Journal of Research Administration
Society of Research Administrators International

DYNAMIC KNN METHOD FOR PREPROCESSING IN THE INTRUSION DETECTION SYSTEM

Mrs.A. Kaveri
Assistant Professor, Department of Computer Science, KPR College of Arts, Science & Research, Coimbatore. Email: kaverimca.a@gmail.com

Dr. M. Punithavalli
Professor, Department of Computer Applications, Bharathiar University, Coimbatore. Email: punithavalli@buc.edu.in

Abstract: This research article focused an efficient intrusion detection system using data mining techniques. Particularly the proposed system is about how to improvise data preprocessing results ie filling missing values using D-kNN algorithm with optimal k value and data reduction method as minimal maximal normalization in intrusion detection system implemented using KDDCUP 99 benchmark dataset.

Keywords: Data Preprocessing, Data Cleaning, Filling Missing Values, Data Reduction, Data Transformation, IDS, kNN, D-kNN, Min Max Algorithm.

1. Introduction

The need to do research regarding network-security has arrived because the numbers of cyberattacks keep increasing significantly every day. The intrusion detection systems are developed to prevent the applications, information and illegal access to the computers from intrusions. The Intrusion detection systems can differentiate between intrinsic and extrinsic intrusions in the network systems & servers.

Therefore, initiate the warning buzzer if any threat to the security of the above mentioned take place in a network [1].
Intrusion typically means producing malicious and operative infringement. The main function of intrusion detection systems is to identify all kinds of intrusions, so far explored and unexplored hazard; to explore and prevent data from unexplored hazard; and to identify intrusions in a prompt-pattern in a network [2].
The exploratory research was done by Anderson [3] who suggested various ways to test data. His work was to develop the algorithms and procedures for online automated systems. Audit-trails with improved protection and to consider various ways to analyze automated systems were initiated by the Sytek project [4]. These observations played a significant part in acquiring the first factual evidence of the theory stating that the end users can be differentiated from each other by using their usage information [5].

The foundation of real-time IDS provided as the evidence of SRI and Sytek-studies [6]. These systems continuously monitored the behavior of the users irrespective of being normal. The real time IDS depends upon different methods: (1) intrusions both regular and doubted shall be easily noticed by the marked departure from the factual-patterns collected from the operator and (2) The rule-based expert systems are used to track potential susceptibilities and intrusions of the system-targeted security protocols. To assess the IDSs, stability of precision and detection are two main measures applied [7], and to enhance these measures, various analysis in the domain of IDS have taken place in recent years [8]. The rule- based-expert-system and statistical- approach were primarily focused during the research studied in the beginning stages. Yet, it was not precise when applied to large dataset [9]. To overcome this shortcoming, data-mining-approaches [10, 11] and machine learning techniques come into the industry subsequently [12]. The researchers further explored some of the machine-learning-paradigms including Graph based methods [13], Linear Genetic Programming [14], Bayesian Network [15], k-NN [16], K-means-clustering [17], Hidden-Markov-Model [18], Self-organizing-map [19], etc. Machine Learning [20] has the potential to

identify the similarities between characteristics and classes that are identified in the training-data and it can also detect similar subdivision of features by feature-selection and dimensionality-reduction, the data acquired from this can be further used to construct a model to classify the data and to forecast. The data- dimensionality relevant to data-mining and machine-learning has increased rapidly in the past few years that lead to many queries on the education techniques of the present date [21]. The extreme fundamental characteristics make the model become over trained to master, which ultimately results in the deterioration of the performance of the particular model.

2. Existing Work

From the knowledge of Keserwani, Govil, & Pilli, (2023), the IDS types, six benchmark network datasets, high distributed dimensionality reduction methods, and classification techniques are purely based on machine learning and deep learning for intrusion identification. This is because they have the potential to discover the effectiveness and solidity of IDSs. To add to it, a common structure for NIDS has been introduced based on the literature review. In the end, the model for NIDS was developed by applying the said structure.

This model accomplished accuracy rate of 98.11% and detection rate of 97.81% successfully and also has achieved better performance comparing to the other approaches.

The contribution of Amuthaet.al(2022) for protective Network Detection system is a significant application for the protection in the domain of cyber security. It is necessary to select the features manually in traditional methods that use machine learning; which was a definite disadvantage. The concept of deep learning provides IDS structures which is useful to enhance the operation in data security and detection of malicious attacks. One of the greatest advantages of it is its ability to learn sequentially in its NID. The researchers consider the unforecastable importance of the input hidden layer as a significant problem in NID. Hence, they are concerned regarding the confluency and the time taken. The NID Recurrent Neural Network (RNN) method is recommended to integrate NID with Long- short-term-memory (LSTM).

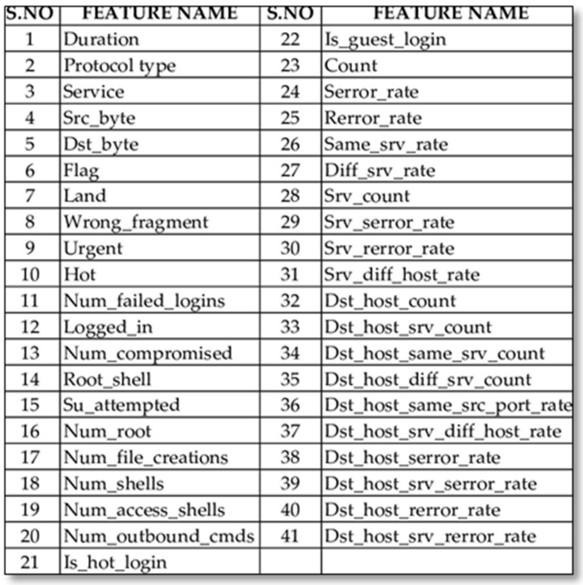
The methodology of Sarkar, Sharma, & Singh, (2023) has the listed advantages; (1) It has two techniques of differentiating intrusions on the two conventional datasets using Machine Learning models. (2) The KDD-Cup99 and NSL KDD datasets are redistributed using data-augmentation. (3) It offers a three
instructional techniques for enhancing identification of intrusion using Multi
Layer Perception (MLP). (4) A cascaded meta specialized classifier architecture has been introduced to differentiate between classes using a specialized one. (5) The dataset's non flagged connections were assessed by almost all the meta-specialists. The classification accuracy ranges to 89.32% and the FPR to 1.95%, and hence this approach is considerably good as it increases the detection quality. (6). The efficient way to forecast are united by maximizing their weights to enhance the capability to detect. On the other hand, the NSL-KDD-dataset reports greater accuracy of 87.63% and a declining 1.68% of FPR of this approach. The design that had improved brilliance and accuracy of network intrusion identification and reduction of wrong alarm, a new deep-neural-network (NDNN) was designed by Jia,Wang& Wang (2019). The NDNN was further developed with four hidden layers which is important to identify and differentiate the intrusion characteristics of the KDD99 and NSL-KDD training data. The reports of the research on KDD99 and NSL-KDD datasets have shown that the NDNN-based method enhances the operations of the IDS and the reports show accuracy rate upto99.9%, which is comparatively greater than many other IDS. Hence this NDNN model is the best choice to make the system more secure.

