A Novel Framework for NIDS through Fast kNN Classifier on CICIDS2017 Dataset

K. Vamsi Krishna, K. Swathi, B. Basaveswara Rao

Abstract: This paper investigates the performance of a Fast k-Nearest Neighbor Classifier (FkNN) for Network Intrusion Detection System (NIDS) on Cloud Environment. For this study Variance Index based Partial Distance Search (VIPDS) kNN [7] is adopted as an FkNN classifier. A benchmark dataset CICIDS2017[16] is considered for the evaluation process because it is a 78 featured dataset with most updated cloud related attacks. To achieve this objective a framework is proposed for implementing FkNN and compared with kNN classifier by considering two performance measures Accuracy and computational time. This study explores the gain in the computational time without compromising the Accuracy while using FkNN instead of kNN over a large featured dataset. The conclusions are drawn as per the results obtained from the experiments conducted on CICIDS2017 dataset.

Keywords: Fast kNN Classifier, Network Intrusion Detection System, Variance Indexing, and Cloud.

I. INTRODUCTION

Now a days, a wide number of cloud computing servers across globe are being attacked with Distributed Denial of Service (DDoS) attacks, for financial, grudge and other political reasons. In order to put an end to this is serious cyber threats, this segment has been chosen by several researchers [10]. It is strenuous for the attackers to attack the server physically, so the attackers to choose the malicious activities (like resource consumption, IP addresses spoofing etc.), for unavailability of services to legitimate users and to damage the Cloud Service Providers (CSP) economically.

However, the attackers widely use traditional DDoS attacks against customer chain of cloud networks, named as Economic Denial of Sustainability (EDoS)[20] because the attackers concentrated the financial losses of the CSPs. Many of the CSPs took a leap on pay-as-you-use schemes in order to provide service only to paid customers, obeying Service Level Agreement (SLA), in order to allocate customers required amount of resources. This payment method drastically resulted in slackening of customers count. Researchers and scholars across the globe came up with many significant measures to detect and mitigate DDoS attackers and provide service only to the legitimate customers.

Revised Manuscript Received on January 15, 2020.

K. Vamsi Krishna, Ph. D. scholar, Dept. of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, India.
K. Swathi, Professor, Department of CSE, NRI Institute of Technology, Vijayawada.
B. Basaveswara Rao, Ph.D. scholars Dept. of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, India.

Statistical and Machine Learning (ML) techniques through systematic, simulation and integration methods are implemented by several researchers as key solutions to mitigate these attacks [1] and a framework was proposed using FkNN over a large featured dataset in terms of computational time with out compromising the accuracy[4][8][9].

An ML approach is a highly reliable and feasible platform compared to that of statistical modeling because of ML works on a set of predefined algorithms which iterate in parallel with customer execution predicting whether user is a legitimate or malware [3][5].

Several researchers from the last two decades implemented ML algorithms for NIDS, out of these studies kNN classifier is also widely used. To overcome the lazy learning nature of the kNN classifier, a fast kNN classification approach based on Partial Distance Search (PDS) is implemented[8][9] and the experiments were conducted on Kyoto and NSLKDD datasets were implanted[14][15].

After review of these studies the following observations are identified. (i) Many of these works are not implemented on cloud environment and also not concentrated on new types of attacks. (ii) Most of the studies are adopted with traditional attack types and consists of less number of features like KDDCUP 99 and KYOTO-2006+ datasets [14][15]. (iii) These benchmark datasets are not generated on cloud environment and they are obsolete. In this connection to address this problem, with the adoption of a benchmark dataset CICIDS2017 and to perform experiments using fast kNN classifier adopted by [7][8] for NIDS. The main objectives of this paper are of two fold i) to give the prior knowledge to the defenders about the different types of malicious activities and how to implement new type of mitigation algorithms against to these activities effectively within a short span of time. ii) To investigate the impact of the fast kNN classifier in spite of traditional kNN. The motivation of this study is provide a bird-eye-view on proper prior knowledge on cloud based NIDS using ML approaches with the help of recent bench mark dataset. The rest of the paper is organized as follows:

