



## Comparative Analysis of Performance and Influence of PCA On Machine Learning Models Leveraging The NSL-KDD Dataset

Osa Edosa<sup>a</sup>, Ekinkonye Ifeoma O.<sup>b</sup>

<sup>a</sup>Department of Electrical/Electronic Engineering, Faculty of Engineering, University of Benin, Benin City, P.M.B. 1154, Nigeria.

<sup>b</sup>Department of Mathematics and Computer Science, Western Delta University, Oghara, P.M.B. 10, Nigeria.

### Article Info

#### Keywords:

PCA, Feature Selection, Machine Learning, NSL-KDD.

Received 18 April 2023

Revised 24 April 2023

Accepted 25 April 2023

Available online 07 June 2023

<https://doi.org/10.5281/zenodo.8014233>

ISSN-2682-5821/© 2023 NIPES Pub.

All rights reserved.

### Abstract

Cyber-attacks have become prevalent in the digital sphere with varied forms and shades of attacks orchestrating significant damage in information systems. Intrusion detection systems which detect attacks on a network are being developed rapidly. Machine learning algorithms are also being utilized in developing such systems with their performance being evaluated by various relevant metrics as well as techniques that could improve their performance. This paper is aimed at performing a comparative analysis of the performance of some machine learning models with respect to the NSL-KDD dataset. The impact of Principal Component Analysis on the models is also investigated. Random Forest, Logistic Regression, Support Vector Machine (SVM), Artificial Neural Network (ANN) and K-Nearest Neighbour (KNN) were considered with and without feature selection. Performance metrics such as Accuracy, F1-score, Precision and Recall were used as basis for comparing the models. Results show that Random Forest gives the best accuracy compared to the other models.

## 1. Introduction

The internet by January 2021 was estimated to have about 4.66 billion active subscribers who represented about 59.5% of world population as against year 2000 with active subscriber statistic of 400 million (which figure represented about 4.8% of world population at the time) [1]. This high increase in percentage underscores the extent to which computer networks in particular and the internet in general have become integrated with human activities. From crucial applications in healthcare, governance and military, to relative ease at which financial transactions can be made with just the punch of a button, our world has become so computer-reliant. The scenario can be described as analogous to a person being connected to a life support machine (a pattern only likely to exacerbate with time). However, this increased reliance on the internet has resulted in the ensuing problem of cybercrime. An increasing number of cyberattacks, such as ransomware, phishing and malicious software and the like are dark sides to implementation of computer networks, as criminally motivated attackers desire ransom as rewards for money theft and identity theft. Motives for cyber-attacks also include espionage, political statements, and so on [2]. These cyber-threats are therefore highly significant issues that information technology-based organizations should address critically [3].

Cybersecurity could be described as the process of defending critical information systems as well as sensitive data from digital attacks, due to technology, people and processes [2]. A lot of research into the development of intelligent cyber security interventions have been going on lately as against the existing traditional Intrusion Detection Systems (IDS) such as Signature-based IDS schemes and

Anomaly-based IDS. Artificial intelligence (AI) techniques such as Machine Learning (ML) are being utilized to detect malicious traffic in computer networks. ML is a sub- field of AI that develops systems with the capability to learn automatically and improve based on experience without explicit programming [4]. These models can therefore carry out intelligent detection of intrusions since they can self-learn from experience thereby improving detection rate and reducing the rate of false positives.

### **1.1 Intrusion Detection System (IDS)**

Intrusion with respect to computer networks is an illegal attempt to gain access into a computer system or network. Intrusion detection is therefore the process of detecting intrusions into a network or a computer. Intrusion Detection Systems could be software, hardware or combination of both, that monitor the network traffic searching for intrusions. They review all incoming and outgoing network activity and thereafter log suspicious patterns. They can further be classified into Host-based Intrusion Detection System (HIDS) [5] and Network-based Intrusion Detection System (NIDS) [6].

#### **1.1.1. Host-based Intrusion Detection System**

Host-based intrusion detection system exist on a single host or device such as a server or personal computer [7]. It is usually a software solution that is installed on the host in order to protect it from intruders.

#### **1.1.2. Network-based Intrusion Detection System**

A network-based intrusion detection system (NIDS) monitors and analyzes network traffic to protect a network system from threats due to intrusion [8]. This solution inspects all inbound packets in search of any suspicious patterns. The two well-known types of NIDS are the signature-based and anomaly-based detection systems.

