

AN EFFECTIVE INTRUSION DETECTION FRAMEWORK BASED ON SUPPORT VECTOR MACHINE USING NSL - KDD DATASET

Jamal Hussain

Professor, Dept. of Mathematics and Computer Science, Mizoram University, Aizawl- 796004, India
jamal.mzu@gmail.com

Aishwarya Mishra

Research Scholar, Dept. of Mathematics and Computer Science, Mizoram University, Aizawl
aishwaryamishra1987@gmail.com

Abstract. Intrusion Detection System (IDS) has become necessary for the security and privacy of a system and it takes a major role in network security because of its detection capacity to various types of attacks in the network domain. Recently, Support Vector Machines (SVM) has been applied to provide useful solutions for intrusion detection systems. With its many variants for classification, SVM is a state-of-the-art machine learning algorithm and its performance depends on selection of the appropriate parameters. In this paper, we propose a model based on linear and nonlinear kernel SVMs using NSL-KDD dataset. The parameters for SVM are described in the tabular manner. Then by using the NSL-KDD dataset, our model gives the best result *i.e.*, 100% for accuracy (Both Quadratic and Cubic SVMs).

Keywords: Intrusion Detection System (IDS), Linear and Nonlinear Support Vector Machines (SVMs), Performance Matrices

1. Introduction

Due to vast volume of information on network domain, the information contents engross varieties of attacks leading thereby, increase in intrusions. Intrusion detection is extremely necessary to stop the intruders to interrupt into or misuse our systems [Campos, “*et al.*” (2005); Zainal “*et al.*” (2006)]. The increasing range of threats against and vulnerabilities of a diverse set of targets, such as military, government and commercial network systems, require increasing situational awareness and various cyber security measures [Kabiri and Ghorbani (2005); Depren “*et al.*” (2005), Inayat “*et al.*” (2016)]. Intrusion detection system is a type of security management system for computers and networks. An Intrusion Detection system gathers and analyzes information from various areas. The intrusion detection systems are critical components in the network security to identify possible security breaches within a computer or a network. Data mining techniques [Dokas “*et al.*” (2002); Lee “*et al.*” (1999)] are used to explore and analyze large datasets and find useful patterns. Classification [Benwal and Arora (2012)] is the category that consists of identification of class labels of records that are typically described by set of features in dataset. An Intrusion Detection System (IDS) is a software application or device that monitors the system or activities of network for policy violations or malicious activities and generates reports to the management system. Here, the NSL-KDD dataset [Jabez and Muthukumar (2015)] is used for the experiments and two types of class are present out of which, one is normal and another one is anomaly (<http://www.unb.ca/cic/research/datasets/nsf.html>). All the experiments are done using MATLAB.

There are good numbers of research papers conducted using the NSL-KDD dataset for developing models for IDS and varieties of methods also exist. Some of methods are summarised here. The authors have used the k-means data mining algorithm to detect normal and attack data in order to reduce the false negative rate. The Weka tool is chosen for simulation and Random Tree is selected as classifier to know the data as attack or normal. A new approach is used *i.e.* Outlier Detection approach to detect the intrusion which consists of big datasets with distributed environment that improves the performance of IDS. Also, this method is tested with the KDD dataset. The authors have used least square support vector machine (OA-LS-SVM) method to tackle various types of attacks in ID using KDD 99 dataset. Two levels IDS is used to allow the system to analyze network traffic and the detection levels are coarse-grained IDS and fine-grained IDS. VFDT has proved its efficiency in both generalization tree and new attacks detection. The authors are focused on development of online IDS using modified Q-learning algorithm and RST having 98% accuracy. The Random forest classifier is outperformed among all the classifiers using performance Matrices (precision, recall, F1-Score and accuracy) *i.e.* accuracy of 99%. Both the feature selection and classification methods are used for anomaly detection. The

authors propose the time varying chaos particle swarm optimization method to provide a new machine learning intrusion detection methodology based on two conventional classifiers; multiple criteria linear programming (MCLP) and support vector machine (SVM) [Duquea and Omar (2015); Aggarwala and Sharma (2015); Jabez and Muthukumar (2015) Kabir “*et al.*”(2017)].

