Contents

List of Figures

List of Tables

A. DATASET

1. Data Set Description

One of the most common publicly available datasets for Intrusion Detection System experiments is the NSL-KDD Dataset, hence, this dataset was used in this research. The NSL-KDD was created and released in 2009 by Tavallaee et al. at the University of New Brunswick, Canada. It was created to address a few of the shortcomings and difficulties in the original KDD Cup 1999 dataset and to be an advanced dataset over it. The improvements include the removal of redundant records in the training set. (Tavallaee et al., 2009). The NSL-KDD dataset intends to give researchers studying intrusion detection systems (IDS) a more realistic and representative dataset.

KDDTrain+ is one of the NSL-KDD data set contains four sub data sets. The NSL-KDD shape of training dataset consist of 125,972 rows and 43 columns. The features used and the breakdown of the data set can be seen in Appendix A.

The summary of the NSL-KDD Dataset in found in table 1 below.

Datasets	**All Records**	**Normal Records**	**DOS**	**R2L**	**U2R**	**Probe**
KDDTrain+	125,973	67,343 (53%)	45927 (37%)	995 (0.85%)	52 (0.04%)	11656 (9.11%)

Table 1: Summary of NSL-KDD Dataset

NSL-KDD datasets comprises four attack curricula: Denial of Service (DoS), Root-to-Local (R2L), User-to-Root (U2R), Probing Attack (Probe) Table 2 below defines the mentioned attacks and attack types (Hassan, 2017).

Attack	**Definition**	**Attack Classes**
DoS	An intrusion attack that overloads network resources, hence, the resources are unavailable to the authenticated and verified users	teardrop, smurf, pod, Neptune, land, back, worm, processtable

Probe	This is a type of intrusion attack which involves the collection of network-related information with respect to activities done in the network.	satan, portsweep, nmap, ipsweep, sain, mscan
U2R	This is a type of intrusion attack that grants an attacker the root access from a normal user account.	perl, buffer overflow, rootkit, loadmodul, xterm
R2L	An intrusion attack that grants an attacker access to the local network especially when there is compromise in the network's integrity	spy, ftp write, imap, phf, warezclient, snedmail, httptunnel

Table 2: Definition of Mentioned Attacks

2. Data Preprocessing

The NSL-KDD dataset has an enormous number of data, hence, there was a need to clean the data and also perform some preprocessing operations to convert the datasets into a suitable format for machine learning. The processes performed are as follows:

i. Column names was assigned to the dataset using assigning attribute name.

$$dataset\_train = pd.read\_csv("KDDTrain.csv", header = None, names = col\_names)$$

ii. Scikit-Learn LabelEncoder was used to convert categories from text to numbers since most machine learning algorithms prefer to work with numbers.

iii. Binary feature was used transform all categorical features using one-hot encoding. This is done using Scikit-Learn OneHotEncoder class.

iv. The dataset was categorized into four attack classes using label such as 0 for normal attack, 1 for Denial-of-Service Attack, 2 for Probing Attack, 3 for Root to Local and 4 for User to Root attack.

3. Data Splitting

The Train+ NSL-KDD dataset employs the percentage splitting methodology of 80% for training and 20% for testing. The training set is used to build the model while the testing set is used to evaluate the model.

4. Feature Selection

NSL-KDD Train+ dataset had 43 columns initially, after one-hot encoding was done, a technique employed in a machine learning model to encode category variables as numerical values, the dataset's output was 124 columns. StandardScaler was used to scale the data frames. Lastly, Univariate Feature Selection using ANOVA F-test was achieved. The characteristics obtained from the ANOVA F-test was used to build the model. The characteristics that were chosen when ANOVA was employed are shown in Appendix B.

5. Machine Learning Libraries

Machine learning libraries serves several purposes and can be used in intrusion detection system to process, analyze and model data for detecting and preventing intrusions.

a. Numpy
b. Pandas
c. Scikit-learn
d. Matplotlib

Numpy: is used to process huge multidimensional arrays and matrices via variation of complex mathematical operations. NumPy is often used to handle the numerical aspects of data, which could be useful in preprocessing and feature extraction for intrusion detection.