#### **1.1.3. Feature Selection**

Feature selection is the process of selecting a subset of original data features based on some certain conditions. It is a popular dimensionality reduction technique applied in data mining process. It operates by reducing the number of features, removing irrelevant, noisy or redundant data thus bringing the immediate effects for data applications. Such effects could be speeding up data mining algorithm to improve mining performance such as predictive accuracy as well as result comprehensibility [9].

#### **1.1.4. Principal Component Analysis (PCA)**

Principal Component Analysis as a useful statistical technique is a dimensionality-reduction technique that is employed to reduce the dimension of large datasets. It operates by transforming the large variables into smaller ones that still contain most of the information in the larger dataset. PCA is implemented in areas such as face recognition, image compression, etc. since it is a useful technique for finding patterns in high dimension data [10].

### **1.2 Statement of Problem**

Typically, signature-based IDS utilize a predefined database of security attacks' signatures and attempt to match system events and traffic to the specific attack patterns in the database [11].

However, this operation cannot detect new attacks where pattern and signature are unknown (zero-day attacks). Anomaly-based IDS on the other hand attempt to learn normal system behaviors and categorize all else as anomaly [12]. Nonetheless, they suffer from false positive problems as a normal traffic could be seen as anomaly. Machine learning based IDS could be employed to surmount the above challenges since they can self-learn and detect zero-day attacks. They also can improve on the false positive problem. Investigation into techniques that may improve the performance of the models should therefore be carried out by researchers.

## 2. Methodology

The methodology involved the design, training, testing and feature selection of the supervised machine learning classification algorithms on the NSL-KDD dataset for detecting network intrusion. The flowchart is described in Figure 1.

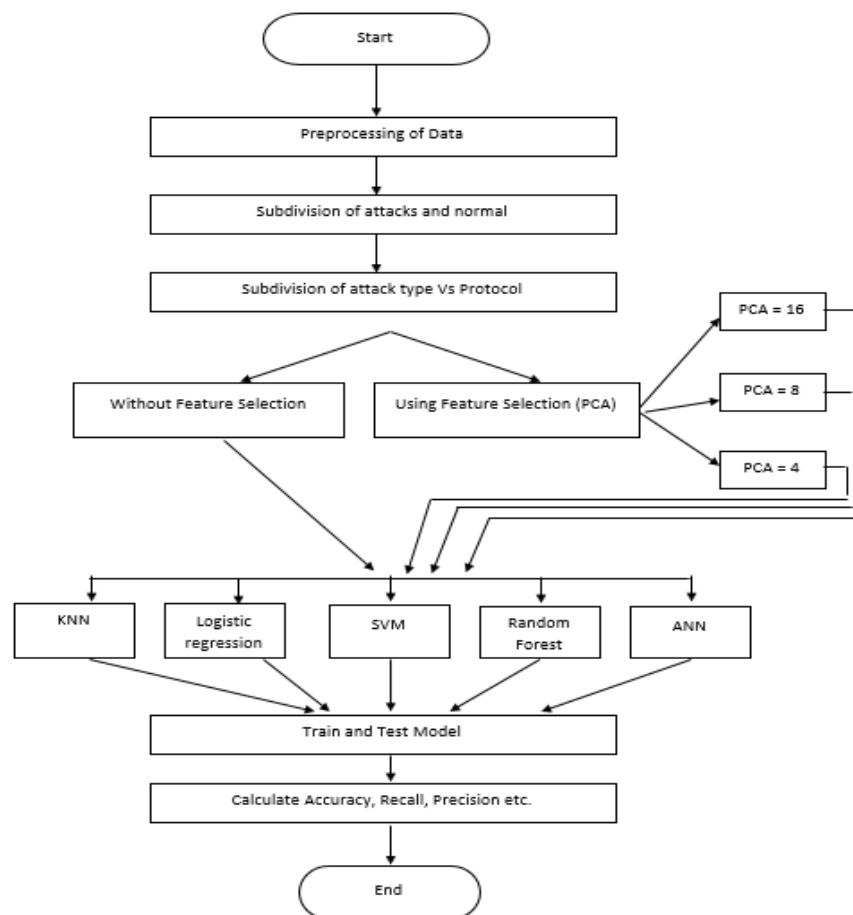


Figure 1: Flowchart of Methodology

### 2.1.Exploring NSL-KDD Dataset

In machine learning operations, dataset is primary. NSL-KDD network dataset, a refined version KDD CUP 99 is employed in this paper. NSL-KDD attempts to solve the inherent problems of KDD CUP 99 [13]. The NSL-KDD records a train dataset of 125971 and test dataset of 22542 as shown from the last five records at the tail() end of the dataset (see Appendix 1). These reasonable record sizes reduce the need for random selection of a small data portion for experiments.