Addition to this, another model, Spark-Chi-SVM-model, was introduced for intrusion-detection by Othman et.al (2018). In this technique, the Chi-Sq- Selector has been used to select the characteristics, and an intrusion-detection- model has been developed with the help of Support Vector Machine (SVM) classifier on Apache Spark Big Data platform.

To test the model, KDD99 has been used. The comparison of Chi-SVM-classifier with Chi-Logistic-Regression-classifier has also been done by the researcher; the outcomes of which displayed that the Spark-Chi- SVM-model has the greater functionality, consumes less time for training and has higher capability to handle Big Data.

3. Data Set

A benchmark dataset called KDD cup 99 dataset is used to evaluate Intrusion Detection techniques. Since 1999, KDD'99 has been the most wildly used data set for the evaluation of anomaly detection methods. KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. Table 2 shows the 41 features of KDD CUP 99.



S.NO	FEATURE NAME	S.NO	FEATURE NAME
1	Duration	22	Is_guest_login
2	Protocol type	23	Count
3	Service	24	Serror_rate
4	Src_byte	25	Rerror_rate
5	Dst_byte	26	Same_srv_rate
6	Flag	27	Diff_srv_rate
7	Land	28	Srv_count
8	Wrong_fragment	29	Srv_serror_rate
9	Urgent	30	Srv_rerror_rate
10	Hot	31	Srv_diff_host_rate
11	Num_failed_logins	32	Dst_host_count
12	Logged_in	33	Dst_host_srv_count
13	Num_compromised	34	Dst_host_same_srv_count
14	Root_shell	35	Dst_host_diff_srv_count
15	Su_attempted	36	Dst_host_same_src_port_rate
16	Num_root	37	Dst_host_srv_diff_host_rate
17	Num_file_creations	38	Dst_host_serror_rate
18	Num_shells	39	Dst_host_srv_serror_rate
19	Num_access_shells	40	Dst_host_rerror_rate
20	Num_outbound_cmds	41	Dst_host_srv_rerror_rate
21	Is_hot_login		

Table1. KDD CUP 99 - 41 Features

4. Proposed Method

4.1 Preprocessing of the Data.

Transforming and normalizing of data is executed on the 20% of the NSLKDD dataset in the process of data preprocessing. It has an absolute advantage of better predictive performance which is achieved by better exposure of the basic design of the data to the learning- algorithm.

4.1.1 Transformation of the Data.

In simple terms, the operation of transformation is used to convert the nominal values to numeric values. Some differentiation methods are unable to control the nominal characteristics and the IDSs are considered as the classification issue [66]. As part of the transformation function, the attributes of KDDCUP 99 20% dataset, which includes the type of protocol, service, and flags.

These are being transformed from the nominal value to its numeric equivalent and the entire numeric values for the classification process are stored in the final KDD_CUP 20% dataset.

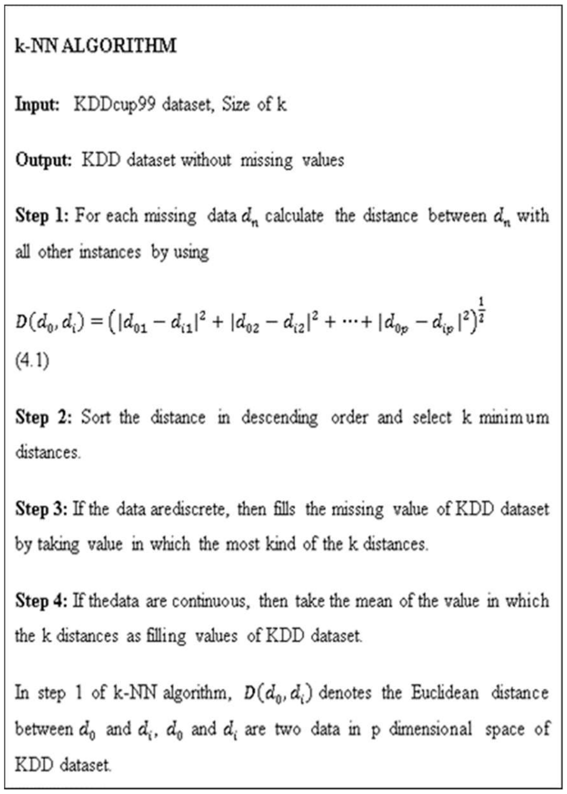
4.1.2. Normalization of the Data

Data normalization is one of the most important paradigms, especially in the field of classification. In the linear classification approaches, the instances are considered as a multidimensional-area. Some of the objective functions do not work in accordance without normalization. The reason may be the wide variations of data. By using the normalization function, a greater enhancement is achieved in terms of accuracy and speed. Here is an example to prove that, if the value of a certain characteristics has broad scope, then the distinct feature is able to control the range within the points. The importance of normalization can be understood by the given example. The normalization of the numeric characteristics is compulsory in order for all the characteristics to provide with close proportions to the actual distance. Minimal Maximal Normalization Approach is being used in the dataset for this study.

4.1.3 Filling Missing Values

The D-kNN algorithm fills the missing values with optimal k values for each missing data. Because it is impractical for applying a fixed k value for all test samples. The optimal k values are selected through kTree method. For fast learning an optimal k- value for each test sample kTree method is introduced. It included a training stage into the conventional kNN method and thus returning a training model i.e., building a decision tree called kTree.

Particularly in the training stage, kTree is used to reconstruct each training sample by all training samples through designing a sparse-based reconstruction model, which outputs an optimal-k-value for each training sample. After that a decision tree is constructed using training samples and their corresponding optimal k-values, i.e., regarding the learned optimal k-value of each training sample as the label. The training stage is offline and each leaf node stores an optimal k value in the constructed kTree. In the test stage, given a test sample, first search for the constructed kTree from the root node to a leaf node, whose optimal k value is assigned to this test sample so that using traditional kNN classification to assign it with a label by the majority rule.



k-NN ALGORITHM

Input: KDDcup99 dataset, Size of k

Output: KDD dataset without missing values

Step 1: For each missing data d_n calculate the distance between d_n with all other instances by using

$$D(d_0, d_i) = \left(|d_{01} - d_{i1}|^2 + |d_{02} - d_{i2}|^2 + \cdots + |d_{0p} - d_{ip}|^2\right)^{\frac{1}{2}}$$
(4.1)

Step 2: Sort the distance in descending order and select k minimum distances.

Step 3: If the data arediscrete, then fills the missing value of KDD dataset by taking value in which the most kind of the k distances.

Step 4: If thedata are continuous, then take the mean of the value in which the k distances as filling values of KDD dataset.