Pandas: is used for data analysis. It can help in data preprocessing, cleaning, and exploration, which are essential steps before building an Intrusion Detection System.

Scikit-learn: can be used for data analysis and data mining. It can be used to build and train intrusion detection models on labeled data, making it a crucial component for creating a machine learning-based Intrusion Detection System.

Matplotlib: A well-known Python package for data visualization. When patterns in the data are required, it is most helpful for analyzing the results of Intrusion Detection System to create 2D graphs and charts.

B. METHODOLOGY

This research proposes an overall methodology for detecting intrusions by combining unsupervised machine learning algorithm and supervised machine learning algorithms. Furthermore, the two different classification models to detect intrusions for network traffic analysis, that is, supervised machine learning algorithms such as k-Nearest Neighbor, Decision Tree, and Random Forest for signature-based intrusion detection system and unsupervised machine learning algorithms such as autoencoder for Anomaly-detection intrusions are also deployed dependently. (Pandeeswari & Jeyanthi, 2022) and (Jiang & Tian, 2022).

For the overall methodology for intrusion detection system, the data is first analyzed to detect if it is normal or anomaly, if detected that it is an anomaly, then, it is further analyzed using supervised machine learning algorithms to determine the attack class. The workflow for designing and developing the overall intrusion detection system follows these steps:

1. Dataset collection
2. Data Splitting – Split data into 80% train and 20% test set.
3. Select the normal samples in the train set and discard the anomalies
4. Use MinMax scaling on the numerical features
5. Use one hot encoding on the categorical features
6. Use ANOVA F-test to select top features
7. Build autoencoder model
8. Autoencoder Architecture
9. Train autoencoder for 25 epochs, Loss metric: Mean Squared Error (MSE)
10. Use autoencoder to predict test data
11. Find threshold for classifying as anomaly (Mean (Mean Squared Error) – 0.1 x Standard Deviation (Mean Squared Error)
12. Compare Mean Squared Error of prediction to threshold to assign labels
13. Evaluate
14. Fit algorithm on train set
15. Evaluate on test set