## 2.2.Environmental setup

The algorithm development was implemented on a Windows 10 system with a 2.86GHZ Intel(R) Core i3 processor with 4GB of RAM. Jupyter Notebook with Python 3 installed on Anaconda Navigator was also employed as well as libraries like Pandas, Sklearn etc.

## 2.3.Data extraction and Data Processing

The dataset was first downloaded from Kaggle.com, then checked to determine the total number of rows from both the train and the test dataset. Thereafter, the column labels were included to label the dataset for both training and testing as shown in the Figure 2.

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	

**Figure 2: Dataset with Label for Easy Classification**

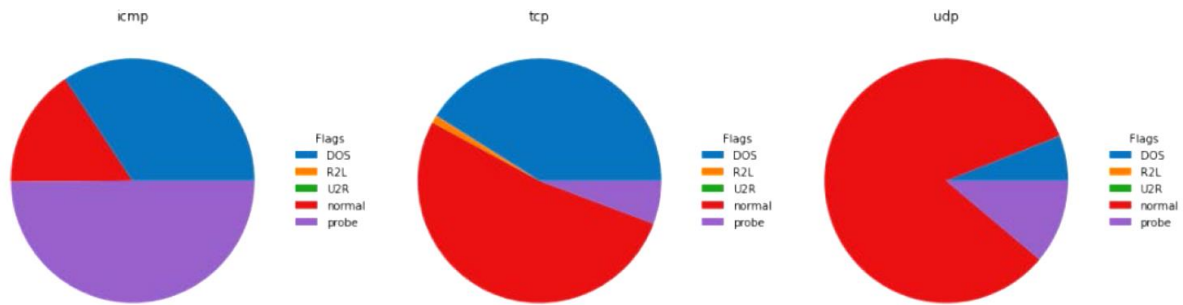
## 2.4.Data Transformations

The attack field was first transformed by adding a column that encodes ‘normal’ values as *0* and ‘attack’ values as *1*. This represented a simple binary classification model that identifies any attack (see Appendix 2). Each attack was classified according to attack type for a more granular prediction model. Four attack types were classified as DOS, Probe, U2R, and R2L. The training dataset consisted twenty-one different attacks which also appear in the test dataset. The known attack types are contained in the training dataset while the novel attacks are the additional attacks in the test dataset (i.e., those not included in the training dataset). An evaluation on the dataset was carried out to visualize the attack type versus the protocol type using *pd. crosstab*. Figure 3 shows that most attacks are against the TCP protocol.

protocol_type	icmp	tcp	udp
<b>attack_map</b>			
DOS	2847	42188	892
R2L	0	995	0
U2R	0	40	3
normal	1309	53608	12434
probe	4135	5857	1664

**Figure 3: Number of Attacks versus Network Protocol**

Using *plt. show()* further displayed the different protocols (i.e., icmp, tcp and udp) and their attack types as shown in Figure 4.



**Figure 4: Pie chart of Network protocols and Attacks**

## 2.5. Classifier Algorithms

Five different classifiers were employed namely; Random Forest, Artificial Neural Network (ANN), Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and Logistic Regression. The rationale was that since each classifier belongs to a different family of classifiers, they would result in different models despite the same dataset being used as input.

## 2.6. Model Building

### 2.6.1. Data Split

Data was split into train and test set using the scikit-learn library with the `train_test_split()` function. Four variables were simultaneously created as follows: (`x_train` and `y_train`) for the training set and (`x_test` and `y_test`) for test set.

### 2.6.2. Model Fitting

Due to the nature of the dataset, Decision trees were a good starting point for building the predictive models. Random forest was first used to build the model. `RandomForestClassifier()` function from the `sklearn.linear_model` sub-module was imported to train the model. Next, the function was assigned to a variable `rf_model` and the `.fit()` function performed the actual model training on the input data `x_train` and `y_train`.

