ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 10(14), DOI: 10.17485/ijst/2017/v10i14/93690, April 2017* ISSN (Online) : 0974-5645

Fast kNN Classifiers for Network Intrusion Detection System

B. Basaveswara Rao* and K. Swathi

Computer Center, Acharya Nagarjuna University, Guntur - 522510, Andhra Pradesh, India;
bbrao@anu.ac.in, kswathi@pvpsiddhartha.ac.in

Abstract

Objective and Background: To adapt two fast kNN classification algorithms i.e., Indexed Partial Distance Search kNearest Neighbor (IKPDS), Partial Distance Search kNearest Neighbor (KPDS) and comparing with traditional kNN classification for Network Intrusion Detection. **Methods/Statistical Analysis:** NSL-KDD data set is used to evaluate the kNN classification, KPDS and IKPDS with 10 fold cross validation test. This experiment results shows that the IKPDS reduces the classification completion time compare with kNN and KPDS by preserving the same classification accuracy as well as the same error rate for different types of attacks. A novelistic method proposed for classifying the unknown patterns whether it is a malicious or legitimate using IKPDS algorithm. **Findings:** These algorithms efficiency were tested with the sample of 12597 instances and verified with actual class label. The resultsshow that 99.6% accuracy of the proposed method. **Applications/Improvements:** A deep analysis can be performed on DoS and Probe attacks as they are exhibiting similar characters andfeature selection techniques may also be implemented inorder to improve the accuracy and reduce the computational time.

Keywords: IKPDS, Intrusion Detection, kNN Classification, NSL-KDD, Partial Distance Search

1. Introduction

As the Internet usage is increasing significantly, security becomes more challenging problem. A network is securedonly when it is provided by a software/hardware protection system with a strong monitoring, analyze and defense mechanisms. A class of these types of systems is named as Network Intrusion Detection Systems (NIDS). It is to monitor the dynamic behavior of intrusion from time to time and implement the defiance mechanisms within in a short span of time. Different types of anomaly detections and comparative study is carried out[1].

A better NIDS is defined by two characteristics i.e., fast detection with less false alarms and fast activation of the defense mechanisms. The objective of NIDS is to detect the any suspicious activities and to identify specific actions to prevent or stop the intrusion. These two objectives should be fulfilled by the NIDS with in short time by analyzing the large network traffic data to determine the anomaly patterns with respect to normal patterns and also implement proper detection mechanisms. There is a necessity to apply machine learning algorithms to the NIDS with less computational time. KNN classification algorithm is one of the efficient algorithms of machine learning techniques even though some drawbacks like it take more computational time.

Data mining techniques, like classification, clustering and association rule miningare alsoused in NIDS[3,5,8,13]. Among all these data mining techniques kNearest Neighbor classification (kNN) algorithm was used prominently in the NIDS even though some drawbacks[11,13,14] due to its better detection rates.

But kNN is a lazy learner and takes much computational time as well ashigh storage cost. To overcome these drawbacks, from last two decades many researchers proposed, developed and implementedfast kNN classification[2,7,13]. These studies are reduced the computational time of the classification without

* *Author for correspondence*

compromising the accuracy of kNN. They applied these algorithms to image classification and pattern recognition. This paper investigates the adoptability aspects of fast KNN classifier for NIDS.

The fast kNN algorithms are adopted to reduce the computational time of the classification inorder to make NIDS more faster[7,10]. This reason has motivated to find the impact of fast KNN algorithms for NIDS, in view of computational time on a benchmark NSL-KDD dataset[9]. An evolutionary analysis is carried out between traditional kNN and fast kNN algorithms with respect to computational time by preserving the same classification accuracy. In the remaining of the paper is organized as follows: in section 2 the related work is briefly discussed. The basics of Fast kNN algorithms are presented in section 3. The section 4 Adaptation procedure of fast kNN algorithms for NIDS is explained. In section 5 proposed models for intrusion detection system is given. Then finally results and conclusion are given in section 6 and 7 respectively.

2. Existing Work

Several authors have studied the kNN classification and other classifications as a machine learning techniques applied to, image processing pattern recognition and NIDS are summarized.

Author in[2] proposed partial distance search algorithms and its implementation on kNN classification in a wavelet domain. This model was implemented on Texture and pattern recognition which reduced the computational time when compared to traditional kNN classification algorithm. Author in[10] extended the work of the author[2]. An approach of indexing the features based on variance vector as a part of preprocessing is introduced on partial distance search in k nearest-neighbor classification algorithm. This experiment was done on Brodatz texture images and observed better results.