In the next section the related work is presented. The basics and dataset description are given in section 3. The methodology for conducting experiments also explained in section 4. Finally results and conclusions are mentioned in section 5.
II. RELATED WORK

Several researchers have been working in the field of NIDS techniques through ML approaches from last few decades[2]. In view of the wide usage of cloud, the malicious activities are also increased in the cloud environment so the researchers also developed defense mechanisms from the last decade. This section discusses some of them briefly. Table I presents a brief description of various researchers along with the datasets considered, methodologies adopted/proposed.

![Table I: Related work](image)

From the related work, it is observed that most of the authors proposed various machine learning techniques like kNN, SVM, Random Forest, Deep learning etc. Very few authors concentrated on the reduction of the computational time which is one of the most important parameters of NIDS. All these works were tested on benchmark datasets like KDDCUP 99, NSL-KDD, CIDDS-01 among them most of the datasets are older and might not addressing the latest attacks.

From the above observations the work is carried out to minimize the detection time of the classifier without compromising the detection rate.

III. BASICS AND DATASET DESCRIPTION

This section discusses the basic notations and concepts that are using in the rest of this paper.

A. Basics

The work in this paper is based on the kNN classifier and its variations. The kNN classifier is a distance based algorithm and Euclidian distance is a common distance metric used by most of the researchers. The Euclidian distance formula used in this paper is given below.

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

Where d is the distance metric, x and y are data samples for which distance is to be measured, n is the number of features in data samples.

- **k-Nearest Neighbor Classifier:**
  k-NN classifier is a distance based learner. In this approach no prior model is built before testing due to this, the model is named as lazy learning. Even though the kNN classifier gives best accuracies due to its high computational time for testing limits the most of the researchers to use this approach.

- **Partial Distance Search based kNN:**
  A PDS based kNN classifier is a variation of kNN classifier that minimizes the computation time of the classifier by applying a partial distance search (PDS) as a distance metric instead of Euclidian distance. The PDS algorithm will discards an instance (S_i) from the sample set (S) that is having high distance value when compared with the current k nearest distances at the earliest without completely computing the distance.

The following is the formula for PDS adopted in this methodology.

\[ D(x,S_i) = \sum_{i=1}^{l} (x_i - y_i)^2 \]  

Here D(x,S_i) is a squared distance function, x is the data sample for which need to find the k nearest neighbors from the available sample set S. \( 0 \leq i \leq n \) where n is the total number features. If \( l = n \) then the sample S_i is considered as new nearest to x otherwise if the D(x,S_j) is greater than the maximum k distances computed upto r-1 then the sample S_j is discarded as it not in k nearest neighbors of x in this case the value of l is less than n.

- **Fast kNN classifier (FkNN):**
  The basic idea behind FkNN 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 VIPDS, all the features in the vector are indexed based on their contribution in the distance measure. The variances of features of these vectors determine their contributions to the distance to speed up searching the k closest vectors.

B. DATASET:

CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IP addresses, source and destination ports, protocols and attack (CSV files). Also available is the extracted features definition.
Generating realistic background traffic was top priority in building this dataset. The data capturing period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m. on Friday July 7, 2017, for a total of 5 days. Monday is the normal day and only includes the benign traffic. Different attack types - FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. They have been executed both morning and afternoon on Tuesday, Wednesday, Thursday and Friday[16].

In the current study only Friday morning period of data is considered. This data set consists of 1,91,033 instances and 78 features including decision attribute having two class labels namely benign and Bot. A detailed description of the features in the dataset is skipped in this paper because it is available in [6].

C. PERFORMANCE METRICS:

Accuracy, Precision and Recall are considered as the performance measures for identifying the impact of classifiers efficiency. A confusion matrix is calculated with the entries of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values.