In step 1 of k-NN algorithm, $D(d_0, d_i)$ denotes the Euclidean distance between d_0 and d_i, d_0 and d_i are two data in p dimensional space of KDD dataset.

$$\min_{W} \sum_{i=1}^{n} \|Xw_i - x_i\|_2^2$$

$$= \min_{W} \|XW - X\|_F^2 \qquad (1)$$

In Eq. (1), $W = [w_1, w_2, \ldots w_n] \in$

In Eq. (1), $W = [w1, w2, \dots wn] \in \mathbb{R}n\times n$ denotes the reconstruction coefficient or the correlation between training samples and themselves.
The relation between two training samples in X is imposed to be reflected in the relation between their predictions by defining the following embedding function:

$$\frac{1}{2}\sum_{i,j} s_{ij} \|x^i W - x^j W\|_2^2 \qquad (2)$$

The reconstruction processis interpreted to learn the optimal k values for training samples. The training samples are denoted as $X \in \mathbb{R}d\times n = [x1, x2, \dots xn]$, where d number of training features and n denotes the number of training samples. Each training sample xi is reconstructed with the goal that the distance between
Xwi and xi is as small as possible. $wi \in \mathbb{R}n$denotes the reconstruction coefficient matrix.
This can be achieved by using a least square loss function which is given as
follows,

$$\min_W \|XW - X\|_F^2 + \rho_1 \|W\|_1 + \rho_2 R(W), W \geq 0 \qquad (3)$$

In Eq. (2),sij denotes an element in the
feature similarity matrix S which encodes the relation between feature vectors. As for the similarity matrix$S$, a data adjacency graph is constructed by regarding each feature as a node and using k-NNs along with a heat kernel function $f(a, b)$ to compute the similarities. The final objective function is defined for the reconstruction process which is given as follows,

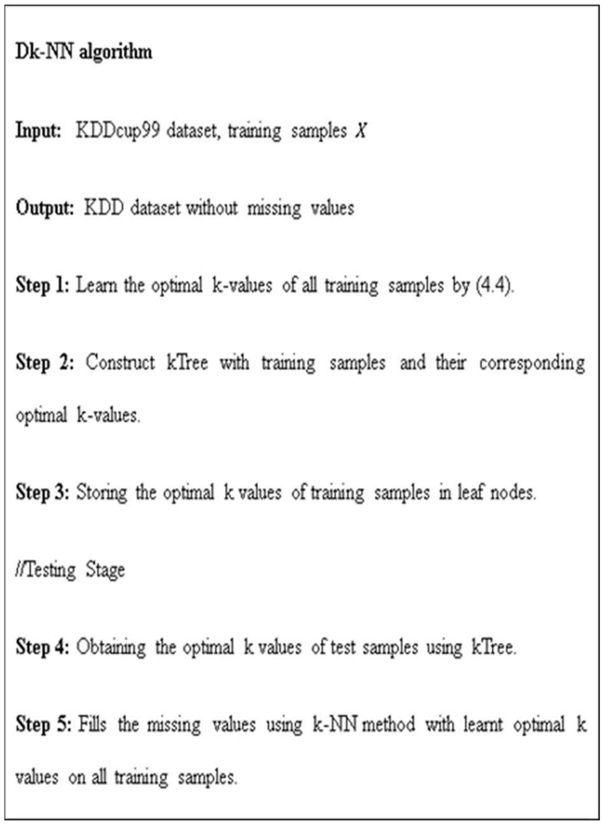
$$\min_W \|XW - X\|_F^2 + \rho_1 \|W\|_1 + \rho_2 R(W), W \geq 0 \qquad (3)$$

In Eq. (3), $\rho1$ and $\rho2$ are tuning parameters and $R(W)$ denotes the regularization term. As the objective function is convex, the W satisfying the
(3) is a global optimum solution. The optimal solution $W*$ the weight matrix or the correlation between training samples and themselves is obtained after optimizing the objective function. The element wij of $W*$denotes the correlation
between the ith training sample and jth
training sample.

The positive weight indicates that the ith training sample and jth training sample are positively correlated and the negative weight indicates that their correlation is negative. After obtaining the optimal k values using kTree , take the average value in optimal k distances and it filled as a missing value.
The Dk-NN algorithm fills the missing value with optimal k values for each missing data. The Optimal K values are selected by kTree method which is introduced for fast learning Training Stage.

Training Stage

kTree is used to reconstruct each training sample by all training samples through designing a Sparse based reconstruction model which outputs a optimal k value. The learned optimal k value of each sample has label.



```
Dk-NN algorithm

Input: KDDcup99 dataset, training samples $X$

Output: KDD dataset without missing values

Step 1: Learn the optimal k-values of all training samples by (4.4).

Step 2: Construct kTree with training samples and their corresponding
optimal k-values.

Step 3: Storing the optimal k values of training samples in leaf nodes.

//Testing Stage

Step 4: Obtaining the optimal k values of test samples using kTree.

Step 5: Fills the missing values using k-NN method with learnt optimal k
values on all training samples.
```

Figure 2: D-kNN Algorithm Testing Stage

Given test sample first search for the constructed kTree from the root to leaf node. Then Fill the missing value with kNN method with learnt optimal k values on all training samples.

5 Performance Metrics

Evaluating the performance of a proposed system is important step to build the effective system. To evaluate the system’s performance or quality, different types of metrics are used. The performance metrics are also known as evaluation

metrics. These performance metrics helps to understand how well our proposed system has performed for the given data. In this method, we can improvise the system's performance by tuning its various
parameters.

5.3. Accuracy
Accuracy is the ratio of the total number of correct predictions to the actual dataset size. It is calculated as,
$TP + TN$

Metrics are used to monitor and measure the system's performance during training and testing phase.

5.1 True Positive Rate (TPR)

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

$$\rightarrow \quad (6)$$

TPR measures the proportion of actual attacks are correctly identified as attacks. It can be calculated as,

Precision is calculated based on the Intrusion Detection and malicious node detection at true positive and false positive
predictions.

$TPR = True\ Postive\ (TP)$
$(TP{+}False\ Negative(FN))$

$$Precision = \frac{TP}{TP+FP} \quad \rightarrow \quad (7)$$

Where, TP is the number of instances correctly predicted as attacks FN is the number of instances wrongly predicted as non-attacks

5.2 False Positive Rate (FPR)

FPR measures the proportion of

5.5 Recall

Recall is calculated based on the Intrusion Detection and malicious node detection at true positive and false negative predictions.

normal traffic is identified as an attack. It can be calculated as,

$Recall =$ TP
$TP+FN$

5.6 F - Measure

 (8)

$FPR = False\ Positive\ (FP)$
$(FP+True\ Negative\ (TN))$

 (5)