Author in[11] suggested that attribute normalization improves the performance of intrusion detection. They implemented the experiment on three methods, kNN, PCA and SVM, that are employed on the normalized data as well as on the original data for comparison of the detection results.

Author in[12] proposed a 2-class classification strategy on NSL-KDD dataset with 10-fold cross validation method to produce the final classification results, with a high detection rate and low false alarm rate in terms of normal or intrusion.

Author in[13] introduced a nonlinear valuation function based on lattice based nearest neighbor classifier to tune the performance of the intrusion detection. This work was evaluated by using KDD Cup'99 Data Set.

Author in[14] developed a classification model named as an Ensemble classifier is a technique which uses a combination of a plurality of classifiers i.e., the use of multiple classification algorithms for obtaining better accuracy. This model consists of two classification algorithms; they are k- nearest neighbors and neural network for the abuse detection. This model was implemented on KDD Cup'99 dataset.

Author in[15] designed and developed three different classifiers based on kNN classifier's concept for facial age estimation to achieve high efficiency. In the first classifier, kNN-distance approach was adopted to calculate minimum distance between different test face images. In the second classifier a modified-KNN version was proposed and the classifier scoring results interpolated to calculate the exact age estimation. Finally, as third classifier kNN-regression classifier was used to combine the classification and regression approaches to improve the accuracy of the age estimation system.

Author in[16] proposed an attributed weight setting method for kNN based classifiers using quadratic programming. This method also suitable for small training sets. The method was tested on various data sets like Breast Cancer data set, Pima Indians Diabetes data set etc.,.

Different authors were worked with different mining algorithms on various application areas. These studies are categorized with reference of various algorithms and datasets presented precisely for ready reference to the researchers in Table 1. The evolution of performance of kNN classification algorithm, computational time for classification was studied by less number of authors whereas efficiency of accuracy studied by more number of authors.

Table 1. Details of the different classifiers and datasets for NIDS

Authors	Algorithms			Factors in Study		Data SetUsed		
	kNN	Partial kNN	Others	Accuracy	Time	KDDCup99	NSL-KDD	Others
[2]		√			√			√
[10]		√			√			√
[11]	√		√	√		√		
[12]			√	√			√	
[13]	√			√				√
[14]	√		√	√		√		
[15]	√			√				√
[16]	√			√	√			√

3. kNN Classification Algorithms

In this section kNN and fast kNN classification algorithmsare explained.

3.1 kNN Classification

kNN classifier is based on a distance function that measures the difference or similarity between two instances. The standard Euclidean distance d(x, y) between two instances x and y is defined in [12] as

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

where, $\mathbf{x_i}$ is the i[th] featured element of the instance $\mathbf{x}$, $\mathbf{y_i}$ is the i[th] featured element of the instance y and $\mathbf{n}$ is the total number of features in the data set.

Assume that the design set for kNN classifier is U. The total number of samples in the design set is S. Let C ={C_1,C_2,…C_L} are the L distinct class labels that are available in S. Let *x* be an input vector for which the class label need to be predicted. Let y_i denotes the i[th] vector in the design set S. kNN algorithm is to find *k* closest vectors in the design set S to the input vector *x*. Then the input vector *x* is classified to class C_j if the majority of the *k* closest vectors have their class as C_j.

3.2 KPDS Algorithm

KPDS is the variations of kNN classifiers that reduce the processing time. The partial distance search approach is introduced in vector quantization encoding methods[2]. KPDS is a partial distance search algorithm that applied on kNN classification[10]. This algorithm reduces the computational time compared with kNN classification. The KPDS algorithm is:

Step 1: Find the first k squared distances D (d_1,d_2, . . d_k) among the first k vectors(y_1, y_2, . . .y_k) in S with the input vector*x* for which class label need to be predicted where, $d_i = d^2(x,y_i)$, i=1,2,3...k.

Step 2: Arrange the k distances in order such that $d_1 \leq d_2 \leq .. \leq d_k$. Set $t = k+1$.

Step 3: Find the distance of $y_t$from *x* as follows:

If $\sum_{p=1}^{n}(x_p - y_{tp})^2 \geq d_k$ where $1 \leq n \leq N$. go to Step 4.

Otherwise set $d_k = d^2(x, y_t)$ and reorder the k distances as Step 2.

Step 4: Increase *t* by 1. If *t* is equal to S terminate. The algorithm, otherwise go to Step3.