Where TP is the total number of correctly predicted normal samples, TN is the total number of correctly predicted attack samples, FP and FN the number of normal samples predicted as attacks and number of attack samples predicted as normal respectively.

Accuracy: Accuracy is considered as the ratio of total number of testing samples correctly classified out of the total number of samples.

\[
\text{ACCURACY} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)
\]

Accuracy can only be considered when all the samples equally distributed as per the class labels. But when the data distribution is skewed, i.e., not distributed equally for positive and negative samples then precision and recall were also considered.

Precision: Precision is the ratio of actual positive samples out of total samples that are predicted as positive samples.

\[
\text{Precision} = \frac{TP}{TP+FP} \quad (4)
\]

Recall: Recall is also known as sensitivity is the ratio of total number of correctly classified positive samples over the total number of actual positive samples. This will helpful when false negative values are high.

\[
\text{Recall} = \frac{TP}{TP+FN} \quad (5)
\]

IV. METHODOLOGY

The methodology of the proposed frame work comprised into three phases, they are i) data preprocessing, ii) variance based feature indexing iii) classification. Figure 1 depicts the proposed methodology. In the first phase only normalization is carried out because all the features in the dataset are numeric and there is no need of transformation.

Normalization: Feature normalization is an important and necessary step in the domain of NIDS. By nature CICIDS17 data set features describe various characteristics of the data and the values are 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. 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 min-max normalization technique is used. The formula for the normalization is as follows:

\[
x' = \left(\frac{x - \text{min}_X}{\text{max}_X - \text{min}_X}\right) \times (\text{new}_\text{max}_X - \text{new}_\text{min}_X) + \text{new}_\text{min}_X \quad (6)
\]

Here X is the feature vector to which normalization is to be applied. x is a value in X and x ‘ is the normalized value of x. minX and maxX are minimum and maximum values of the feature X. new minX and new maxX are new scale into which the X is be to normalized with a scale of (0,1). The CICIDS 2017 dataset is converted into normalized CICIDS 2017N dataset. An example of normalization for three records are given in Appendix -I.

In the second phase the preprocessed dataset is reordered based on feature ranks, the ranks are computed according to the descending order of each feature variance value. The variance formula is given below.

\[
v(X) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (7)
\]

Where v is the variance function, X is a feature vector for which variance needs to be calculated and xi is the i th instance value and \( X \) is the mean of the feature vector X. Appendix –II provides list of feature number, feature names and their variance index ranks in descending order of their variance value. The CICIDS2017N dataset is reordered into CICIDS2017V.

In the classification phase three classification techniques discussed in section 3 were implemented. KNN, PDS-kNN were applied on the CICIDS2017N dataset and FkNN is applied on the CICIDS2017V dataset with a 10 fold cross validation.

In this model the learning ability of three classifiers identified with the implementation of different values of k (i.e., 3, 5 and 7) because the number of features as well instances in the dataset are high and it will consume more computational time if the value of k is large. The following figure exhibits the flow of the methodology.
A Novel Framework for NIDS through Fast kNN Classifier on CICIDS2017 Dataset

V. RESULTS AND DISCUSSIONS

The methodology is implemented in Intel core i3 processor with Windows 10 operating system, with the JDK1.8 as java compiler. After implementing three classifiers kNN, PDS-kNN and FkNN for different values of k (3,5 and 7) with 10 fold cross validation the following results obtained regarding to the Accuracy, Precision, Recall and computational time. The Accuracy, Precision and Recall are calculated form the confusion matrices. The computational time is measured based on execution time of the classifier. In the following subsections the significance of the performance metrics are explained.

A. Accuracy, Precision and Recall:

The following confusion matrix is obtained after execution of three classifiers.

