2011.

[5] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516–524, 2010.

[6] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*. IEEE, 1999, pp. 371–377.

[7] J. Long, F. Fang, and H. Luo, "A survey of machine learning-based iot intrusion detection techniques," in *2021 IEEE 6th International Conference on Smart Cloud (SmartCloud)*. IEEE, 2021, pp. 7–12.

[8] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186 080–186 101, 2020.

[9] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, no. 1, pp. 1–20, 2020.

[10] P. Jha and A. Sharma, "Framework to analyze malicious behaviour in cloud environment using machine learning techniques," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE, 2021, pp. 1–12.

[11] L. Zhiqiang, G. Mohi-Ud-Din, L. Bing, L. Jianchao, Z. Ye, and L. Zhijun, "Modeling network intrusion detection system using feed-forward neural network using unsw-nb15 dataset," in *2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*. IEEE, 2019, pp. 299–303.

[12] M. Hammad, W. El-medany, and Y. Ismail, "Intrusion detection system using feature selection with clustering and classification machine learning algorithms on the unsw-nb15 dataset," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. IEEE, 2020, pp. 1–6.

[13] M. Zeeshan, Q. Riaz, M. A. Bilal, M. K. Shahzad, H. Jabeen, S. A. Haider, and A. Rahim, "Protocol-based deep intrusion detection for dos and ddos attacks using unsw-nb15 and bot-iot data-sets," *IEEE Access*, vol. 10, pp. 2269–2283, 2021.

[14] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.