1. General Architecture for overall system methodology

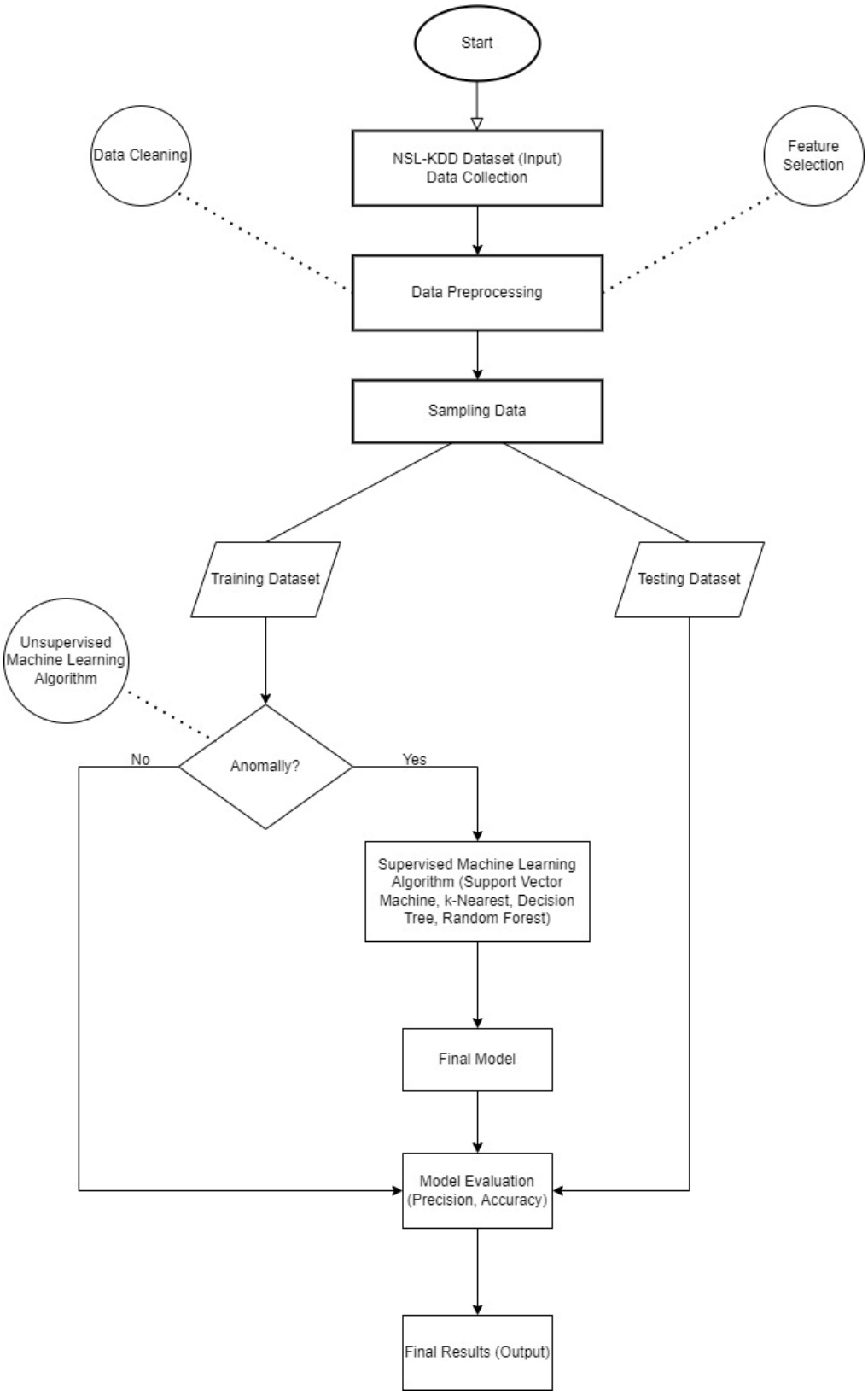

Figure 1: Proposed model for the overall methodology for intrusion detection system

2. General Architecture for Anomaly Based Intrusion Detection System using Unsupervised machine Learning

For the anomaly-based intrusion detection system, unsupervised machine learning model, autoencoder to distinguish between normal traffic or anomaly was deployed. The training dataset for the unsupervised machine learning are unlabeled (Yilmaz, 2022). The workflow for designing and developing an Anomaly-based intrusion detection system using autoencoder follows these steps:

1. Dataset collection
2. Data Splitting – Split data into 80% train and 20% test set.
3. Select the normal samples in the train set and discard the anomalies
4. Use MinMax scaling on the numerical features
5. Use one hot encoding on the categorical features
6. Use ANOVA F-test to select top features
7. Build autoencoder model
8. Autoencoder Architecture
    a. Input Layer: x neurons (x is number of selected features TBD)
    b. Encoder layer: 32 neurons
    c. Encoded layer: 16 neurons
    d. Decoder layer: 32 neurons
    e. Output layer: x neurons
9. Train autoencoder for 25 epochs, Loss metric: Mean Squared Error (MSE)
10. Use autoencoder to predict test data
11. Find threshold for classifying as anomaly (Mean (Mean Squared Error) – 0.1 x Standard Deviation (Mean Squared Error)
12. Compare Mean Squared Error of prediction to threshold to assign labels
13. Evaluate

The proposed model for the anomaly-based intrusion detection system is summarized in figure 2 below