Table II: Confusion matrix for kNN, PDS-kNN and Fast kNN classifiers with different K values (3, 5 and 7)

| Classifier Name | K Value | TP   | TN   | FP   | FN   |
|-----------------|---------|------|------|------|------|
| FkNN            | 7       | 188749| 1762 | 206  | 194  |
| PDS-kNN         | 7       | 188749| 1762 | 206  | 194  |
| kNN             | 7       | 188749| 1762 | 206  | 194  |
| FkNN            | 5       | 188759| 1789 | 196  | 167  |
| PDS-kNN         | 5       | 188759| 1789 | 196  | 167  |
| kNN             | 5       | 188759| 1789 | 196  | 167  |
| FkNN            | 3       | 188786| 1826 | 169  | 130  |
| PDS-kNN         | 3       | 188786| 1826 | 169  | 130  |
| kNN             | 3       | 188786| 1826 | 169  | 130  |

By observing the confusion matrix

i) All the entries are equal irrespective of type of classification. So there is no impact on detection rate by using PDS-kNN, FkNN instead of kNN.

ii) There is no significant difference between the various values of k.

The Accuracy, Precision and Recall are calculated as per the formulas defined in section III.C from the above confusion matrix.

Table III presents Accuracy, Precision and Recall measures of the classifiers at different values of k.

Table III: Accuracy, Precision and Recall measures for kNN classifier at different k values (3, 5 and 7) of different classifiers.

| Classifier & K Value | Accuracy | Precision | Recall |
|----------------------|----------|-----------|--------|
| 3                    | 99.8434  | 99.9106   | 99.9312|
| 5                    | 99.8099  | 99.8963   | 99.9116|
| 7                    | 99.7905  | 99.8090   | 99.8973|

The Accuracy, Precision and Recall values of all three classifiers are equal because the confusion matrix entries are equal. The variation in the k value is not much influencing on these three metrics.
For k=3 the gain in computational time among kNN and FkNN is 88.52% which is observed as a highest value. Among the kNN and FkNN the gain in computational is stabilized at 88% for different values of k.

VI. CONCLUSION

In this paper, a FkNN classifier is introduced as an intrusion detection system for cloud environment on CICIDS2017 benchmark dataset. The FkNN is compared with kNN and PDS-kNN in terms of Accuracy, Precision, Recall and computational time. From the experimental results it is concluded that the FkNN classifier is a better classifier with less detection time and without loss in Accuracy. The gain in computational time is more than 88% as compared to traditional kNN. The authors suggested FkNN as a better machine learning algorithm for NIDS with latest attack types and with high dimensional dataset for cloud environment to mitigate attacks and reduce the economic losses of CSPs with less span of time. As a future scope of this study it is to implement on a real-time environment and can be compared with other classifiers also.

Appendix – I

List of three original and three normalized samples from CICIDS 2017 dataset

| Feature Name | Variance Value |
|--------------|----------------|
| ACK Flag Count | 0.1930032 |
| PHF Flag Count | 0.1753727 |
| URG Flag Count | 0.089501 |

Appendix – II

List of features in variance index ranking in descending order from CICIDS 2017 dataset along with #feature, feature name and corresponding ranks

| Feature Name | Variance Value |
|--------------|----------------|
| init.Win_bytes_forward | 0.0441928 |
| init.Win_bytes_backward | 0.01987275 |
| Flow IAT | 0.1507072 |
| Fwd IAT | 0.0149585 |
| Idle Max | 0.01409249 |
| Idle Mean | 0.01339179 |
| Idle Min | 0.01350581 |