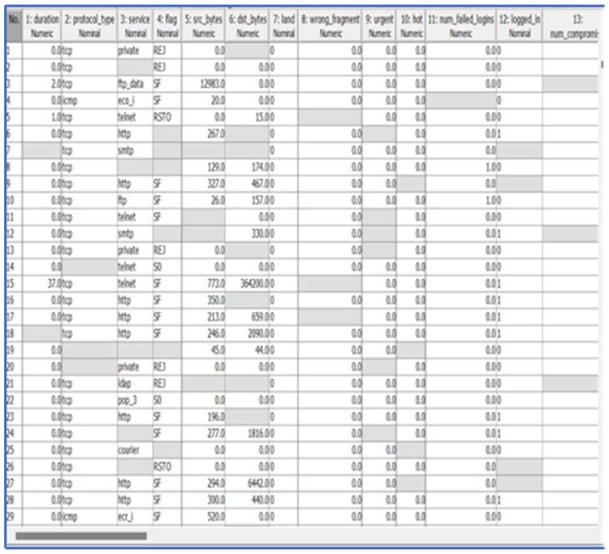
F-measure is the harmonic mean of precision and recall. It is calculated as,

Where, FP is the number of instances wrongly predicted as attacks TN is the number of instances wrongly predicted as non-attacks

$F - measure = 2\times Precision\times Recall$
$Precision+Recall$
(9)

6. Results and Discussion

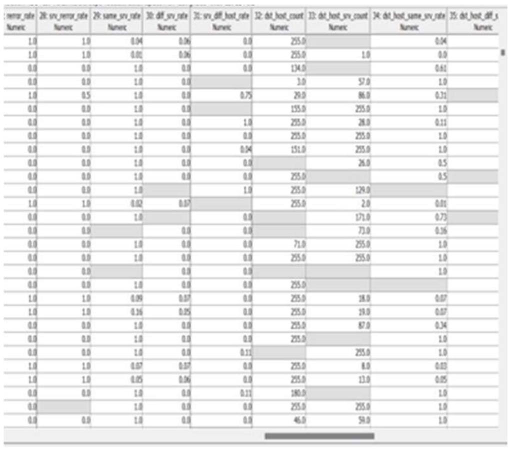
The proposed method is implemented by python. There are 41 features are in KDDCUP 99 dataset.



No.	1: duration Numeric	2: protocol_type Nominal	3: service Nominal	4: flag Nominal	5: src_bytes Numeric	6: dst_bytes Numeric	7: land Nominal	8: wrong_fragment Numeric	9: urgent Numeric	10: hot Numeric	11: num_failed_logins Numeric	12: logged_in Nominal	13: num_compromi
1	0.0	tcp	private	REJ	0.0		0	0.0	0.0	0.0	0.0	0	
2	0.0	tcp		REJ	0.0	0.0	0	0.0	0.0	0.0	0.0	0	
3	2.0	tcp	ftp_data	SF	12983.0	0.0	0	0.0	0.0	0.0	0.0	0	
4	0.0	icmp	eco_i	SF	20.0	0.0	0	0.0	0.0	0.0		0	
5	1.0	tcp	telnet	RSTO	0.0	15.0	0		0.0	0.0	0.0	0	
6	0.0	tcp	http		267.0		0	0.0		0.0	0.0	1	
7		tcp	smtp				0	0.0	0.0	0.0	0.0		
8	0.0	tcp			129.0	174.0	0	0.0	0.0	0.0	1.0	0	
9	0.0	tcp	http	SF	327.0	467.0	0	0.0	0.0		0.0		
10	0.0	tcp	ftp	SF	26.0	157.0	0	0.0	0.0	0.0	1.0	0	
11	0.0	tcp	telnet	SF		0.0	0	0.0		0.0	0.0	0	
12	0.0	tcp	smtp			330.0	0	0.0		0.0	0.0	1	
13	0.0	tcp	private	REJ	0.0		0	0.0		0.0	0.0	0	
14	0.0		telnet	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0	
15	37.0	tcp	telnet	SF	773.0	364200.0	0		0.0	0.0	0.0	1	
16	0.0	tcp	http	SF	350.0		0	0.0	0.0	0.0	0.0	1	
17	0.0	tcp	http	SF	213.0	659.0	0		0.0	0.0	0.0	1	
18		tcp	http	SF	246.0	2090.0	0	0.0	0.0	0.0	0.0	1	
19	0.0				45.0	44.0	0	0.0	0.0		0.0	0	
20	0.0		private	REJ	0.0	0.0	0	0.0		0.0	0.0	0	
21	0.0	tcp	ldap	REJ			0	0.0	0.0	0.0	0.0	0	
22	0.0	tcp	pop_3	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0	
23	0.0	tcp	http	SF	196.0		0	0.0	0.0	0.0	0.0	1	
24	0.0	tcp		SF	277.0	1816.0	0	0.0		0.0	0.0	1	
25	0.0	tcp	courier		0.0	0.0	0	0.0	0.0		0.0	0	
26	0.0	tcp		RSTO	0.0	0.0	0	0.0	0.0	0.0	0.0		
27	0.0	tcp	http	SF	294.0	6442.0	0	0.0	0.0		0.0		
28	0.0	tcp	http	SF	300.0	440.0	0	0.0	0.0	0.0	0.0	1	
29	0.0	icmp	ecr_i	SF	520.0	0.0	0	0.0	0.0	0.0	0.0	0	

Figure 1. Training Data set with Missing Values

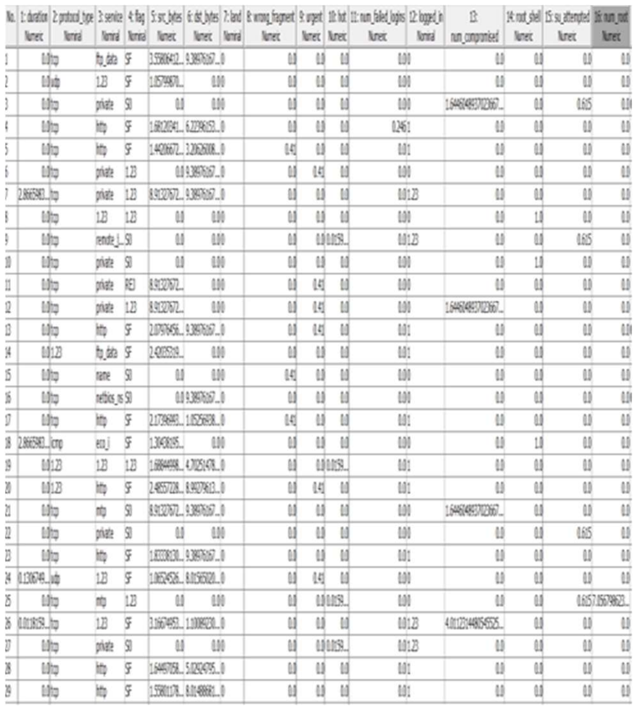
Figure 1 shows the missing data set that in duration_Numeric field two values are missed, Protocol_type 3 values are missed, in service normal 3 values are missed, in flag normal 5 values are missed; src_bytes numeric 3 values are missed and so on. Figure 2 shows the missing values in remaining features.