2. Proposed Framework

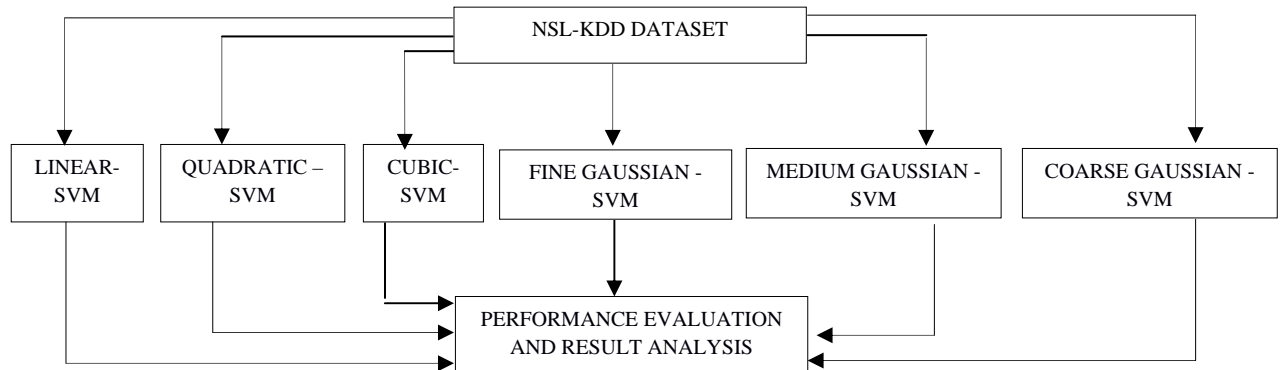


Fig. 1: The proposed experimental framework

Here as depicted above in Figure-1, NSL-KDD dataset is taken as input to the classification methods followed by performance evaluation. Six types of classification methods are used here [Wu “*et al.*” (2008); Joachims (2006); Folino and Sabatino (2016)] and result analysis are done. The experiments are done by using Matlab [Bryant and Garbar (1999); Sumathi and Paneerselvan (2010)]. The 5 fold cross-validation method is used, where 4 fold are kept for training purpose and 1 fold is taken for testing purpose [Kohavi (1995)]. Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection and the advantages of support vector machines are, effective in high dimensional spaces, still effective in cases where number of dimensions is greater than the number of samples, uses a subset of training points in the decision function (called support vectors) and it is also memory efficient, and versatile as different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

3. Dataset Description

The dataset descriptions have been shown below in figures. Figure 2, 3, 4 and 4A shows the scattering of NSL-KDD dataset in the plane using its variables. Figure 5, 6, 7, 8, 9, 10 are used to show the parallel coordinates plot of validation set of linear and kernel SVMs.