3.3 IKPDS Algorithm

The basic idea behind IKPDS approach is to reorder the features in the vector in such a way that the feature which contributes major part in the distance measure to be placed first. i.e., before performing the KPDS, all the features in the vector are indexed based on their contribution in the distance measure[10]. The variances of features of these vectors determine their contributions to the distance to speed up searching the k closest vectors as follow.

Step 1: The mean value vector of feature vectors of design set is M, where the element $\mathbf{M_i} = \frac{1}{S}\left(\sum_{j=1}^{S}\mathbf{y_{ij}}\right)$.

$\mathbf{where\ i = 1, 2, 3, \ldots N}$.

Step 2: Find V, thevariance vector, where the entry

$V_i = \frac{\Sigma_{j=1}^{S}(y_{ij} - M_i)^2}{(S-1)}, \mathbf{i = 1, 2, \ldots, N}$ Store those subscripts into a variance vector whose dimension is N.

Step 3: Rearrange these variance vector in descending order of their variances.

Step 4: Arrange the features of the samples S and the input vector *x* in their feature variance order.

Step 5: Perform the KPDS on S.

4. Adaption of Fast kNN for NIDS

The adaption process consists of two stages. In the first stage data preprocessing is carriedout and in the second stage traditional kNN,KPDS and IKPDS are executed for finding the computational time and classification accuracy. The data preprocessing stage contains transformation and normalization phases for kNN and KPDS where as in IKPDS apart from these two phases there is an additional phase is needed i.e., feature indexing phase. The experimental flow of adaptive process is shown in Figure 1.

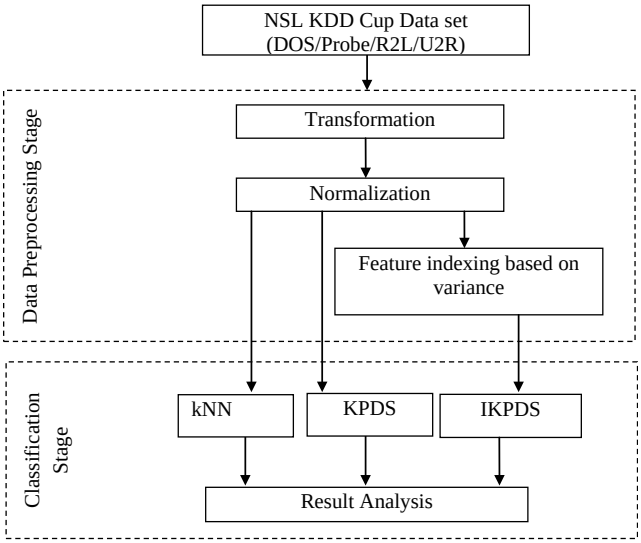

Figure 1. The system overview for adoption.

4.1 Data Preprocessing

As kNN classifier is the distance based classifiers, it requires all its features of the data set to be numerical. Some of the features in KDD data set are of categorical, and should be converted. Out of 41 features 4 features are of categorical and 37 are numerical. All these categorical features must be transformed into continuous features and then normalized.

i) Data Transformation: The features Protocol-type(2), Service(3), Flag(4) Class Label (41) are the categorical features in the data set and are converted into numeric. For example the class label normal is converted into 22 and service type tcp is converted into 3. The following is the example of 2 data samples Figure 2 shows the original NSL-KDD data samples and Figure 3 shows the transformation of the categorical values into nominal values[17].

```
0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.
00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.0
0,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00,0.00,normal.
```

Figure 2. Original samples from NSL-KDD dataset.

```
0,3,19,10,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0,0,0,0,1,0,0,9,9,1,0,0,1
1,0,0,0,0,22
0,3,19,10,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0,0,0,0,1,0,0,19,19,1,0,0.
05,0.0,0.0,0.22
```

Figure 3. Results after data transformation.

ii) Normalization: Feature normalization is an important and necessary activity in the domain of NIDS. By nature NSL-KDD data set features describe various characteristics of the data and the values are qualitative or quantitative with different ranges. These feature values influenced the data analysis or classification process. For example features with higher values can dominate the features with less value. The qualitative features are converted in the transformation step. Now the features are need to be normalized to eliminate such dominance by scaling them all within a specific range. In this paper for normalization process the mean-scale normalization technique is used[11].

$$\mathbf{X_i} = \frac{\mathbf{V_i} - \mathbf{min(V_i)}}{\mathbf{max(V_i)} - \mathbf{min(v_i)}}$$

where, v_i is the actual value of the feature, and the maximum and minimum are taken over all values of the feature. Normally x_i is set to zero if the maximum is equal to the minimum. Figure 4 shows an NSL-KDD Data Sample after applying normalization. The sample set of 20 patterns after completion of transformation and normalization phases are presented in Appendix-I.

```
0.0,0.0,0.47,0.2,0.0,0.0,0.0,0.0,0.0,0.0,0.03,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.31,0.31,1.0,0.0,
0.01, 0.0,0.01,0.01,0.01
```

Figure 4. Result after normalization.