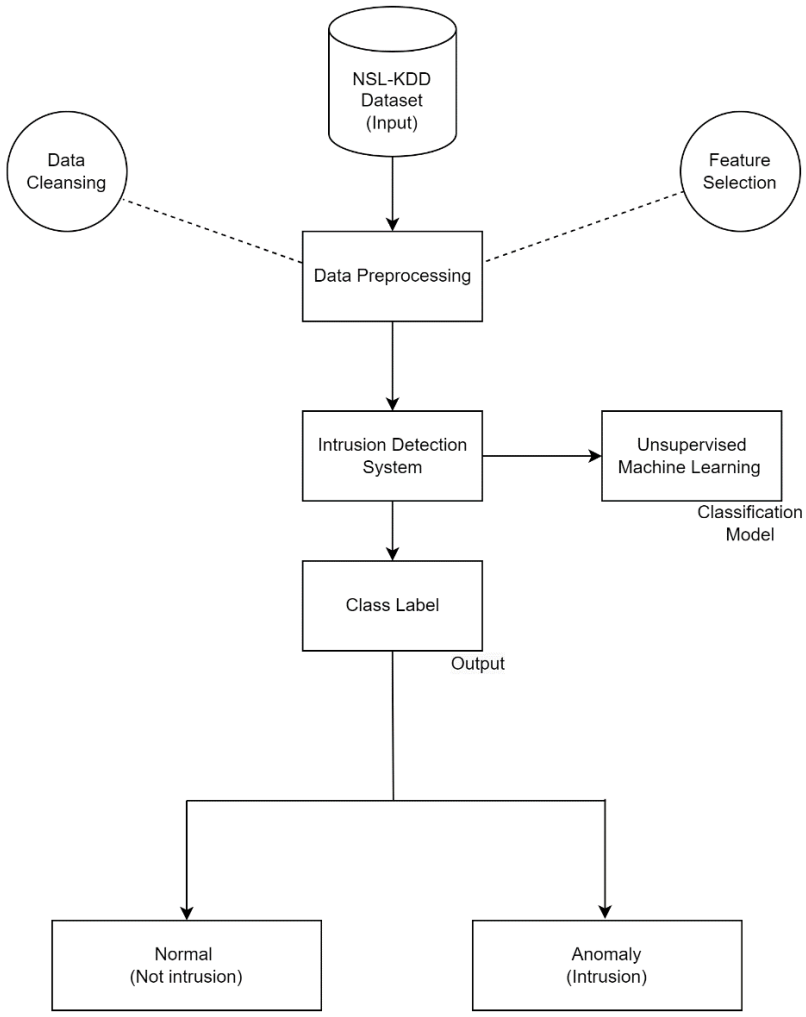

Figure 1: Proposed model for the Anomaly-based intrusion detection system

3. General Architecture for Signature Based Intrusion Detection System using Supervised machine Learning

For the signature-based intrusion detection system, supervised machine learning models such as k-Nearest Neighbor, Decision Tree, and Random Forest was deployed. The training dataset for the supervised machine learning are labeled. The workflow for designing and developing a Signature based intrusion detection system using supervised machine learning follows these steps:

1. Dataset collection
2. Data preprocessing and group attack types
3. Data Splitting – Split data into 80% train and 20% test set.
4. Use MinMax scaling on the numerical features.
5. Use One hot Encoding on the categorical features
6. Use ANOVA F-test to select top features
7. Fit algorithm on train set
8. Evaluate on test set

The following are the model training used for the supervised machine learning:

i. K-nearest Neighbor (k-NN): uses a non-parametric algorithm that classifies instances based on their similarity to neighboring data points. It has been used in IDS for both anomaly detection and classification tasks, where the network traffic is classified based on the behavior of its nearest neighbors (Aljohani & Bushnag, 2021).

ii. Decision Tree: Decisions are made using feature values using decision trees, which are hierarchical structures. They have been used in IDS for their interpretability and ability to capture complex decision boundaries. Decision trees can classify network traffic based on various features and rules (Panigrahi et al., 2021).

iii. Random Forest: uses an ensemble learning method that improves prediction accuracy by combining numerous decision trees. It has been widely used in Intrusion detection system

due to its robustness and ability to handle high-dimensional data. Random Forest can effectively classify network traffic and detect anomalies (Chen et al., 2021).