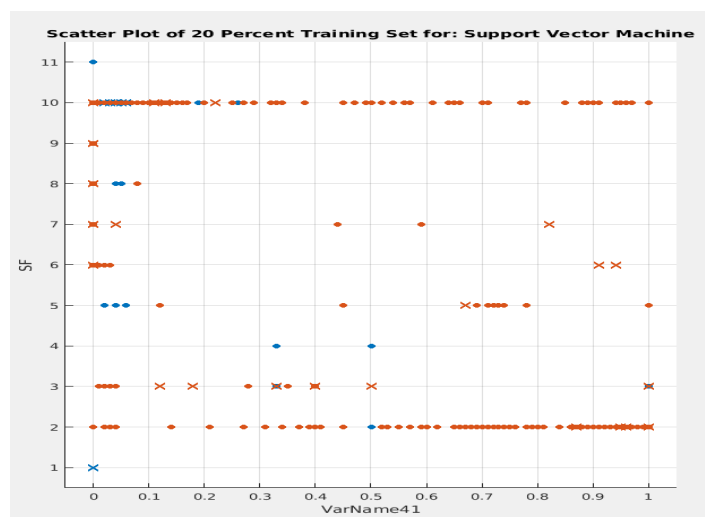


Fig. 2: Scatter plots of the different variables of NSL-KDD dataset

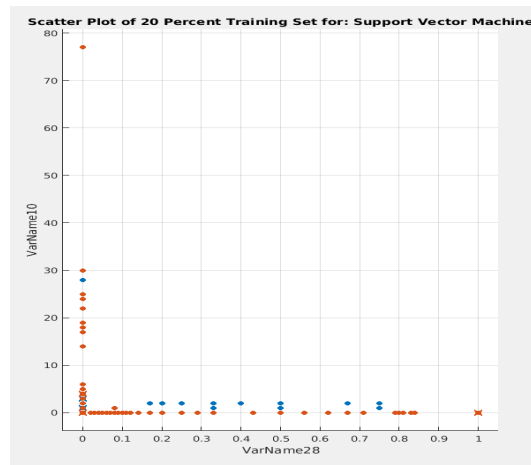


Fig. 3: Scatter plots of the different variables of NSL-KDD dataset

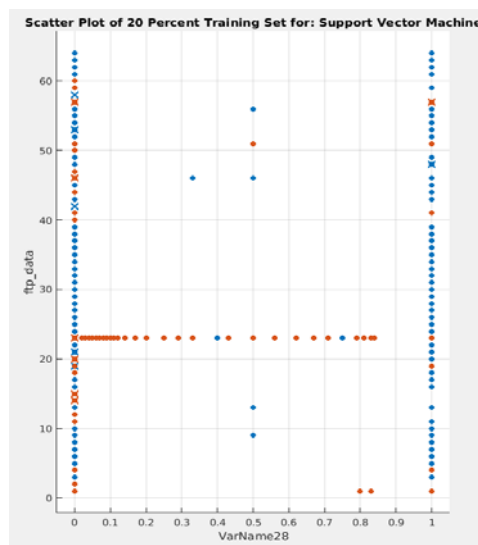


Fig. 4: Scatter plots of the different variables of NSL-KDD dataset

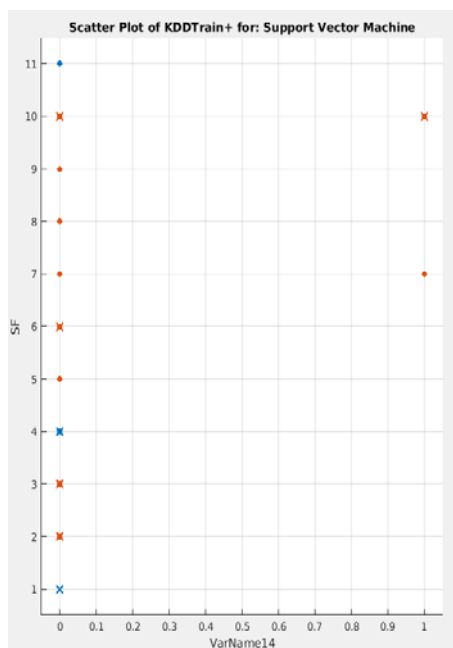


Fig. 4-A: Scatter plots of the different variables of NSL-KDD dataset

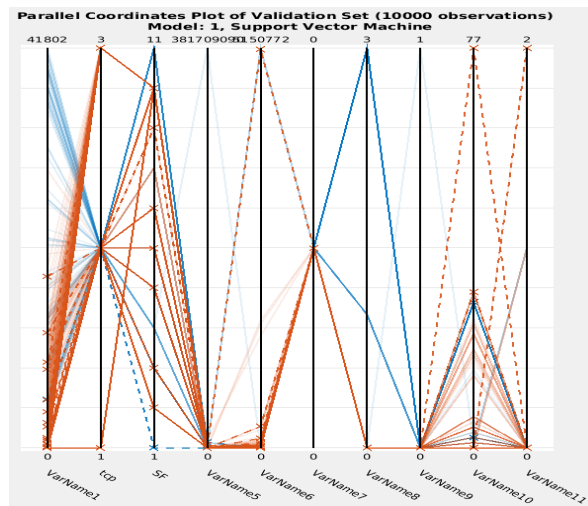


Fig. 5: Parallel coordinates plot of validation set model-1 Support Vector Machines

(Vertical axes arranged from left to right along the x-axis visualize the data along with their corresponding attributes for model-1 with the highest value at the top (100%) and the lowest at the bottom (0%))

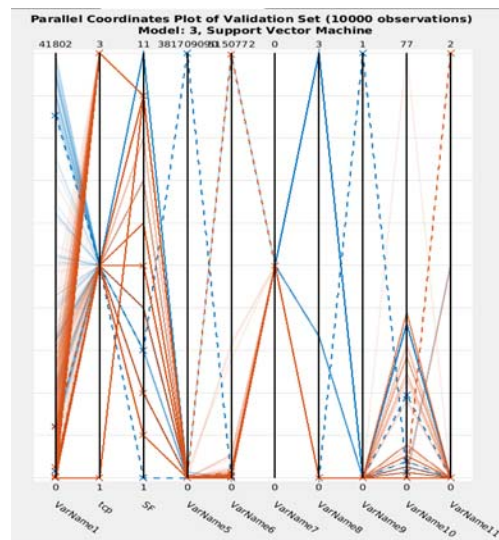


Fig. 6: Parallel coordinates plot of validation set model- 2 Support Vector Machines

Vertical axes arranged from left to right along the x-axis visualize the data along with their corresponding attributes for model-2 with the highest value at the top (100%) and the lowest at the bottom (0%)

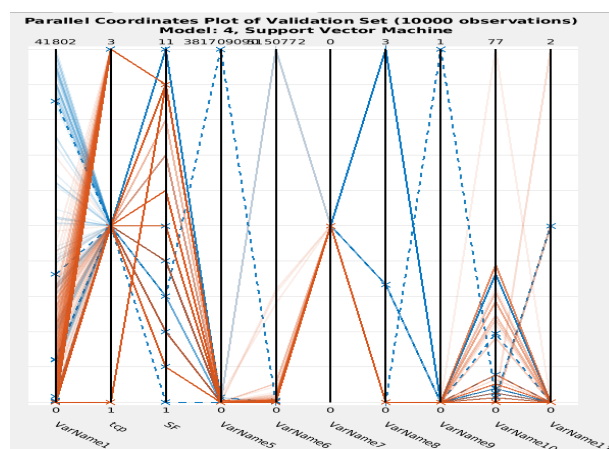


Fig. 7: Parallel coordinates plot of validation set model- 3 Support Vector Machines

(Vertical axes arranged from left to right along the x-axis visualize the data along with their corresponding attributes for model-3 with the highest value at the top (100%) and the lowest at the bottom (0%))

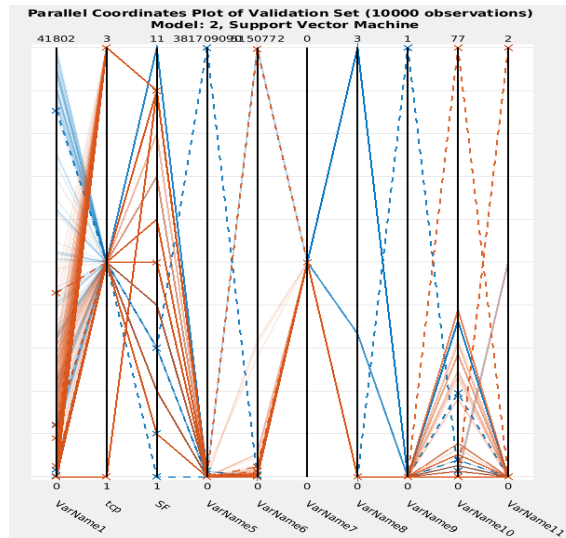


Fig. 8: Parallel coordinates plot of validation set model- 4, Support Vector Machines

(Vertical axes arranged from left to right along the x-axis visualize the data along with their corresponding attributes for model-4 with the highest value at the top (100%) and the lowest at the bottom (0%))

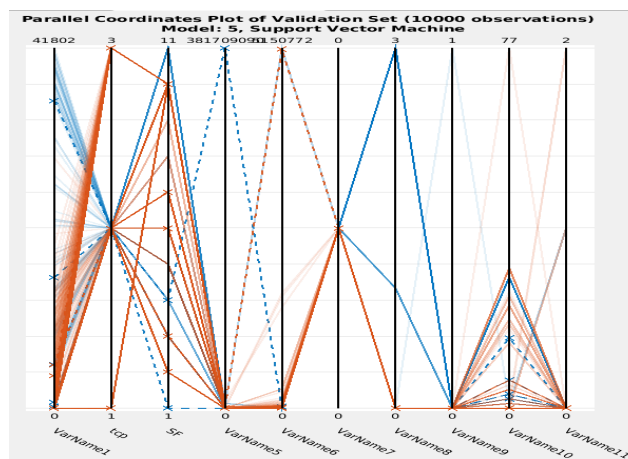


Fig. 9: Parallel coordinates plot of validation set model- 5 Support Vector Machines