Retrieval Number: E6580018520/2020©BIEESP
DOI:10.35940/ijrte.E6580.018520
A Novel Framework for NIDS through Fast kNN Classifier on CICIDS2017 Dataset

| S.No | Feature# | Feature Name             | Variance value |
|------|----------|--------------------------|----------------|
| 17   | 69       | min_seg_size_forward     | 0.01298375     |
| 18   | 29       | Bwd IAT Max              | 0.01283467     |
| 19   | 43       | FBN Flag Count           | 0.01270941     |
| 20   | 14       | Bwd Packet Length Std    | 0.00965567     |
| 21   | 27       | Bwd IAT Mean             | 0.00749812     |
| 22   | 22       | Fwd IAT Mean             | 0.00739829     |
| 23   | 30       | Bwd IAT Min              | 0.00716403     |
| 24   | 25       | Fwd IAT Min              | 0.00706301     |
| 25   | 18       | Flow IAT Std             | 0.0056745      |
| 26   | 54       | AvgBwd Segment Size      | 0.00529375     |
| 27   | 13       | Bwd Packet Length Mean   | 0.00529375     |

| S.No | Feature# | Feature Name             | Variance value |
|------|----------|--------------------------|----------------|
| 28   | 52       | Average Packet Size      | 0.00518673     |
| 29   | 40       | Packet Length Mean       | 0.00507787     |
| 30   | 36       | Fwd Packets/s            | 0.00489103     |
| 31   | 51       | Down/Up Ratio            | 0.00410754     |
| 32   | 11       | Bwd Packet Length Max    | 0.0037216      |
| 33   | 41       | Packet Length Std        | 0.00261116     |
| 34   | 23       | Fwd IAT Std              | 0.00248545     |
| 35   | 12       | Bwd Packet Length Min    | 0.0024418      |
| 36   | 28       | Bwd IAT Std              | 0.00190441     |
| 37   | 16       | Flow Packets/s           | 0.00186731     |
| 38   | 39       | Max Packet Length        | 0.00135974     |
| 39   | 17       | Flow IAT Mean            | 0.00126041     |
| 40   | 75       | Idle Std                 | 0.00080115     |
| 41   | 10       | Fwd Packet Length Std    | 0.00051836     |
| 42   | 7        | Fwd Packet Length Max    | 0.00049936     |
| 43   | 38       | Min Packet Length        | 0.00044208     |
| 44   | 20       | Flow IAT Min             | 0.00039501     |
| 45   | 9        | Fwd Packet Length Mean   | 0.00038827     |
| 46   | 53       | AveFwd Segment Size      | 0.00038827     |
| 47   | 8        | Fwd Packet Length Min    | 0.00032509     |
| 48   | 37       | Bwd Packets/s            | 0.00028271     |
| 49   | 45       | RST Flag Count           | 0.00025136     |
| 50   | 50       | ECE Flag Count           | 0.00025136     |
| 51   | 74       | Active Max               | 0.00015841     |
| 52   | 42       | Packet Length Variance   | 0.00015721     |
| 53   | 15       | Flow Bytes/s             | 0.00014007     |
| 54   | 71       | Active Std               | 9.57E-05       |

| S.No | Feature# | Feature Name             | Variance value |
|------|----------|--------------------------|----------------|
| 55   | 70       | Active Mean              | 6.71E-05       |
| 56   | 73       | Active Min               | 5.18E-05       |
| 57   | 5        | Total Length of Fwd Packets | 4.12E-05    |
| 58   | 63       | SubflowFwd Bytes         | 4.12E-05       |
| 59   | 34       | Fwd Header Length        | 2.81E-05       |
| 60   | 65       | SubflowBwd Bytes         | 2.80E-05       |
| 61   | 6        | Total Length of Bwd Packets   | 2.80E-05  |
| 62   | 62       | SubflowFwd Packets       | 2.79E-05       |
| 63   | 3        | Total Fwd Packets        | 2.79E-05       |
| 64   | 68       | act_data_plt_fwd         | 2.77E-05       |
| 65   | 35       | Bwd Header Length        | 2.73E-05       |
| 66   | 4        | Total Backward Packets   | 2.71E-05       |
| 67   | 64       | SubflowBwd Packets       | 2.71E-05       |
| 68   | 56       | FwdAvg Bytes/Bulk        | 0             |
| 69   | 61       | BwdAvg Bulk Rate         | 0             |
| 70   | 33       | Fwd URG Flags            | 0             |
REFERENCES

1. Othman, Suad Mohammed, et al. "Intrusion detection model using machine learning algorithm on Big Data environment." Journal of Big Data 5.1 (2018): 34.