The proposed model for the signature-based intrusion detection system is summarized in figure 1 below

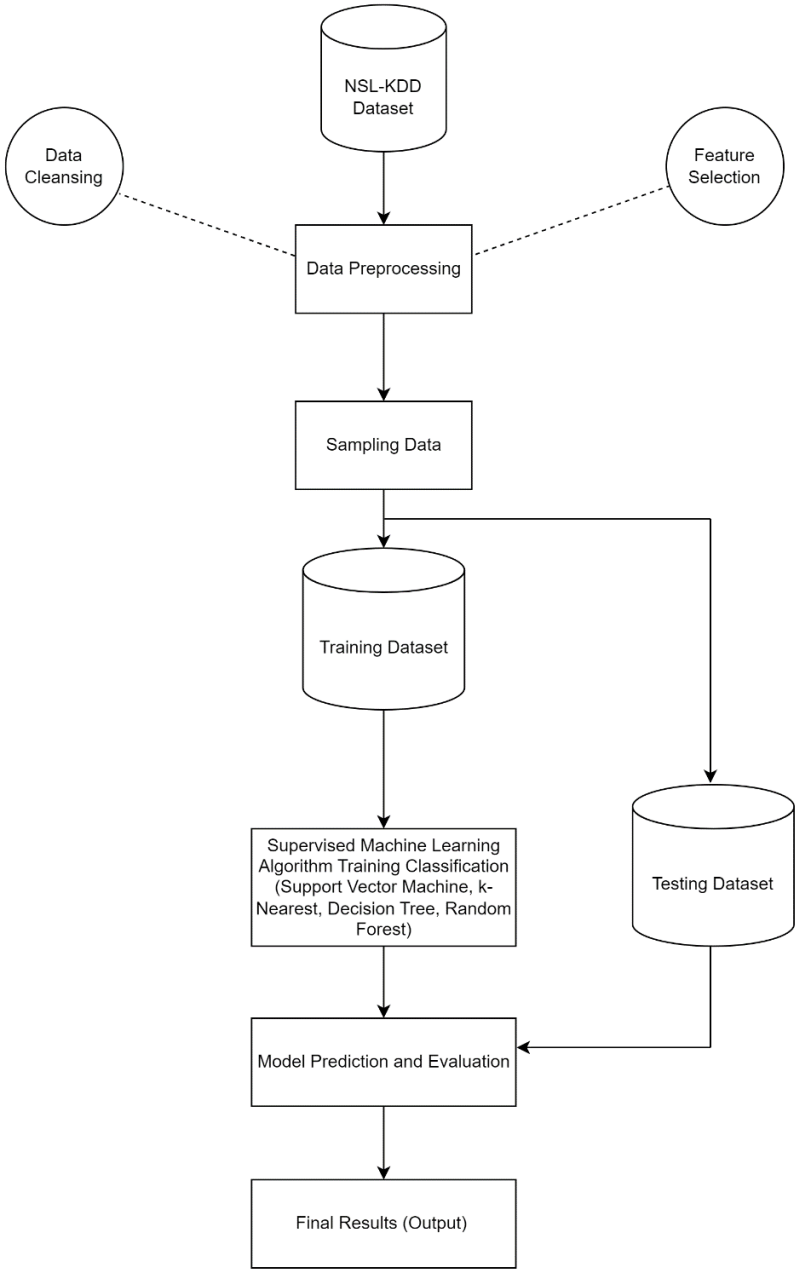

Figure 3: Proposed model for Signature-based intrusion detection system

C. PERFORMANCE EVALUATION

Accuracy, precision, recall, and f-measure were employed as assessment criteria for the performance of the intrusion detection model. (Aljohani & Bushnag, 2021). True Positives (TP) represents the count of anomaly records identified as anomalies, which is akin to all such instances. The False positives (FP) signifies the quality of normal records erroneously marked as anomalies, aligning to the count of inconsistently rejected records. The True Negatives (TN) corresponds to the count of normal records correctly identified as typical, mirroring the cases appropriately recognized. The False Negative (FN) indicates the volume of anomaly records inaccurately classified as normal, and it is comparable to those wrongly permitted or allowed (Jisna et al., 2021).