(Vertical axes arranged from left to right along the x-axis visualize the data along with their corresponding attributes for model-5 with the highest value at the top (100%) and the lowest at the bottom (0%))

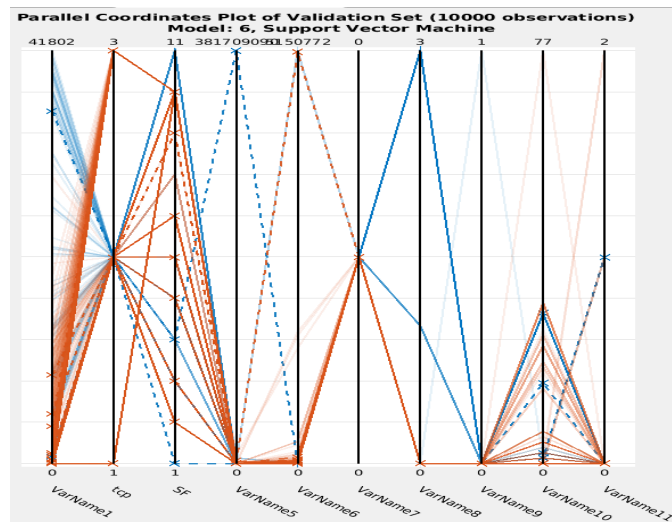


Fig. 10: Parallel coordinates plot of validation set model- 6 Support Vector Machines







(Vertical axes arranged from left to right along the x-axis visualize the data along with their corresponding attributes for model-6 with the highest value at the top (100%) and the lowest at the bottom (0%))

Here, we used 10 fold cross validation technique to validate the dataset for different SVM classifiers. Validation phase is used to estimate how well the model has been trained (dependent upon the size of the data, input etc), which means these classifiers is shown the best results for NSL-KDD dataset.

4. Experimental Results

The results, after plotting of data for both binary and multiclass and their corresponding memory usage, interpretability and model flexibility for different classifiers are shown below in Table-1.

Table 1: Information description of different classifiers

Classifier Type	Prediction Speed	Memory Usage	Interpretability	Model Flexibility
Linear SVM 	Binary: Fast Multiclass: Medium	Medium	Easy	Low Makes a simple linear separation between classes.
Quadratic SVM 	Binary: Fast Multiclass: Slow	Binary: Medium Multiclass: Large	Hard	Medium
Cubic SVM 	Binary: Fast Multiclass: Slow	Binary: Medium Multiclass: Large	Hard	Medium
Fine Gaussian SVM 	Binary: Fast Multiclass: Slow	Binary: Medium Multiclass: Large	Hard	High-decreases with kernel scale setting. Makes finely detailed distinctions between classes, with kernel scale set to $\sqrt{P}/4$.
Medium Gaussian SVM 	Binary: Fast Multiclass: Slow	Binary: Medium Multiclass: Large	Hard	Medium Medium distinctions, with kernel scale set to \sqrt{P} .
Coarse Gaussian SVM 	Binary: Fast Multiclass: Slow	Binary: Medium Multiclass: Large	Hard	Low Makes coarse distinctions between classes, with kernel scale set to $\sqrt{P} \times 4$, where P is the number of predictors.

4.1 Description of Parameters after Training

Different classifiers such as, Linear SVM, Quadratic SVM, Cubic SVM, Fine Gaussian SVM, Medium Gaussian SVM, and Coarse Gaussian SVM have been used for classification of NSL-KDD dataset. Table- 2 shows the parameters of different classifiers after train the dataset [Joachims (2006)].

Table- 2: Description of Parameters after Training

A	Linear SVM:	Training time: 00:02:57 Classifier options: type = SVM, kernel function: linear, manual kernel scale = 1.0, kernel scale mode = auto, box constraint level = 1.0, multi class method = One-vs-One, standardize data = true Feature selection options: feature count before selection = 41, feature excluded = 0, feature included = 41
B	Quadratic SVM:	Training time: 00:01:39 Classifier options: type = SVM, kernel function: Quadratic, manual kernel scale = 1.0, kernel scale mode = auto, box constraint level = 1.0, multi class method = One-vs-One, standardize data = true Feature selection options: feature count before selection = 41, feature excluded = 0, feature included = 41
C	Cubic SVM:	Training time: 00:13:33 Classifier options: type = SVM, kernel function: cubic, manual kernel scale = 1.0, kernel scale mode = auto, box constraint level = 1.0, multi class method = One-vs-One, standardize data = true Feature selection options: feature count before selection = 41, feature excluded = 0, feature included = 41
D	Fine Gaussian SVM:	Training time: 00:01:09 Classifier options: type = SVM, kernel function: Gaussian, manual kernel scale = 1.6, kernel scale mode = manual, box constraint level = 1.0, multi class method = One-vs-One, standardize data = true Feature selection options: feature count before selection = 41, feature excluded = 0, feature included = 41
E	Medium Gaussian SVM:	Training time: 00:00:39 Classifier options: type = SVM, kernel function: Gaussian, manual kernel scale = 6.4, kernel scale mode = manual, box constraint level = 1.0, multi class method = One-vs-One, standardize data = true Feature selection options: feature count before selection = 41, feature excluded = 0, feature included = 41
F	Coarse Gaussian SVM:	Training time: 00:00:31 Classifier options: type = SVM, kernel function: Gaussian, manual kernel scale = 26.0, kernel scale mode = auto, box constraint level = 1.0, multi class method = One-vs-One, standardize data = true Feature selection options: feature count before selection = 41, feature excluded = 0, feature included = 41