rerror_rate Numeric	28: srv_rerror_rate Numeric	29: same_srv_rate Numeric	30: diff_srv_rate Numeric	31: srv_diff_host_rate Numeric	32: dst_host_count Numeric	33: dst_host_srv_count Numeric	34: dst_host_same_srv_rate Numeric	35: dst_host_diff_s Numeric
1.0	1.0	0.04	0.06	0.0	255.0		0.04	
1.0	1.0	0.01	0.06	0.0	255.0	1.0	0.0	
0.0	0.0	1.0	0.0	0.0	134.0		0.61	
0.0	0.0	1.0	0.0		3.0	57.0	1.0	
1.0	0.5	1.0	0.0	0.75	29.0	86.0	0.31	
0.0	0.0	1.0	0.0		155.0	255.0	1.0	
0.0	0.0	1.0	0.0	1.0	255.0	28.0	0.11	
0.0	0.0	1.0	0.0	0.0	255.0	255.0	1.0	
0.0	0.0	1.0	0.0	0.04	151.0	255.0	1.0	
0.0	0.0	1.0	0.0	0.0		26.0	0.5	
0.0	0.0	1.0	0.0	0.0	255.0		0.5	
0.0	0.0	1.0		1.0	255.0	129.0		
1.0	1.0	0.02	0.07		255.0	2.0	0.01	
0.0	0.0	1.0		0.0		171.0	0.73	
0.0	0.0		0.0	0.0		73.0	0.16	
0.0	0.0	1.0	0.0	0.0	71.0	255.0	1.0	
0.0	0.0	1.0	0.0	0.0	255.0	255.0	1.0	
0.0	0.0		0.0	0.0			1.0	
0.0	0.0	1.0	0.0	0.0	255.0			
1.0	1.0	0.09	0.07	0.0	255.0	18.0	0.07	
1.0	1.0	0.16	0.05	0.0	255.0	19.0	0.07	
0.0	0.0	1.0	0.0	0.0	255.0	87.0	0.34	
0.0	0.0	1.0	0.0	0.0	255.0		1.0	
0.0	0.0	1.0	0.0	0.11		255.0	1.0	
1.0	1.0	0.07	0.07	0.0	255.0	8.0	0.03	
1.0	1.0	0.05	0.06	0.0	255.0	13.0	0.05	
0.0	0.0	1.0	0.0	0.11	180.0		1.0	
0.0		1.0	0.0	0.0	255.0	255.0	1.0	
0.0	0.0	1.0	0.0	0.0	46.0	59.0	1.0	

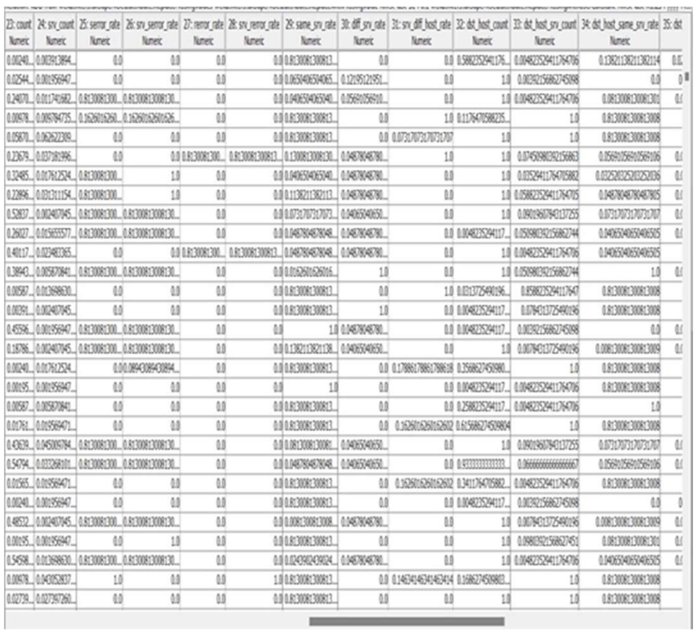
Figure 2. Training Data set with Missing Values

Figure 3 and 4 show the filling of missing values by proposed method. It Fills the missing values using DkNN method with learned optimal k values on all training samples.



No.	1: duration Numeric	2: protocol_type Nominal	3: service Nominal	4: flag Nominal	5: src_bytes Numeric	6: dst_bytes Numeric	7: land Nominal	8: wrong_fragment Numeric	9: urgent Numeric	10: hot Numeric	11: num_failed_logins Numeric	12: logged_in Nominal	13: num_compromised	14: root_shell Numeric	15: su_attempted Numeric	16: num_root Numeric
1	0.0	tcp	ftp_data	SF	3.5580641 2...	9.3897616 7...	0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0
2	0.0	udp	1.23	SF	1.0579987 0...	0.0	0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0
3	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0	1.6446048937023667...	0.0	0.615	0.0
4	0.0	tcp	http	SF	1.6812074 1...	6.2239615 3...	0	0.0	0.0	0.0	0.246	1	0.0	0.0	0.0	0.0
5	0.0	tcp	http	SF	1.4420667 2...	3.2062600 8...	0	0.41	0.0	0.0	0.0	1	0.0	0.0	0.0	0.0
6	0.0	tcp	private	1.23	0.0	9.3897616 7...	0	0.0	0.41	0.0	0.0	0	0.0	0.0	0.0	0.0
7	2.8665983...	tcp	private	1.23	8.9132767 2...	9.3897616 7...	0	0.0	0.0	0.0	0.0	1.23	0.0	0.0	0.0	0.0
8	0.0	tcp	1.23	1.23	0.0	0.0	0	0.0	0.0	0.0	0.0	0	0.0	1.0	0.0	0.0
9	0.0	tcp	remote_j...	S0	0.0	0.0	0	0.0	0.0	0.0159...	0.0	1.23	0.0	0.0	0.615	0.0
10	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0	0.0	1.0	0.0	0.0
11	0.0	tcp	private	REJ	8.9132767 2...	0.0	0	0.0	0.41	0.0	0.0	0	0.0	0.0	0.0	0.0
12	0.0	tcp	private	1.23	8.9132767 2...	0.0	0	0.0	0.41	0.0	0.0	0	1.6446048937023667...	0.0	0.0	0.0
13	0.0	tcp	http	SF	2.0797645 6...	9.3897616 7...	0	0.0	0.41	0.0	0.0	1	0.0	0.0	0.0	0.0
14	0.0	1.23	ftp_data	SF	2.4203551 9...	0.0	0	0.0	0.0	0.0	0.0	1	0.0	0.0	0.0	0.0
15	0.0	tcp	name	S0	0.0	0.0	0	0.41	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0
16	0.0	tcp	netbios_ns	S0	0.0	9.3897616 7...	0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0
17	0.0	tcp	http	SF	2.1739698 3...	1.0525693 8...	0	0.41	0.0	0.0	0.0	1	0.0	0.0	0.0	0.0
18	2.8665983...	icmp	eco_i	SF	1.3045819 5...	0.0	0	0.0	0.0	0.0	0.0	0	0.0	1.0	0.0	0.0
19	0.0	1.23	1.23	1.23	1.6864499 8...	4.7025147 8...	0	0.0	0.0	0.0159...	0.0	1	0.0	0.0	0.0	0.0
20	0.0	1.23	http	SF	2.4855722 8...	8.9927961 3...	0	0.0	0.41	0.0	0.0	1	0.0	0.0	0.0	0.0
21	0.0	tcp	smtp	S0	8.9132767 2...	9.3897616 7...	0	0.0	0.0	0.0	0.0	0	1.6446048937023667...	0.0	0.0	0.0
22	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.615	0.0
23	0.0	tcp	http	SF	1.8330813 0...	9.3897616 7...	0	0.0	0.0	0.0	0.0	1	0.0	0.0	0.0	0.0
24	0.1306749...	udp	1.23	SF	1.0652452 6...	8.0156502 0...	0	0.0	0.41	0.0	0.0	0	0.0	0.0	0.0	0.0
25	0.0	tcp	smtp	1.23	0.0	0.0	0	0.0	0.0	0.0159...	0.0	0	0.0	0.0	0.615	7.05679823...
26	0.0118159...	tcp	1.23	SF	3.1667495 3...	1.1008923 0...	0	0.0	0.0	0.0	0.0	1.23	4.0112314480545525...	0.0	0.0	0.0
27	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0159...	0.0	1.23	0.0	0.0	0.0	0.0
28	0.0	tcp	http	SF	1.6449705 8...	5.0292479 5...	0	0.0	0.0	0.0	0.0	1	0.0	0.0	0.0	0.0
29	0.0	tcp	http	SF	1.5580117 8...	8.0148868 1...	0	0.0	0.0	0.0	0.0	1	0.0	0.0	0.0	0.0