1. Accuracy

Accuracy quantifies how well a model's predictions are generally right. The ratio of accurate forecasts to total predictions is used to compute it. A higher accuracy suggests that more accurate predictions were made on average.

$$Accuracy = (True\ Positives\ (TP) + True\ Negatives\ (TN)) \ / \ (True\ Positives + True\ Negatives + False\ Positives\ (FP) + False\ Negatives\ (FN))$$

2. Precision

Precision: Precision is the percentage of successfully anticipated positive cases among all positive instances that were forecasted. It emphasizes the model's capacity to reduce false positives. It is determined by dividing the number of true positives by the total number of true positives and false positives. Increased accuracy translates into fewer false positives and more accurate detection of positive events.

$$Precision = True\ Positives\ (TP) \ / \ (True\ Positives\ (TP) + False\ Positives\ (FP))$$

3. Recall

Recall: The proportion of accurately anticipated positive cases out of all actual positive instances is measured by recall, also known as sensitivity or true positive rate. It focuses on the model's capacity to reduce false negatives. It is determined as the ratio of true positives to the total of true positives and false negatives.

$$Recall = True\ Positives\ (TP)\ /\ (True\ Positives\ (TP) + False\ Negatives\ (FN))$$

4. F-measure

The weighted average of recall and accuracy, which gives both measures the same weight, is used to construct the F-measure. A number of 1 denotes flawless recall and precision, and the scale runs from 0 to 1.

$$F - measure = 2 * ((Precision * Recall)\ /\ (Precision + Recall))$$

D. REFERENCES

Aljohani, A., & Bushnag, A. (2021). *An Intrusion Detection System Model in a Local Area Network using Different Machine Learning Classifiers*. https://doi.org/10.1109/acit52158.2021.9548421

Chen, Z., Zhou, L., & Yu, W. (2021). *ADASYN–Random Forest Based Intrusion Detection Model*. https://doi.org/10.1145/3483207.3483232

Jiang, Q., & Tian, A. (2022). Performance Comparisons of Machine-Learning-Based Intrusion Detection Algorithms through KDD Dataset. In *2022 2nd Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*. https://doi.org/10.1109/acctcs53867.2022.00010

Jisna, P. J., Jarin, T., & Praveen, P. N. (2021). Advanced Intrusion Detection Using Deep Learning-LSTM Network On Cloud Environment. *2021 Fourth International Conference on Microelectronics, Signals & Systems (ICMSS)*. https://doi.org/10.1109/icmss53060.2021.9673607

Pandeeswari, G., & Jeyanthi, S. (2022). *Analysis of Intrusion Detection Using Machine Learning Techniques*. https://doi.org/10.1109/icatiece56365.2022.10047057

Panigrahi, R., Borah, S., Bhoi, A. K., Ijaz, M., Pramanik, M., Kumar, Y., & Jhaveri, R. H. (2021). A Consolidated Decision Tree-Based Intrusion Detection System for Binary and Multiclass Imbalanced Datasets. *Mathematics*, *9*(7), 751. https://doi.org/10.3390/math9070751

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). *A detailed analysis of the KDD CUP 99 data set*. https://doi.org/10.1109/cisda.2009.5356528

Yilmaz, A. A. (2022). *Intrusion Detection in Computer Networks using Optimized Machine Learning Algorithms*. https://doi.org/10.1109/iisec56263.2022.9998258