5. Performance Evaluation

The contingency table is very useful and it is also known as confusion matrix. It gives the information about predicted and actual values. Here (TP) true positives is the number of positive examples that are correctly classified, (TN) true negatives is the number of negative examples that are correctly classified, (FN) false negatives is the number of positive examples that are incorrectly classified as negative and (FP) false positives is the number of negative examples that are incorrectly classified as positive. There are different types of performance matrices [Caruana and Niculescu-Mizil (2004)] are used in this paper that is (1) Overall Accuracy, (2) Specificity, (3) Sensitivity, (4) G-Mean and ROC curve (AUC) [Hans and Kamber (2006); Zhu “*et al.*” (2010)]. The specificity is the proportion of the TN and (TN+FP) and with the higher specificity fewer positive cases are labelled as negatives, so this ratio can be regarded as the percentage of negative cases correctly classified as belonging to the negative class. The proportion of cases that are TP for all the cases that are positive in diagnostic test (TP+FN) is called sensitivity. The Geometric Mean (G-Mean) is another metric used to evaluate the performance results by using both specificity and sensitivity. It ranges from 0 to 1 and an attribute that is perfectly correlated to the class provides a value of 1. The Receiver Operating Characteristic curve (ROC) and its Area under the Curve (AUC) are used for the performance analysis of the classifier. The ROC graph is the trade-off between benefits and costs. Here, 1 is the best possible value. The Quadratic and Cubic SVMs are shown preminent results among all the classifiers *i.e.* 100% (accuracy).

$$\text{Overall Accuracy (OV)} = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (2)$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (3)$$

$$G - \text{Mean} = \sqrt{\text{Specificity} * \text{Sensitivity}} \quad (4)$$

Table 3: Results of SVM techniques using different performance matrices

Methods	Overall accuracy (%)	Specificity (%)	Sensitivity (%)	G-mean (%)	ROC curve (AUC) (%)	Training Time
Linear-SVM	97.90	99.06	96.82	97.93	99.43	00:02:57
Quadratic-SVM	100.00	99.74	99.45	99.59	99.93	00:01:39
Cubic-SVM	100.00	99.60	99.53	99.56	99.90	00:13:33
Fine Gaussian-SVM	99.40	98.89	99.81	99.35	99.98	00:01:09
Medium Gaussian- SVM	99.40	99.74	99.15	99.44	99.95	00:00:39
Coarse Gaussian SVM	97.50	97.64	97.37	97.50	99.49	00:00:31