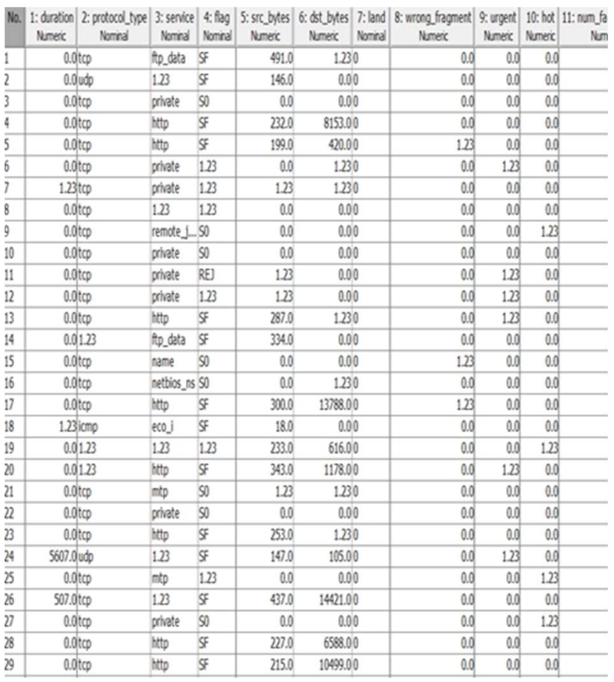
Figure 3. Missing Values Replaced by DkNN



23: count Numeric	24: srv_count Numeric	25: serror_rate Numeric	26: srv_serror_rate Numeric	27: rerror_rate Numeric	28: srv_rerror_rate Numeric	29: same_srv_rate Numeric	30: diff_srv_rate Numeric	31: srv_diff_host_rate Numeric	32: dst_host_count Numeric	33: dst_host_srv_count Numeric	34: dst_host_same_srv_rate Numeric	35: dst
0.00240...	0.00391384...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.0	0.5882352941176...	0.004823529411764706	0.1382113821138214	0.0
0.02544...	0.001956947...	0.0	0.0	0.0	0.0	0.0650406504065...	0.12195121951...	0.0	1.0	0.003921568627450098	0.0	0
0.24070...	0.011741682...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0406504065040...	0.05691056910...	0.0	1.0	0.004823529411764706	0.0813008130081301	0.0
0.00978...	0.009784735...	0.1626016260...	0.16260162601626...	0.0	0.0	0.8130081300813...	0.0	1.0	0.1176470588235...	1.0	0.8130081300813008	
0.05870...	0.062622309...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.07317073170731707	1.0	1.0	0.8130081300813008	
0.23679...	0.037181996...	0.0	0.0	0.8130081300...	0.8130081300813...	0.13008130081300...	0.04878048780...	1.0	1.0	0.07450980392156863	0.056910569105691066	0.0
0.32485...	0.017612524...	0.8130081300...	1.0	0.0	0.0	0.0406504065040...	0.04878048780...	0.0	1.0	0.03529411764705882	0.032520325203252036	0.0
0.22896...	0.031311154...	0.8130081300...	1.0	0.0	0.0	0.1138211382113...	0.04878048780...	0.0	1.0	0.058823529411764705	0.04878048780487805	0.0
0.52837...	0.002407045...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0731707317073...	0.04065040650...	0.0	1.0	0.09019607843137255	0.07317073170731707	0.0
0.26027...	0.015655577...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0487804878048...	0.04878048780...	0.0	0.0048235294117...	0.050980392156862744	0.04065040650406505	0.0
0.40117...	0.023483365...	0.0	0.0	0.8130081300...	0.8130081300813...	0.0487804878048...	0.04878048780...	0.0	1.0	0.004823529411764706	0.04065040650406505	
0.38943...	0.005870841...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0162601626016...	1.0	0.0	1.0	0.050980392156862744	1.0	0.0
0.00587...	0.013698630...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	1.0	0.0313725490196...	0.8588235294117647	0.8130081300813008	
0.00391...	0.002407045...	0.0	0.0	0.0	0.0	0.8130081300813...	1.0	0.0	0.0048235294117...	0.0784313725490196	0.8130081300813008	
0.45596...	0.001956947...	0.8130081300...	0.81300813008130...	0.0	0.0	1.0	0.04878048780...	0.0	0.0048235294117...	0.003921568627450098	0.0	0.0
0.18786...	0.002407045...	0.8130081300...	0.81300813008130...	0.0	0.0	0.1382113821138...	0.04065040650...	0.0	1.0	0.00784313725490196	0.008130081300813009	0.0
0.00240...	0.017612524...	0.0	0.08943089430894...	0.0	0.0	0.8130081300813...	0.0	0.17886178861788618	0.3568627450980...	1.0	0.8130081300813008	
0.00195...	0.001956947...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0048235294117...	0.004823529411764706	0.8130081300813008	
0.00587...	0.005870841...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.0	0.2588235294117...	0.004823529411764706	1.0	
0.01761...	0.019569471...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.1626016260162602	0.615686274509804	1.0	0.8130081300813008	
0.43639...	0.045009784...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0813008130081...	0.04065040650...	0.0	1.0	0.09019607843137255	0.07317073170731707	0.0
0.54794...	0.033268101...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0487804878048...	0.04065040650...	0.0	0.9333333333333...	0.06666666666666667	0.056910569105691066	0.0
0.01565...	0.019569471...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.1626016260162602	0.3411764705882...	0.004823529411764706	0.8130081300813008	
0.00240...	0.001956947...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.0	0.0048235294117...	0.003921568627450098	0.0	0
0.48532...	0.002407045...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0081300813008...	0.04878048780...	0.0	1.0	0.00784313725490196	0.008130081300813009	0.0
0.00195...	0.001956947...	0.0	1.0	0.0	0.0	0.8130081300813...	0.0	0.0	1.0	0.09803921568627451	0.0813008130081301	0.0
0.54598...	0.013698630...	0.8130081300...	0.81300813008130...	0.0	0.0	0.0243902439024...	0.04878048780...	0.0	1.0	0.004823529411764706	0.04065040650406505	0.0
0.00978...	0.043052837...	1.0	0.0	0.0	1.0	0.8130081300813...	0.0	0.14634146341463414	0.1686274509803...	1.0	0.8130081300813008	
0.02739...	0.027397260...	0.0	0.0	0.0	0.0	0.8130081300813...	0.0	0.0	1.0	1.0	0.8130081300813008	