### 2.6.3. Prediction with Test Dataset

After successfully building the model, it was tested using the test dataset to observe its predictive ability. The test feature was passed as the `x_test` and it was allowed to predict the `y_test` after which performance metrics such as Accuracy, Precision, Recall and F1 were evaluated and the confusion matrix was plotted.

The same steps as described above were repeated for the other four classifiers.

### 2.6.4. Feature selection with PCA

The next step was to reduce the features of the dataset to in stages via PCA. The `sklearn.decomposition import PCA`, was imported from the `sklearn` library and assigned values of 16 features, 8 features and 4 features respectively to the `n_components` of the PCA. Standard Scaler (a function which standardizes features by subtracting the mean and scaling to unit variance was employed in scaling the data after performing the `train_test_split()`. This is carried out to reduce bias

from the model. The five classifiers selected were also used to train at the various stages of feature selection. They also were tested on the test dataset and the performance metrics obtained.

### 3. Results and Discussion

Following the use of the five different classifiers from different classification family, after training the model with the training dataset and testing the split data to predict it accuracy, the dataset was then tested to see how the trained model performed. It was then compared to see the classifier that has the highest prediction rate, (i.e., the confusion matrix was evaluated to see the number of true Positive, true negative, false positive and false negative of each classifier’s predictions on the test dataset). Below is the confusion matrix for the test dataset of each of the classifier without feature selection.