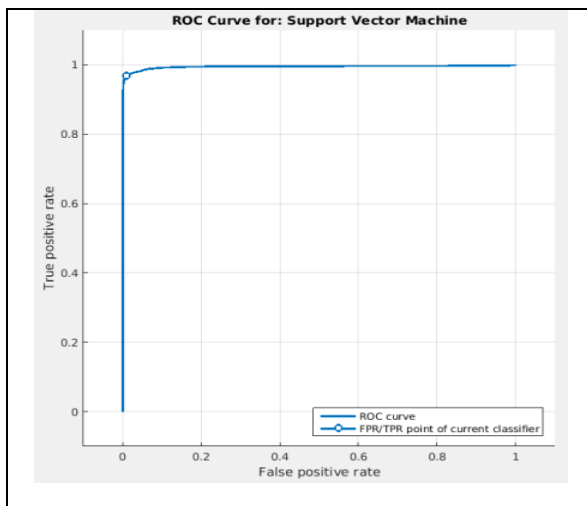


Fig. 11: Roc curve for Linear SVM

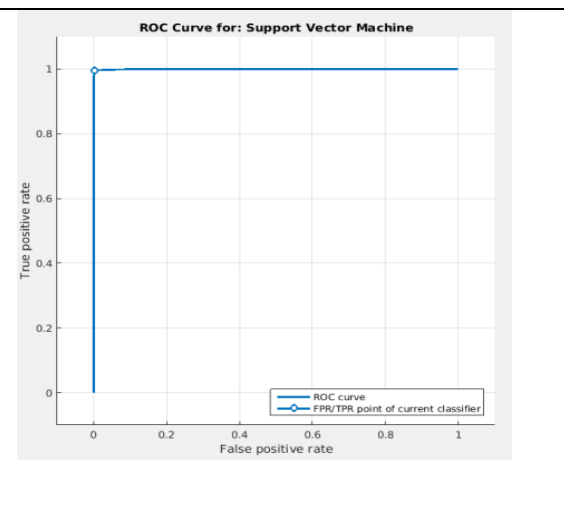


Fig. 12: Roc curve for Quadratic SVM

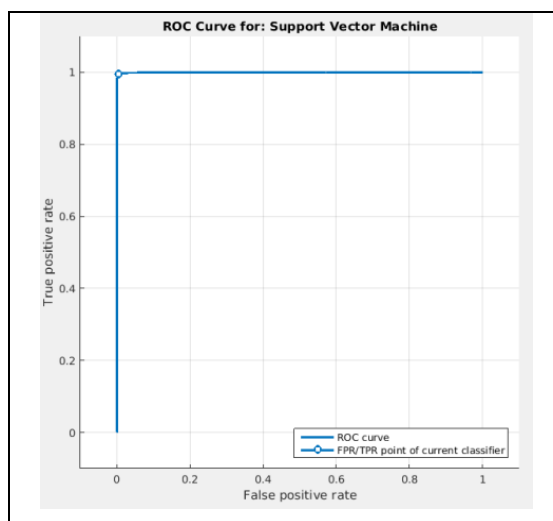


Fig. 13: Roc curve for Cubic SVM

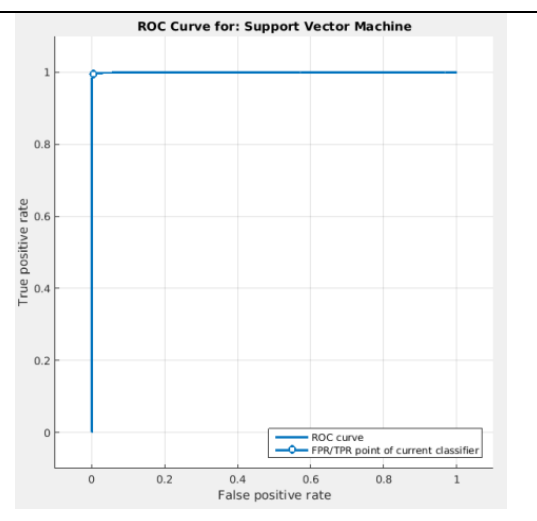


Fig. 14: Roc curve for Fine Gaussian SVM

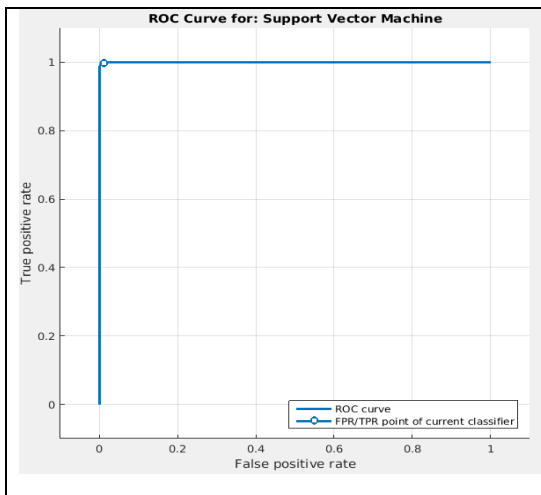


Fig. 15: Roc curve for medium Gaussian SVM

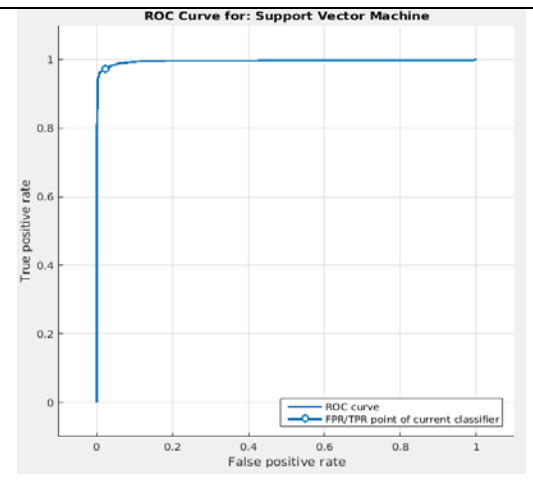
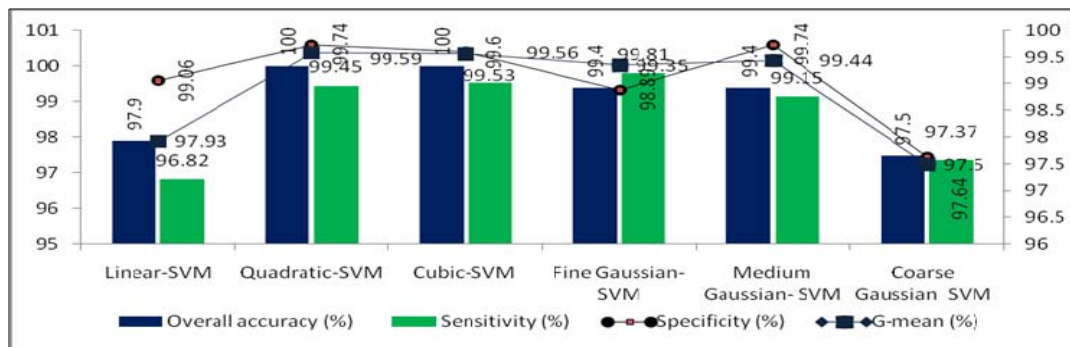


Fig. 16: Roc curve for Coarse Gaussian SVM

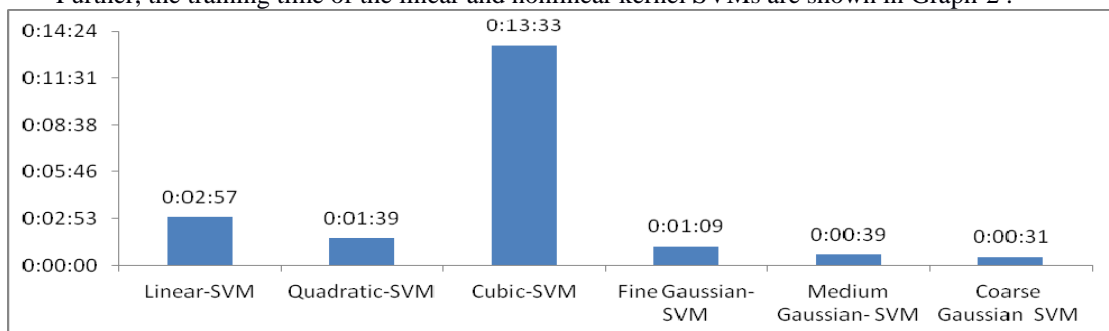
6. Result Analysis

In these experiments, six different types of support vector machine classifiers are used. The performances are measured using various kinds of performance matrices i.e. (Accuracy, Specificity, Sensitivity, G-mean, and ROC curve). Both the quadratic and cubic SVMs have shown the highest results i.e., 100% (accuracy). The quadratic SVM has shown the best result among all the classifiers i.e., 99.59% (G-mean) and the Fine Gaussian SVM has given the maximum result i.e., 99.98% (ROC curve). The Graph-1 placed below shows the results of SVM techniques using various types of performance.