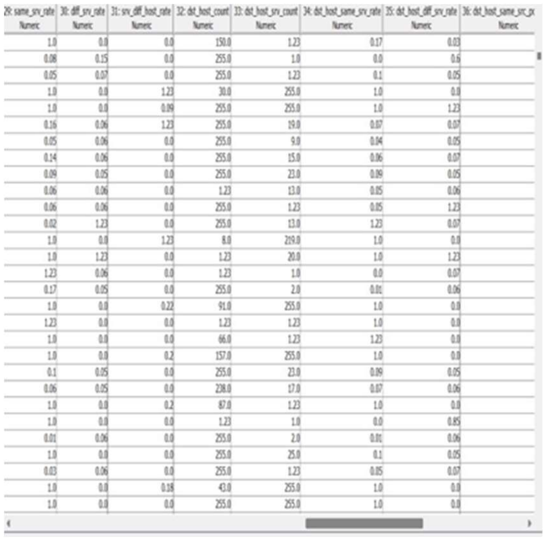
Figure 4. Missing values Replaced by DkNN

After the filling of missing values normalization can make it easier to interpret the results. It is used to convert the inputs will be on a common scale. Figure 5 and 6 shows the conversion of normalization.



No.	1: duration	2: protocol_type	3: service	4: flag	5: src_bytes	6: dst_bytes	7: land	8: wrong_fragment	9: urgent	10: hot	11: num_fa
	Numeric	Nominal	Nominal	Nominal	Numeric	Numeric	Nominal	Numeric	Numeric	Numeric	Num
1	0.0	tcp	ftp_data	SF	491.0	1.23	0	0.0	0.0	0.0	
2	0.0	udp	1.23	SF	146.0	0.0	0	0.0	0.0	0.0	
3	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	
4	0.0	tcp	http	SF	232.0	8153.0	0	0.0	0.0	0.0	
5	0.0	tcp	http	SF	199.0	420.0	0	1.23	0.0	0.0	
6	0.0	tcp	private	1.23	0.0	1.23	0	0.0	1.23	0.0	
7	1.23	tcp	private	1.23	1.23	1.23	0	0.0	0.0	0.0	
8	0.0	tcp	1.23	1.23	0.0	0.0	0	0.0	0.0	0.0	
9	0.0	tcp	remote_j...	S0	0.0	0.0	0	0.0	0.0	1.23	
10	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	
11	0.0	tcp	private	REJ	1.23	0.0	0	0.0	1.23	0.0	
12	0.0	tcp	private	1.23	1.23	0.0	0	0.0	1.23	0.0	
13	0.0	tcp	http	SF	287.0	1.23	0	0.0	1.23	0.0	
14	0.0	1.23	ftp_data	SF	334.0	0.0	0	0.0	0.0	0.0	
15	0.0	tcp	name	S0	0.0	0.0	0	1.23	0.0	0.0	
16	0.0	tcp	netbios_ns	S0	0.0	1.23	0	0.0	0.0	0.0	
17	0.0	tcp	http	SF	300.0	13788.0	0	1.23	0.0	0.0	
18	1.23	icmp	eco_i	SF	18.0	0.0	0	0.0	0.0	0.0	
19	0.0	1.23	1.23	1.23	233.0	616.0	0	0.0	0.0	1.23	
20	0.0	1.23	http	SF	343.0	1178.0	0	0.0	1.23	0.0	
21	0.0	tcp	mtp	S0	1.23	1.23	0	0.0	0.0	0.0	
22	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	
23	0.0	tcp	http	SF	253.0	1.23	0	0.0	0.0	0.0	
24	5607.0	udp	1.23	SF	147.0	105.0	0	0.0	1.23	0.0	
25	0.0	tcp	mtp	1.23	0.0	0.0	0	0.0	0.0	1.23	
26	507.0	tcp	1.23	SF	437.0	14421.0	0	0.0	0.0	0.0	
27	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	1.23	
28	0.0	tcp	http	SF	227.0	6588.0	0	0.0	0.0	0.0	
29	0.0	tcp	http	SF	215.0	10499.0	0	0.0	0.0	0.0	

Figure 5. After Normalization by Min Max Method



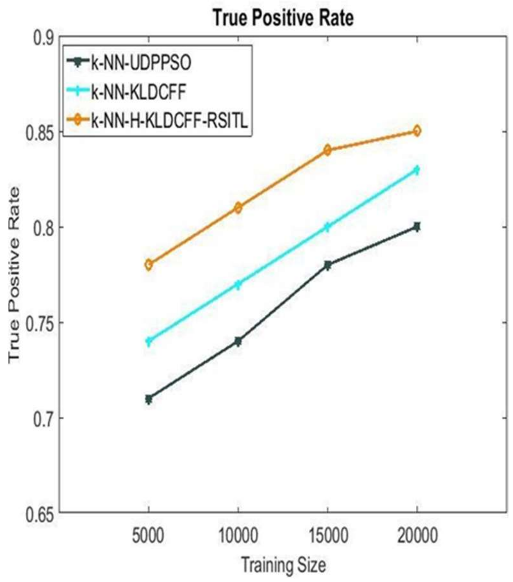
29: same_srv_rate	30: diff_srv_rate	31: srv_diff_host_rate	32: dst_host_count	33: dst_host_srv_count	34: dst_host_same_srv_rate	35: dst_host_diff_srv_rate	36: dst_host_same_src_p
Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
1.0	0.0	0.0	150.0	1.23	0.17	0.03	
0.08	0.15	0.0	255.0	1.0	0.0	0.6	
0.05	0.07	0.0	255.0	1.23	0.1	0.05	
1.0	0.0	1.23	30.0	255.0	1.0	0.0	
1.0	0.0	0.09	255.0	255.0	1.0	1.23	
0.16	0.06	1.23	255.0	19.0	0.07	0.07	
0.05	0.06	0.0	255.0	9.0	0.04	0.05	
0.14	0.06	0.0	255.0	15.0	0.06	0.07	
0.09	0.05	0.0	255.0	23.0	0.09	0.05	
0.06	0.06	0.0	1.23	13.0	0.05	0.06	
0.06	0.06	0.0	255.0	1.23	0.05	1.23	
0.02	1.23	0.0	255.0	13.0	1.23	0.07	
1.0	0.0	1.23	8.0	219.0	1.0	0.0	
1.0	1.23	0.0	1.23	20.0	1.0	1.23	
1.23	0.06	0.0	1.23	1.0	0.0	0.07	
0.17	0.05	0.0	255.0	2.0	0.01	0.06	
1.0	0.0	0.22	91.0	255.0	1.0	0.0	
1.23	0.0	0.0	1.23	1.23	1.0	0.0	
1.0	0.0	0.0	66.0	1.23	1.23	0.0	
1.0	0.0	0.2	157.0	255.0	1.0	0.0	
0.1	0.05	0.0	255.0	23.0	0.09	0.05	
0.06	0.05	0.0	238.0	17.0	0.07	0.06	
1.0	0.0	0.2	87.0	1.23	1.0	0.0	
1.0	0.0	0.0	1.23	1.0	0.0	0.85	
0.01	0.06	0.0	255.0	2.0	0.01	0.06	
1.0	0.0	0.0	255.0	25.0	0.1	0.05	
0.03	0.06	0.0	255.0	1.23	0.05	0.07	
1.0	0.0	0.18	43.0	255.0	1.0	0.0	
1.0	0.0	0.0	255.0	255.0	1.0	0.0	