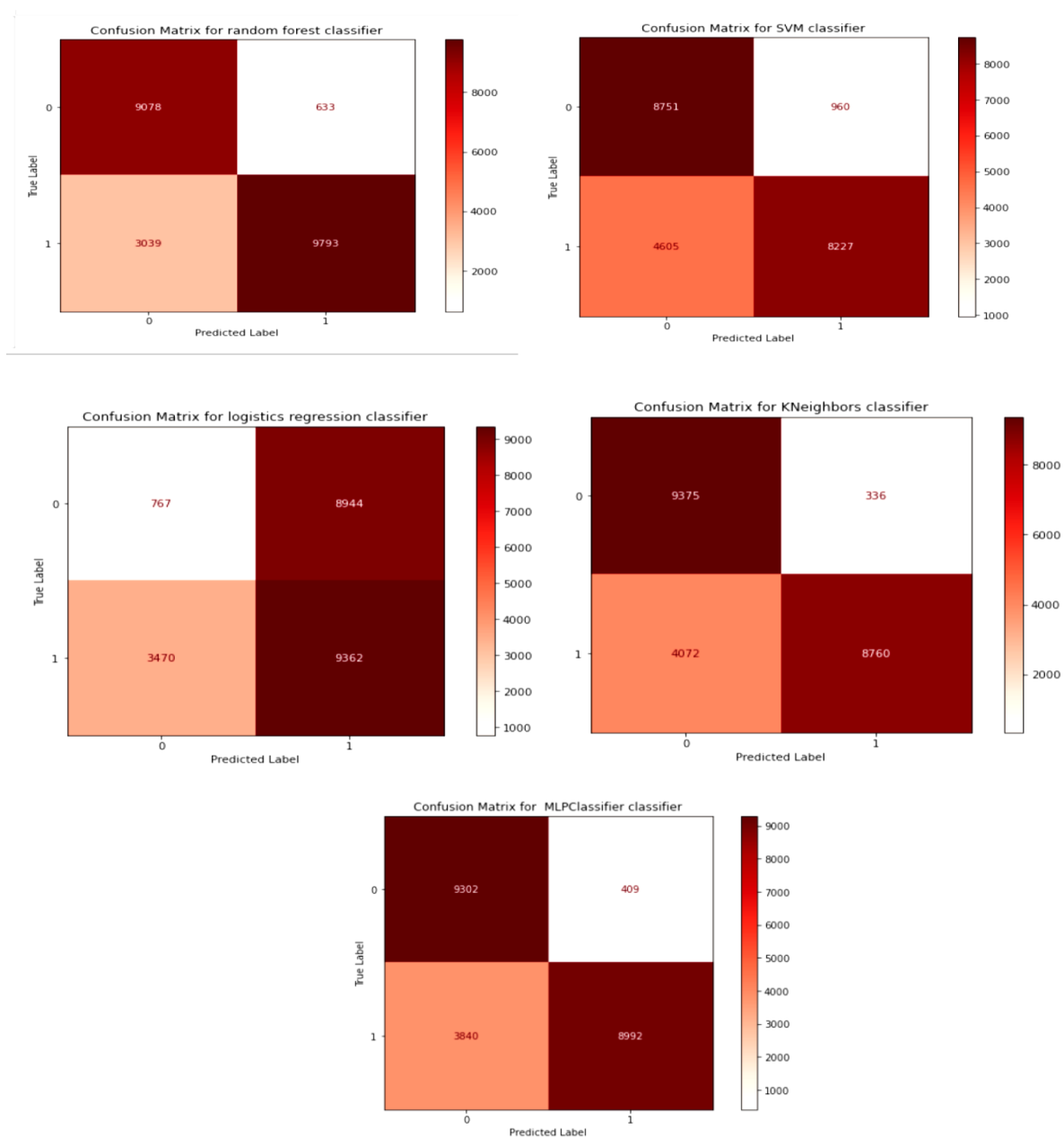


Figure 5: Confusion Matrices for the Five Classifiers

The above confusion matrix plots show information regarding the metrics (Accuracy, Precision, Recall, and F1 score) of the trained model when applied to an unknown test dataset.

The Table 1 also show the metrics for the trained models when an unknown dataset, i.e., the test dataset is pass through the model to predict the number of intrusions present in the network dataset. Table 1 interprets the confusion matrices as it displays the metrics for the trained models when an unknown dataset (i.e., the test dataset) is passed through the model to predict the number of intrusions present in the network dataset.

**Table 1: Performance Metrics of Classifier Models without Feature selection.**

Model Name	Accuracy	Precision	Recall	F1
Random Forest	0.837111	0.939286	0.763170	0.842119
SVM	0.753138	0.895505	0.641132	0.747264
Logistic Regression	0.449319	0.511417	0.729582	0.601323
ANN	0.811516	0.956494	0.700748	0.808888
KNN	0.804463	0.963061	0.682668	0.798978

From Table 1, Random Forest had the highest accuracy of approximately 84% followed by ANN (81%), KNN (80%), SVM (75%) and logistic regression (45%) in the order of accuracy.

**Table 2: Performance Metrics of Classifier Models with feature selection (PCA = 16)**

Model Name	Accuracy	Precision	Recall	F1
Random Forest	0.823582	0.917807	0.757949	0.830253
SVM	0.753183	0.895516	0.641209	0.747321
Logistic Regression	0.691301	0.934844	0.491973	0.644677
ANN	0.777448	0.950640	0.642378	0.766684
KNN	0.781617	0.958705	0.644093	0.770522

**Table 3: Performance Metrics of Classifier Models with feature selection (PCA = 8)**

Model Name	Accuracy	Precision	Recall	F1
Random Forest	0.826997	0.935113	0.747974	0.831140
SVM	0.766757	0.898884	0.665056	0.764490
Logistic Regression	0.511955	0.567708	0.597880	0.582403
ANN	0.759482	0.848870	0.702541	0.768804
KNN	0.793373	0.961808	0.663342	0.785167

**Table 4: Performance Metrics of Classifier Models with feature selection (PCA = 4)**

Model Name	Accuracy	Precision	Recall	F1
Random Forest	0.835337	0.921286	0.777120	0.843084
SVM	0.753138	0.895505	0.641132	0.747264
Logistic Regression	0.678703	0.932117	0.469763	0.624696
ANN	0.781972	0.955996	0.646743	0.771533
KNN	0.783259	0.962192	0.644560	0.771981

From Tables 2 to 4 it is observed that although the overall capability of intrusion detection was influenced by some margin for different PCA implementations, the most accurate model namely Random Forest with values 83% without PCA, 82% with PCA of 16 features, 83% with PCA of 8 features, and 84% with PCA of 4 features remained virtually constant in accuracy. It can therefore be deduced that feature selection using PCA with varying number of features did not alter the accuracy of the model. Instead, it reduced the volume of computational resources required as well as processing time.

#### 4. Conclusion

This paper describes the design, implementation and testing of a network-based intrusion detection system. Five machine learning classifiers were used to rank the forty-one features in the NSL-KDD dataset and comparison was done based on prediction accuracy level. The performance of the classifiers was also evaluated based on their predictive accuracy. The results show that Random Forest had the best accuracy compared to the others. Furthermore, investigation was made to find out if feature selection using Principal Component Analysis can improve performance of the models. It was found that accuracy for Random Forest was virtually constant but computational resources employed were reduced by some percentage.