4.2 Classification

This stage involves execution of three kNN classification algorithms over four different data subsets separately. Each data subset contains DOS/Probe/U2R/R2L attack patterns and normal patterns. The feature indexing is based on variance in descending order of 41 features for different types of attacks. The classification is done with a 10 fold cross validation for the k values 3,5 and 10. The results are presented in Appendix-II.

In this experiment the performance metrics classification computational time and classification accuracy are calculated for four types of attacks independently. The results are exhibited in table as well as graph and enclosed in Appendix-II.

5. A Novel Method for Intrusion Detection

This section presents a methodology to predict the class label of an unknown pattern. In a real NIDS scenario the input patterns contains malicious patterns and legitimate patterns. There is a need to identify whether it is alegitimate or malicious requestwithin a short span of time. For this purpose a novelistic method is presented based on IKPDS learning algorithm and given in Figure 5.

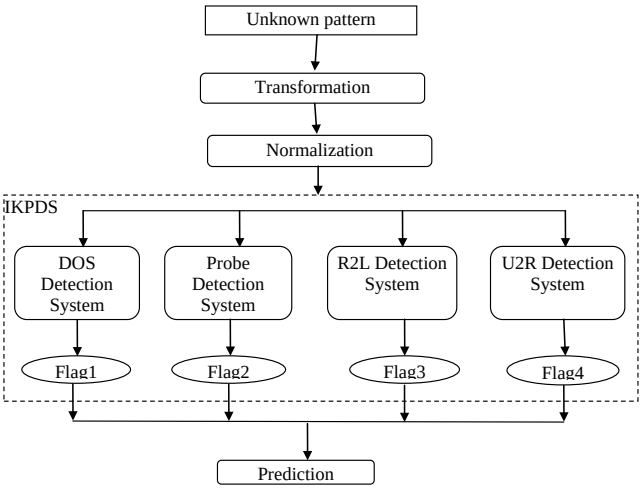

Figure 5. A novelistic IDS model.

An unlabeled/unknown pattern is a tuple of 41 feature set. The aim of these multi-layered IDS is to identify the attacks. The unknown pattern is preprocessed as specified in Figure 1 and the resultant preprocessed tuple is provided as an input for four attack detection systems (DOS, PROBE, R2L, and U2L systems). An Attack Detection System for a specific attack type is one in which the knowledge of related attack is maintained, and performs IKPDS classification on the input data according to its knowledge. For all four types of attacks in NIDS the variance feature indexingin which the preprocessed tuple is further indexed based on the specified knowledge of the attack detection system it has undergone.

The result of the each attack detection system is either 0 when it is a normal record or 1 when it is an attack. After completion of all 4 types of attack detection systems to get 4 binary values which under gone, the flow of the attack detection system is depicted as in Figure 5. The result of the decision system is again 0 or 1. The result becomes 0 if all the input values of the detection system are 0 i.e., (0,0,0,0) i.e., it is the tuple identified as normal. The result of the decision system is 1 if any of the input values of decision systems is 1 i.e., if any or some of the attack detection system identified the tuple as an attack.

6. Results

The adoption of kNN classifications and Novelistic IKPDS learning model are developed in Java 1.6 on Intel core i5 processor, with 4 GB RAM and windows 7 operating systems.The evaluation is done for different values of k (i.e., for 3,5,10 values of k). The computational time for three classification algorithms, their computational time differences are given in Appendix-II. Fast kNN algorithms eliminate the disqualified objects faster, which take less computational time to identify the nearest k objects for a specific object, but the final k nearest object for kNN and fast kNN are the same. So the values in the confusion matrix are same. This leads to the accuracy for traditional kNN and two fast kNN classification algorithms. When the k value is set to 3 the classifier provides the better accuracy. These results are presented in Table 2.