Figure 6. After Normalization by Min Max Method

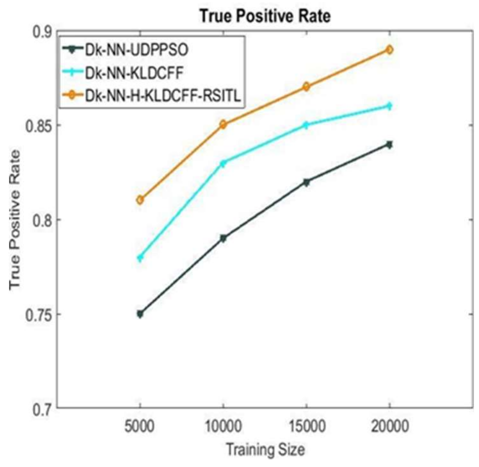
True Positive Ratio(TPR) of k-NN with UDPPSO, KLDCFF and H- KLDCFF-RSITL and Dk-NN with DPPSO, KLDCFF and H-KLDCFF - RSITL is measured and their results shown below in Table 3.

Training Size	k-NN			Dk-NN		
	UDPPSO	KLDCFF	H-KLDCFF-RSITL	UDPPSO	KLDCFF	H-KLDCFF-RSITL
5000	0.71	0.74	0.78	0.75	0.78	0.81
10000	0.74	0.77	0.81	0.79	0.83	0.85
15000	0.78	0.8	0.84	0.82	0.85	0.87
20000	0.8	0.83	0.85	0.84	0.86	0.89

Table 3.TPR vs. Training Size



(a)



(b)

Figure 7 Evaluation of TPR. (a) TPR with k-NN, (b) TPR with Dk-NN

Figure 7 shows the TPR of different pre-processing technique with different subspace selection algorithms for Intrusion Detection. When the training size is 20000, the TPR of k-NN-H-KLDCFF-
RSITL is 2.4% greater than k-NN- KLDCFF and 6.25% greater than k-NN- UDPPSO. When the training size is 20000, the TPR of Dk-NN-H-KLDCFF-RSITL is
3.48% greater than Dk-NN- KLDCFF and 5.95% greater than k-NN- UDPPSO.
If the training size is 20000, then the TPR of Dk-NN-H-KLDCFF-RSITL is
4.71% greater than k-NN-H-KLDCFF- RSITL.
From this analysis, it is proved that the proposed Dk-NN has high TPR than the kNN and the H-KLDCFF-RSITL has high TPR than other feature and sample selection algorithms.

7. Conclusion

The proposed framework, shown by experiment result, provided higher accuracy than other similar work and traditional ML methods, especially in the case of multiclass classifications. In the upcoming time, effort will be made to improve the two functions, the restore function and retraining function, turning retraining function into adaptive update so as to reduce manual intervention, whereas getting traceable effect in restore function. Furthermore, this research will serve as the basis for further research and investigation to enable the development of effective IDSs that can be used in a complex network environment. From this analysis, it is proved that the proposed Dk-NN has high TPR than the kNN and the H-KLDCFF- RSITL has high TPR than other feature and sample selection algorithms.

References

[1]. Keserwan, P. K., Govil, M. C., & Pilli,
E. S. (2023). An effective NIDS framework based on a comprehensive survey of feature optimization and classification techniques. Neural Computing and Applications, 35(7), 4993-5013.
[2]. Amutha, S., Kavitha, R., Srinivasan, R., & Kavitha, M. (2022, January). Secure network intrusion detection system using NID-RNN based Deep Learning. In 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI) (pp. 1-5). IEEE.
[3]. Sarkar, A., Sharma, H. S., & Singh, M.
M. (2023). A supervised machine learning- based solution for efficient network intrusion detection using ensemble learning based on hyper parameter optimization. International Journalof Information Technology, 15(1), 423-434.
[4]. Jia, Y., Wang, M., & Wang, Y. (2019).
Network intrusion detection algorithm based on deep neural network. IET Information Security, 13(1), 48-53.
[5]. Othman, S. M., Ba-Alwi, F. M., Alsohybe, N. T., & Al-Hashida, A. Y. (2018). Intrusion detection model using machine learning algorithm on Big Data environment.Journal of big

data,5(1),1-12. [6]. M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," Expert Systems with Applications, vol. 141, Article ID 112963, 2019.
[7]. H. He, X. Sun, H. He, G. Zhao, L. He,
and J. Ren, "A novel multimodal- sequential approach based on multi-view features for network intrusion detection," IEEE Access, vol. 7, pp. 183207–183221, 2019.
[8]. P. Sun, P. Liu, Q. Li et al., "DL-IDS:
extracting features using CNN-LSTM hybrid network for intrusion detection system,"Security and Communication Networks, vol. 2020, Article ID 8890306,
11 pages, 2020.
[9]. M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi,"Deep learning approach combining sparse autoencoder withsvm for network intrusion detection," IEEE Access, vol. 6,pp. 52843–52856, 2018.
[10]. R.Abdul hammed, H. Musafer, A. Alessa et al., "Features dimensionality reduction approaches for machine learning based network intrusion detection," Electronics, vol. 8, no. 3, 2019.
[11]. G. C. Fern´andez and S. Xu, "A case study on using deep learning for network intrusion detection," in Proceedings ofthe MILCOM 2019-2019 IEEE Military
Communications
Conference (MILCOM), pp. 1–6, IEEE, Norfolk, VA, USA,2019.
[12]. C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "Enhancing network intrusion detection classifiers using supervised adversarial training," Fe Journal of Supercomputing, vol. 76,no. 9, pp. 6690–6719, 2020.