2. Salo, Fadi, Ali Bou Nassif, and Aleksander Essex. "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection." Computer Networks 148 (2019): 164-175.

3. Wagh, SharmilaKishor, Vinod K. Pachghare, and Satish R. Kolhe. "Survey on intrusion detection system using machine learning techniques." International Journal of Computer Applications 78.16 (2013).

4. Chiba, Z., Abghour, N., Moussaid, K., El Omri, A., &Rida, M. (2019, June). An Efficient Network IDS for Cloud Environments Based on a Combination of Deep Learning and an Optimized Self-adaptive Heuristic Search Algorithm. In International Conference on Networked Systems (pp. 233-249). Springer, Cham.

5. Idhammad, Mohamed, Karim Afdel, and Mustapha Belouch. "Distributed intrusion detection system for cloud environments based on data mining techniques." Procedia Computer Science 127 (2018): 35-41.

6. Abdulhammed, Razan, et al. "Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection." Electronics 8.3 (2019): 322.

7. Basaveswara Rao B &Swathi K (2016) Variance-Index Based Feature Selection Algorithm for Network Intrusion Detection, IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-8772, Volume 18, Issue 4, Ver. V (Jul.-Aug. 2016), PP 01-11.

8. Basaveswara Rao B &Swathi, K. (2017), Fast kNN Classifiers for Network Intrusion Detection System. Indian Journal of Science and Technology, 10(14).

9. Swathi, Kailasam, and BobbaBasaveswara Rao. "Impact of PDS Based kNN Classifiers on Kyoto Dataset." International Journal of Rough Sets and Data Analysis (IJRSDA) 6.2 (2019): 61-72.

10. Ali, Mohammed Hasan, and Mohamad FadliZolklipli. "Intrusion-Detection System Based on Fast Learning Network in Cloud Computing." Advanced Science Letters 24.10 (2018): 7360-7363.

11. Ahmed, H. A. S., Ali, M. H., Kadhium, L. M., Zolklipli, M. F., &Alsariera, Y. A. (2017). A review of challenges and security risks of cloud computing. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9(1-2), 87-91.

12. Ali, Mohammed Hasan, et al. "A hybrid Particle swarm optimization-Extreme Learning Machine approach for Intrusion Detection System." 2018 IEEE Student Conference on Research and Development (SCoReD). IEEE, 2018.

13. Umer, Muhammad Fahad, Muhammad Sher, and Yaxin Bi. "Flow-based intrusion detection: Techniques and challenges." Computers & Security 70 (2017): 238-254.

14. "Nsl-kdd data set for network-based intrusion detection systems." Available on: http://nsl.cs.unb.ca/KDD/NSLKDD.html, March 2009.

15. Kyoto 2006+ dataset is available on : http://www.takakura.com/Kyoto_data/.

16. Basaveswara Rao B, et al., “A Fast KNN Based Intrusion Detection System for Cloud Environment”, Journal of Adv Research in Dynamical & Control Systems, Vol. 10, Issue 7, 2018 PP 1509–1515.

17. Ring, Markus, Sarah Wunderlich, DenizScheuring, Dieter Landes, and Andreas Ritho. "A Survey of Network-based Intrusion Detection Data Sets." Computers & Security (2019).

18. Ahumam, A., Maglaras, L., Ferrag, M. A., Dordour, M., &Janicke, H. (2019, May). A novel hierarchical intrusion detection system based on decision tree and rules-based models. In 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS) (pp. 228-233). IEEE.

19. Bulla, Suneetha, et al. "An experimental evaluation of the impact of the EDoS attacks against cloud computing services using AWS." International Journal of Engineering & Technology 7.1.5 (2018): 202-208.