Table 2. Accuracy of kNN, KPDS and IKPDS

	Accuracy		
k Value	**3**	**5**	**10**
U2R	0.9996	0.9995	0.9994
R2L	0.9988	0.9983	0.9976
Probe	0.9981	0.9976	0.9966
DOS	0.9994	0.9991	0.9985

The graphs for computational times are given in Figures 6, 7, 8 and 9 for various attack types of different kNN classifications. It is observed that the accuracies are almost all equal up to third decimal for various types of attacks and for k values 3,5 and 10. The proposed model, explained in section 5 is tested with a random of 12597 samples i.e., 10% from the trained data of NSL-KDD data set. Each sample has given as an input to the model as an unknown pattern and is preprocessed, and provided for all four detection systems. The result of each system is a flag value whose value is either one when it identifies as an attack or zero when the system detects as normal pattern. The 4 flag values of all these 12597 samples are maintained in an array. The decision i.e., whether the sample is attack or not is generated by OR in the 4 flag values. Out of 12597 test samples, there are 50 misclassifications. So the overall accuracy percentage of the model is 99.6%. The flag values and corresponding results are presented in Table 3 for 20 random sample patterns.

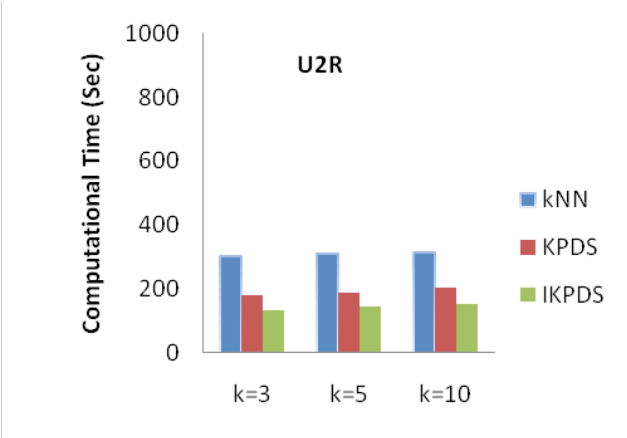

Figure 6. The computational time for U2R.

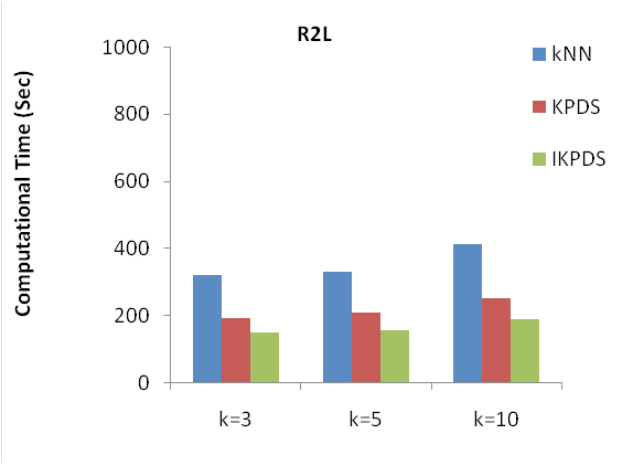

Figure 7. The computational time for R2L.

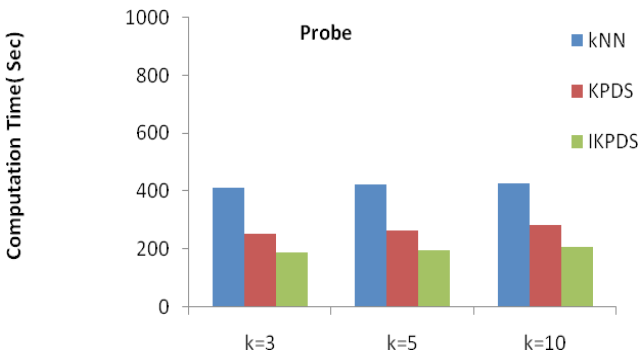

Figure 8. The computational time for probe.

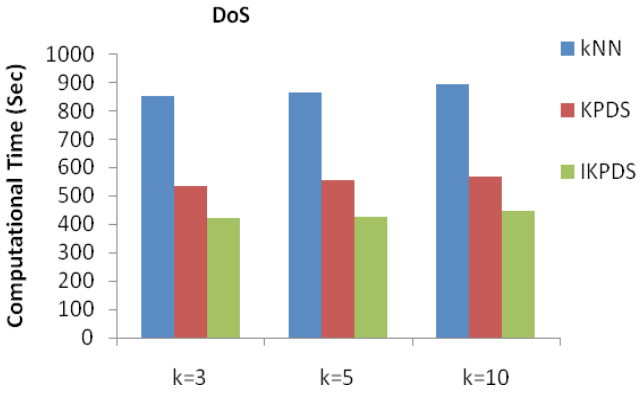

Figure 9. The computational time for DoS.