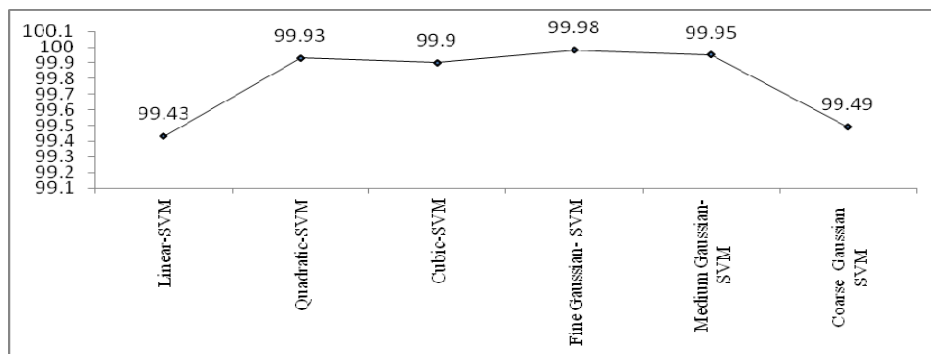
Graph-1: Results of SVM techniques using various type performance matrices

Further, the training time of the linear and nonlinear kernel SVMs are shown in Graph-2 .



Graph-2: Training time of the linear and nonlinear kernel SVMs

Graph-3 visualises the performances of ROC Curve for different liner and non-liner SVM classifiers.



Graph-3: Performance of ROC Curve for different linear and nonlinear kernel SVMs

7. Discussions and Conclusion

In this paper, an effective intrusion detection framework has been proposed and tested by using the NSL-KDD dataset in which six types of different SVM classifiers [Wu “*et al.*” (2008); Burges (1998); Bottou and Lin (2007); Kecman (2001); Alex and Andrew (2000)] such as, Linear-SVM, Quadratic-SVM, Cubic-SVM, Fine Gaussian-SVM, Medium Gaussian- SVM and Coarse Gaussian- SVM are used for the experiments. To measure the experimental results, different performance matrices are used such as, (i) accuracy, (ii) specificity, (iii) sensitivity, (iv) G-mean and (v) ROC curve. A full 41 features of NSL-KDD dataset was used throughout the experiments. Further, all the experiments are done by using MATLAB [Bryant and Garber (1999); Sumathi and Paneerselvam (2010)]. The 5 fold cross-validation method is used where 4 fold are kept for training purpose and 1 fold is taken for testing purpose.

Table-1 visualises the information about different classifiers such as, prediction speed (for both binary and multiclass) memory usage, interpretability and modern flexibility for all the classification methods after performing the experiments. Table-2 expresses the parameters and the training time of linear and nonlinear kernel SVMs after train the dataset. From this table, we deduced that Coarse Gaussian SVM is taking the optimal time i.e., 00:00:31 to train the dataset.

Finally, Table-3 and its corresponding Graph-1, 2 and 3 shows the results of different SVMs techniques using different performance matrices [Caruana and Niculescu-Mizil (2004)] i.e. (1) Overall Accuracy, (2) Specificity, (3) Sensitivity, (4) G-Mean and (5) ROC curve (AUC) [Hans and Kamber(2006)]. Both the quadratic and cubic SVMs show the highest results i.e 100% (accuracy). The quadratic SVM has shown the best result among all the classifiers i.e. 99.59% (G-mean) and the Fine Gaussian SVM has given the maximum result i.e. 99.98% (ROC curve). Again Quadratic and Medium Gaussian- SVM, shows the best results i.e. 99.74% (specificity) and Fine Gaussian- SVM gives the best result i.e. 99.81% (sensitivity).

In IDS accurate detection of various types of attack is a difficult problem, so requiring the analysis of large sets of IDS data. To represent samples from a large dataset is very important to detect intrusions in the field of network security. In this paper, the experiments are done by using six different types of SVMs with NSL-KDD dataset. The graphs for the ROC curve are also shown. The accuracy results using performance matrices are good for both the quadratic and cubic SVMs. But there are many future directions to solve the intrusion detection problems like using of hybrid soft computing techniques, handling multi-class problem [Rout “*et al.*” (2017)] reduce time complexity, decrease the cost, and tackle different types of other attacks.

References

- [1] Aggarwal, P. and Sharma, S.K. (2015). Analysis of KDD Dataset Attributes-Class wise for Intrusion Detection, *Procedia Computer Science*. 57: 842-851.
- [2] Alex, M. and Andrew. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* by Nello Cristianini and John Shawe-Taylor. Cambridge University Press, Cambridge. xiii+ 189: 687-689.
- [3] Al-mamory, S. O. and Jassim, F. S. (2015). On the designing of two grains levels network intrusion detection system, *Karbala International Journal of Modern Science*. 1(1): 15-25.
- [4] Bamakan, S. M. H., Wang, H., Yingjie, T. and Shi, Y. (2016). An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization, *Neurocomputing*. 199: 90-102.
- [5] Belavagi, M.C. and Muniyal, B. (2016). Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection, *Procedia Computer Science*. 89: 117-123.
- [6] Beniwal, S. and Arora, J. (2012). Classification and feature selection techniques in data mining, *International Journal of Engineering Research and Technology*. 1(6):1-6.
- [7] Bottou, L. and Lin, C.J. (2007). Support vector machine solvers. *Large scale kernel machines*.1-27.
- [8] Bryant, M.L. and Garber F.D. (1999). SVM classifier applied to the MSTAR public data set. *AeroSense'99*. International Society for Optics and Photonics. 3721:1-6.