#### References

- [1] J. Johnson, (2021). Worldwide Digital Population as of January 2021. Available at: [www.statista.com](http://www.statista.com).
- [2] IBM, 2021. What is a cyber attack? Why cyber attacks happen. Available at: [www.ibm.com](http://www.ibm.com).
- [3] Von Solms, R., Van Niekerk, J., (2013). From information security to cyber security. *Comput. Secur.* 38, 97 - 102.
- [4] J. Brownlee, (2018). How to think about machine learning,". Available: <https://machinelearningmastery.com/think-machine-learning>.
- [5] R.A. Bridges, T.R. Glass-Vanderlan, M.D. Iannacone, M.S. Vincent and Q. Chen (2019). A survey of intrusion detection systems leveraging host data. *ACM Comput. Surv.* 52, 1–35.
- [6] N. Sultana, N. Chilamkurti, W. Peng and R. Alhadad (2019). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications* 12, 493–501.
- [7] AT&T Cybersecurity, (2021). Hosted IDS: Host-Based intrusion detection system. Available at: [www.cybersecurity.att.com](http://www.cybersecurity.att.com).
- [8] Techopedia, (2021). Network-based Intrusion Detection System (NIDS). What Does Network-based Intrusion Detection System (NIDS) Mean? Available at: [www.techopedia.com](http://www.techopedia.com).



- [9] Hany M. Harb, et al (2011) “Selecting Optimal Subset of Features for Intrusion Detection Systems”, Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 4 Number 2 (2011) pp. 179-192.
- [10] S. Lakhina, S.A. Joseph, & B. Verma (2010). Feature Reduction using Principal Component Analysis for Effective Anomaly – Based Intrusion Detection on NSL-KDD. International Journal of Engineering Science and Technology Vol. 2(6), 2010, 1790-1799.
- [11] M. Masdari and H. Khezri (2020a). A Survey and Taxonomy of the Fuzzy Signature-Based Intrusion Detection Systems. Applied Soft Computing, p. 106301.
- [12] M. Masdari and H. Khezri (2021b). Towards fuzzy anomaly detection-based security: a comprehensive review. Fuzzy Optim. Decis. Making 20, 1–49.
- [13] M. Tavallaei, E. Bagheri, W. Lu and A. Ghorbani (2009) “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

## Appendix

### 1. NSL-KDD DATASET RECORDS

```

from sklearn.metrics import classification_report
from sklearn import metrics

In [74]: #import the dataset to the jupyter notebook
train_data = "nsl-kdd/KDDTrain+.txt"
test_data = "nsl-kdd/KDDTest+.txt"

df_train = pd.read_csv(train_data)

df_test = pd.read_csv(test_data)

df_train.tail()

Out[74]:

```

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	0.17.1	0.03	0.17.2	0.00.6	0.00.7	0.00.8	0.05	0.00.9	normal	20
125967	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.06	0.00	0.0	1.00	1.0	0.00	0.0	neptune	20
125968	8	udp	private	SF	105	145	0	0	0	0	...	0.96	0.01	0.01	0.0	0.00	0.0	0.00	0.0	normal	21
125969	0	tcp	smtp	SF	2231	384	0	0	0	0	...	0.12	0.06	0.00	0.0	0.72	0.0	0.01	0.0	normal	18
125970	0	tcp	klogin	S0	0	0	0	0	0	0	...	0.03	0.05	0.00	0.0	1.00	1.0	0.00	0.0	neptune	20
125971	0	tcp	ftp_data	SF	151	0	0	0	0	0	...	0.30	0.03	0.30	0.0	0.00	0.0	0.00	0.0	normal	21

5 rows x 43 columns

### 2. CREATING VARIABLES TO ALLOCATE 0 TO NORMAL AND 1 TO ATTACK

```

df_test.columns = columns

In [76]: df_train.head()

Out[76]:

```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	

5 rows x 43 columns

```

In [77]: #create a new variable to allocate 0 to normal traffic and 1 to attack traffic
# This means you are doing 2 class classification: Attack versus No Attack

train_attack = df_train.attack.map(lambda x: 0 if x == 'normal' else 1)
testing_attack = df_test.attack.map(lambda x: 0 if x == 'normal' else 1)

#data

df_train['attack_flag'] = train_attack

df_test['attack_flag'] = testing_attack

```