Table 3. Comparison of predicted class label and actual class label of 20 samples

Flag values				Predic-tion	Actual Label	Correctly Classified (Y/N)
DOS	Probe	R2L	U2R			
0	0	0	1	Attack	Attack	Y
0	0	0	0	Normal	Attack	N
1	0	0	1	Attack	Attack	Y
0	0	0	1	Attack	Attack	Y
0	0	0	0	Normal	Attack	N
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	1	0	Attack	Attack	Y
0	0	0	0	Normal	Normal	Y
0	0	0	0	Normal	Normal	Y
1	0	0	0	Attack	Normal	N
0	0	0	0	Normal	Normal	Y

Out of 20 sample patterns only three patterns are misclassified the third unknown pattern was classified both DOS and U2R attacks so it is a dual type of attack pattern. Tables 4 (a,b,c and d) are confusion matrices for individual models of U2R, R2L, Probe and DoS respectively. Each model identifies whether the given pattern is an attack or a non-attack of the specific model type. Each confusion matrix provides values from actual and predicted patterns of same attack or non-attack for a specific model for 12597 test samples. The confusion matrix of the total 12597 test samples is given in Table 5. From Table 4 (c) it is identified that false positive cases are high (4057) out of these 4057 samples only 7 samples are normal misclassified as attack the remaining 4050 samples are of DoS attack type, i.e., DoS attack samples are identified by probe model as probe attack, which cannot be considered as misclassified because these test samples are also attacks. The same case is applied in Table 4 (d) in which, out of 603 misclassifications only 10 are normal samples treated as DoS attack type, the remain 593 samples are of Probe attack and one sample is of U2R attack.

Table 4(a). Confusion matrix for U2R Model

U2R Model	U2R patterns	Actual	
		Non-U2R patterns	Non-U2R patterns
Predicted	U2R Patterns	3	1
	Non-U2R	2	12592

Table 4(b). Confusion matrix for R2L Model

R2L Model		Actual	
		R2L patterns	Non-R2L patterns
Predicted	R2L Patterns	92	2
	Non-R2L	7	12496

Table 4(c). Confusion matrix for Probe Model

Probe Model		Actual	
		Probe patterns	Non-Probe patterns
Predicted	Probe Patterns	1153	4057
	Non-Probe	13	7374

Table 4(d). Confusion matrix for U2R Model

DOS model	DoS patterns	Actual	
		DoS patterns	Non-DoS patterns
Predicted	DoS Patterns	4585	614
	Non-DoS	8	7390

Table 5. Confusion matrix for total test samples

Attack		Actual	
		Attack	Normal
Predicted	Attack	5833	20
	Normal	30	6714

From the overall sample tuples of 12,597 correctly classified tuples are 12,547 and miss classified tuples are 50. Table 6 shows the confusion matrix of the test model which list the total number of predictions of each detection system.

Table 6. Confusion Matrix for proposed model when applied to sample data set of 12597 tuples

Attack type	Total samples	Predicted				
		U2R	R2L	Probe	DoS	Normal
U2R	5	3	0	0	1	2
R2l	99	0	92	0	0	7
Probe	1166	0	0	1153	603	13
DOS	4593	48	0	4050	4585	8
Normal	6734	1	2	7	10	6714

This tables shows that if the sample size increases the percentage of the correctly classification also increases. For U2R the correctly classified patterns are 60% because the sample size is small (5 samples). Whereas for DoS and normal is more than 99.5%. The overall correctly classified patterns are 99.6%.

7. Summary and Conclusion

Adaption of fast kNN classifications for NIDS is studded and NSL KDD data set is used for comparative analysis. The computational time and classification accuracy are calculated for comparative performance metrics.

The main objective of this paper is to find the fast kNN

classification algorithm for NIDS among the existing fast kNN algorithms. The accuracies are same for the three classifications because the entries in confusion matrix are same. It is further observed that IKPDS execution is a less computational time compared to traditional kNN and PKDS for various attack types and different values of k (i.e., 3, 5 and 10) by preserving the same classification accuracy. The novelistic proposed system gives 99.6% accuracy when the IKDS is used for classification over the sample dataset of 12597 records is randomly collected from NSL KDD data set. The experimental results proved that IKPDS was given better classification results within short span of time for NIDS.