Appendix A – Summary of the Features

#	Feature Name	Description	Type	Value Type	Ranges (Between both train and test)
1	Duration	Length of time duration of the connection	Continuous	Integers	0 - 54451
2	Protocol Type	Protocol used in the connection	Categorical	Strings	
3	Service	Destination network service used	Categorical	Strings	
4	Flag	Status of the connection – Normal or Error	Categorical	Strings	
5	Src Bytes	Number of data bytes transferred from source to destination in single connection	Continuous	Integers	0 - 1379963888
6	Dst Bytes	Number of data bytes transferred from destination to source in single connection	Continuous	Integers	0 - 309937401
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	Binary	Integers	{ 0 , 1 }
8	Wrong Fragment	Total number of wrong fragments in this connection	Discrete	Integers	{ 0,1,3 }
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	Discrete	Integers	0 - 3
10	Hot	Number of “hot" indicators in the content such as: entering a system directory, creating programs and executing programs	Continuous	Integers	0 - 101
11	Num Failed Logins	Count of failed login attempts	Continuous	Integers	0 - 4
12	Logged In	Login Status : 1 if successfully logged in; 0 otherwise	Binary	Integers	{ 0 , 1 }
13	Num Compromised	Number of "compromised” conditions	Continuous	Integers	0 - 7479
14	Root Shell	1 if root shell is obtained; 0 otherwise	Binary	Integers	{ 0 , 1 }
15	Su Attempted	1 if "su root" command attempted or used; 0 otherwise	Discrete (Dataset contains ‘2’ value)	Integers	0 - 2

16	Num Root	Number of "root" accesses or number of operations performed as a root in the connection	Continuous	Integers	0 - 7468
17	Num File Creations	Number of file creation operations in the connection	Continuous	Integers	0 - 100
18	Num Shells	Number of shell prompts	Continuous	Integers	0 - 2
19	Num Access Files	Number of operations on access control files	Continuous	Integers	0 - 9
20	Num Outbound Cmds	Number of outbound commands in an ftp session	Continuous	Integers	{ 0 }
21	Is Hot Logins	1 if the login belongs to the "hot" list i.e., root or admin; else 0	Binary	Integers	{ 0 , 1 }
22	Is Guest Login	1 if the login is a "guest" login; 0 otherwise	Binary	Integers	{ 0 , 1 }
23	Count	Number of connections to the same destination host as the current connection in the past two seconds	Discrete	Integers	0 - 511
24	Srv Count	Number of connections to the same service (port number) as the current connection in the past two seconds	Discrete	Integers	0 - 511
25	Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
26	Srv Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)	Discrete	Floats (hundredths of a decimal)	0 - 1
27	Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
28	Srv Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	Discrete	Floats (hundredths of a decimal)	0 - 1

29	Same Srv Rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
30	Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
31	Srv Diff Host Rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	Discrete	Floats (hundredths of a decimal)	0 - 1
32	Dst Host Count	Number of connections having the same destination host IP address	Discrete	Integers	0 - 255
33	Dst Host Srv Count	Number of connections having the same port number	Discrete	Integers	0 - 255
34	Dst Host Same Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1
35	Dst Host Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1
36	Dst Host Same Src Port Rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
37	Dst Host Srv Diff Host Rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
38	Dst Host Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1

39	Dst Host Srv Serror Rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
40	Dst Host Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1
41	Dst Host Srv Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
42	Class	Classification of the traffic input	Categorical	Strings	
43	Difficulty Level	Difficulty level	Discrete	Integers	0 - 21

Table 3: Features Summary

Appendix B – The Output of Selected Features (ANOVA)

Dataset: Attack	Features selected for attack classes
DoS – ANOVA	['logged_in', 'count', 'serror_rate', 'srv_serror_rate', 'same_srv_rate', 'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'service_http', 'flag_S0', 'flag_SF']
U2R – ANOVA	['urgent', 'hot', 'root_shell', 'num_file_creations', 'num_shells', 'dst_host_count', 'dst_host_srv_count', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'difficulty_level', 'service_ftp_data', 'service_http', 'service_telnet']
R2L – ANOVA	['dst_bytes', 'hot', 'num_failed_logins', 'is_guest_login', 'dst_host_srv_count', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'difficulty_level', 'service_ftp', 'service_ftp_data', 'service_http', 'service_imap4', 'flag_RSTO']
Probe – ANOVA	['logged_in', 'rerror_rate', 'srv_rerror_rate', 'dst_host_srv_count', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'difficulty_level', 'Protocol_type_icmp', 'service_eco_i', 'service_private', 'flag_SF']

Table 3: Output of selected features (ANOVA)