- [9] Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery* 2. 121-167.
- [10] Campos, M.M. , Stengard, P.J. and Milenova, B.L. (2005). Creation and Deployment of Data Mining-Based Intrusion Detection Systems in Oracle Database 10g, In: *Proc. of the 4th Int. Conf. on Machine Learning and Applications*: 97-104.
- [11] Caruana, R. and Niculescu-Mizil, A. (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. *Proc. of the 10th ACM SIGKDD int. conf. on Knowledge discovery and data mining*. 69-78.
- [12] Depren, O., Topallar, M. , Anarim, E. and Ciliz, M.K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert systems with Applications*. 29(4): 713-722.
- [13] Dokas, P. , Ertoz, L. , Kumar, V. , Lazarevic, A. , Srivastava, J. and Tan, P.N. (2002). Data mining for network intrusion detection. In *Proc. NSF Workshop on Next Generation Data Mining*: 21-30.
- [14] Duquea, S. and Omar, M.N. (2015). Using data mining algorithms for developing a model for intrusion detection system (IDS), *Procedia Computer Science*. 61: 46-51.
- [15] Folino, G. and Sabatino, P. (2016). Ensemble based collaborative and distributed intrusion detection systems: a survey, *Journal of Network and Computer Applications* 66. 1–16.
- [16] Hans, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, Burlington, MA.
- [17] Hearst, M.A., Dumais, S.T. , Osuna, E. , Platt, J. and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications*. 13(4): 18-28.
- [18] Hoz, Eduardo De la, Hoz, Emiro De La , Ortiz, A., Ortega, J. and Prieto, B. (2015). PCA filtering and probabilistic SOM for network intrusion detection, *Neurocomputing*. 164: 71-81.
- [19] <https://in.mathworks.com/help/stats/parallelcoords.html>
- [20] <https://people.ece.cornell.edu/land/PROJECTS/Inselberg/>
- [21] Inayat, Z., Gani, A., Anuar, N.B., Khan, M. K. and Anwa, S. (2016). Intrusion response systems: foundations, design, and challenges, *Journal of Network and Computer Applications* 62. 53–74.
- [22] Jabez, J. and Muthukumar, B. (2015). Intrusion Detection System (IDS): Anomaly detection using outlier detection approach, *Procedia Computer Science*. 48: 338-346.
- [23] Joachims, T. (2006). Training linear SVMs in linear time. *Proc. of the 12th ACM SIGKDD int. conf. on Knowledge discovery and data mining*.
- [24] Kabir, E., Hu, J., Wang, H. and Zhuo, G. (2017). A novel statistical technique for intrusion detection systems, *Future Generation Computer Systems*.
- [25] Kabiri, P. and Ghorbani, A.A. (2005). Research on Intrusion Detection and Response: A Survey. *International Journal of Network Security*. 1(2): 84–102.
- [26] Kecman, V. (2001). *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. MIT press.
- [27] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *International Joint Conference on Artificial Intelligence*. 14(2): 1995.
- [28] Lee, W., Stolfo, S. J. and Mok, K.W. (1999). A data mining framework for building intrusion detection models. *Security and Privacy, Proc of the 1999 IEEE Symposium on IEEE*.1-13.
- [29] NSL-KDD Data. <https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/>
- [30] Rout, N. , Mishra, D. and Mallick, M. K. (2017). Analysing the Multi-class Imbalanced Datasets using Boosting Methods and Relevant Information. *International Journal of Computer Technology and Applications*. 10(9): 907-921.
- [31] Schölkopf, B., Burges, C.J.C. and Smola, A.J. (1999). *Advances in kernel methods: support vector learning*. MIT press.
- [32] Sengupta, N., Sen, J., Sil, J. and Saha, M. (2013). Designing of on line intrusion detection system using rough set theory and Q-learning algorithm, *Neurocomputing*. 111:161-168.
- [33] Sumathi, S. and Paneerselvam, S. (2010). *Computational intelligence paradigms: Theory and Applications using MATLAB*. Boca Raton, FL, USA: CRC Press.
- [34] Tan, Y. and Wang, J. (2004). A support vector machine with a hybrid kernel and minimal Vapnik-Chervonenkis dimension. *IEEE Transactions on knowledge and data engineering*. 16(4): 385-395.
- [35] Wu, X. , Kumar, V. , Ross Quinlan, J. , Ghosh, J. , Yang, Q. , Motoda, H. , McLachlan, G. J. , Ng, A. , Liu, B. ,Yu, P.S. ,Zhou, Z.H. ,Steinbach, M. ,Hand, D.J. , Steinberg, D. (2008). Top 10 algorithms in data mining." *Knowledge and information systems*.14: 1-37.
- [36] Zainal, A., Maarof, M.A. and Shamsudin, S.M. (2006). Research Issues in Adaptive Intrusion Detection, In *Proc of the 2nd Postgraduate Annual Research Seminar (PARS'06)*:138-146.
- [37] Zhu, W. , Zeng, N. and Wang, N. (2010). Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations. *NESUG proc. health care and life sciences*, Baltimore, Maryland: 1-9.
- [38] Zien, A. , Rätsch, G. , Mika, S. , Schölkopf, B. , Lemmen, C. , Smola, A. , Lengauer, T. and Müller, K. R. (2000). Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*. 16(9): 799-807.