After observing the results of the model, it is identified that the attack types Probe and DoS are similar type of attacks when compared with U2R and R2L. After all these observations, the following points are to be considered for future work.

i) To analyze the genetic characteristics of DOS and Probe attack types for their similarities.

ii) The IKPDS algorithm may be implemented as embedded in attack defense systems.

iii) To implement the wavelet transformation on kNN classification for NIDS and compare with IKPDS.

iv) Feature selection may also implemented in order to improve the accuracy and reduce the computational time.

8. References

1. Axelsson S. Research in intrusion-detection systems: A survey. Technical Report 98-17, Department of Computer Engineering, Chalmers University of Technology. Goteborg, Sweden; 1998 Dec. p. 1–93.
2. Hwang WJ, Wen KW. Fast kNN classification algorithm based on partial distance search. Electron Letter. 1998; 34(21):2062–3. Crossref
3. Lee W, Stolfo SJ, Mok KW. A data mining framework for building intrusion detection models. Proceedings of the 1999 IEEE Symposium in Security and Privacy; 1999. p. 120–32.
4. Lippmann RP, Fried DJ, Graf I, Haines JW, Kendall P, McClung D, Weber D, Webster SE, Wyschogrod D, Cunningham RK, Zissman MA. Evaluating intrusion detection systems: The 1998 DARPA offline intrusion detection evaluation. Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX). 2000 Jan; 2:12–26.
5. Dokas P, Ertoz L, Kumar V, Lazarevic A, Srivastava J, Tan PN. Data mining for network intrusion detection. Proceedings of NSF Workshop on Data Mining; 2002. p. 21–30.
6. Xie Q, Laszlo CA, Ward RK. Vector quantization technique for nonparametric classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1993; 15(12):1326–30. Crossref
7. Jeng-Shyang, PAN, Yu-Long, QIA, Sheng-He SUN. A fast K nearest neighbors classification algorithm. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2004; 87(4):961–3.
8. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. Proceedings of Computational Intelligence for Security and Defense Applications. 2009 Jul; 8:1–6. Crossref
9. Nsl-kdd data set for network-based intrusion detection systems [Internet]. Available from: Crossref
10. Qiao YL, Pan JS, Sun SH. Improved partial distance search for k nearest-neighbor classification. 2004 IEEE International Conference in Multimedia and Expo, ICME'04. 2004 Jun; 2:1275–8.
11. Wang W, Zhang X, Gombault S, Knapskog SJ. Attribute normalization in network intrusion detection. 10th international symposium in Pervasive Systems, Algorithms, and Networks (ISPAN). 2009; 2:448–53.
12. Panda M, Abraham A, Patra MR. A hybrid intelligent approach for network intrusion detection. Proceedings of International Conference on Communication Technology and System Design. 2012; 30:1–9. Crossref
13. Jamshidi Y, Nezamabadi-pour H. A Lattice based nearest neighbor classifier for anomaly intrusion detection. Journal of Advances in Computer Research. 2013 Nov; 4(4):51–60.
14. Chaurasia S, Jain A. Ensemble neural network and k-NN classifiers for intrusion detection. International Journal of Computer Science and Information Technology. 2014; 5:2481–5.
15. Tharwat A, Ghanem AM, Hassanien AE. Three different classifiers for facial age estimation based on k-nearest neighbor. Proceedings of Computer Engineering Conference (ICENCO); 2013 Dec. p. 55–60. Crossref
16. Zhang L, Coenen F, Leng P. Setting attribute weights for k-NN based binary classification via quadratic programming. Intelligent Data Analysis. 2003 Jan; 7(5):427–41.
17. Bhavsar YB, Waghmare KC. Intrusion detection system using data mining technique: Support vector machine. International Journal of Emerging Technology and Advanced Engineering. 2013 Mar; 3(3):581–6.

Appendix-I

S.No	Feature Name	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
1	duration	0	0	0	0	0	0	0	0	0.92	0	1	0	0	0	0	0	0	0	0	0
2	protocol_type	0	1	0	1	0	0	1	0	0	1	0	0	0	1	1	1	1	0	0	0
3	service	0.45	0.92	0.66	0.92	0.6	0.6	0.92	0.95	0.66	0.92	0.66	0.66	0.49	1	1	1	1	0.49	0.25	0.48
4	flag	0.2	0.2	0.4	0.2	1	1	0.2	1	0	0.2	0	1	0.2	0.2	0.2	0.2	0.2	0.9	0.9	0.9
5	src_bytes	0.01	0	0	0	0	0	0	0	0	0	0	0	1	0.02	0.02	0.02	0.02	0	0	0
6	dst_bytes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	land	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	wrong_fragment	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	urgent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	hot	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
11	num_failed_logins	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	logged_in	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
13	lnum_compromised	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
14	lroot_shell	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	lsu_attempted	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	lnum_root	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	lnum_file_creations	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	lnum_shells	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	lnum_access_files	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	lnum_outbound_cmds	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	is_host_login	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	is_guest_login	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	count	0	0	0	0	0.82	1	0	0.53	0	0	0	1	0	1	1	0.19	0.72	0.01	0.51	0.29
24	srv_count	0	0.04	0	0.01	0	0	0.04	0	0	0.07	0	0	0	1	1	0.19	0.72	0.01	0.03	0.03
25	serror_rate	0	0	1	0	0.06	0.08	0	0	0	0	0	0.08	0	0	0	0	0	1	1	1
26	srv_serror_rate	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
27	rerror_rate	0	0	0	0	0.9	0.91	0	1	1	0	1	0.9	0	0	0	0	0	0	0	0
28	srv_rerror_rate	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0
29	same_srv_rate	1	1	1	1	0	0	1	0	1	1	1	0	1	1	1	1	1	1	0.05	0.11
30	diff_srv_rate	0	0	0	0	1	0.95	0	0.53	0	0	0	1	0	0	0	0	0	0	0.07	0.05
31	srv_diff_host_rate	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
32	dst_host_count	0.01	0	0.12	0	1	1	0.01	1	1	0	1	1	0.81	0.62	1	1	1	0.02	1	1
33	dst_host_srv_count	0.15	1	0	0.22	0	0	0.32	0	0	0.33	0	0	0.81	0.25	0.83	0.38	1	0.01	0.05	0.06
34	dst_host_same_srv_rate	1	1	0.03	1	0	0	1	0	0.01	1	0.01	0	1	0.41	0.84	0.38	1	0.67	0.05	0.06
35	dst_host_diff_srv_rate	0	0	0.72	0	1	0.91	0	0.53	0.5	0	0.5	1	0	0.04	0.1	0.02	0	0.33	0.07	0.06
36	dst_host_same_src_port_rate	1	1	0.75	1	0	0	1	1	1	1	1	0	0	0.41	0.84	0.38	1	0.17	0	0
37	dst_host_srv_diff_host_rate	0.36	1	0	0.52	0	0	0.5	0	0	1	0	0	0	0	0	0	0	0	0	0
38	dst_host_serror_rate	0	0	0.75	0	0.08	0.08	0	0	0	0	0	0.13	0	0.01	0	0	0	0.67	1	1
39	dst_host_srv_serror_rate	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
40	dst_host_rerror_rate	0	0	0	0	0.91	0.92	0	1	1	0	1	0.84	0.01	0.4	0	0	0	0	0	0
41	dst_host_srv_rerror_rate	0	0	0	0	1	1	0	1	1	0	1	1	0.01	0	0	0	0	0	0	0
42	Coded Class Labels	17	12	18	18	7	7	18	21	21	12	21	7	1	14	14	14	14	4	4	4

Sample(S1, S2, S3 . . . S20)of 20 preprocessed Patterns

Appendix-II Comparison of Computational Time(in seconds) for Traditional kNN, KPDS and IKPDS for various types of attacks with k=3,5 and 10.

	U2R			R2L			Probe			DoS		
	k=3	k=5	k=10	k=3	k=5	k=10	k=3	k=5	k=10	k=3	k=5	k=10
kNN	300.784	311.087	313.349	307.113	320.864	330.807	412.05	423.857	425.863	848.374	861.017	893.7
KPDS	178.576	187.842	202.721	177.021	193.706	209.237	252.217	264.696	283.832	534.144	554.643	567.633
IKPDS	133.689	143.671	150.49	140.805	148.349	157.24	189.93	198.023	208.808	422.619	425.89	444.765
KPDS - kNN	122.208	123.245	110.628	130.092	127.158	121.57	159.833	159.161	142.031	314.23	306.374	326.067
IKPDS - kNN	167.095	167.416	162.859	166.308	172.515	173.567	222.12	225.834	217.055	425.755	435.127	448.935
IKPDS -KPDS	44.887	44.171	52.231	36.216	45.357	51.997	62.287	66.673	75.024	111.525	128.753	122.868