ISSN 1870-4069

Selección de características con método wrapper para un sistema de detección de intruso: caso CICIDS-2017

Edson Manuel Ortiz Martínez, Pedro Arguijo, Antonio Hiram Vázquez López, Roberto Ángel Meléndez Armenta

Tecnológico Nacional de México,
Instituto Tecnológico Superior de Misantla,
México

edsonortizm@gmail.com, pedro_arguijo@excite.com,
{jahvazquezl, ramelendeza}@misantla.tecnm.mx

Resumen. Los sistemas de detección de intrusos (IDS) monitorean los eventos de un sistema en una red, siendo responsables de distinguir entre el tráfico normal y el anormal que se considera como algún tipo de ataque. El IDS comúnmente trata con una gran cantidad de tráfico de datos, lo que implica características irrelevantes y redundantes. Consecuentemente, la selección de características es uno de los factores prominentes que influyen en la calidad del IDS. En este trabajo, se analiza el desempeño de un IDS al seleccionar características con dos algoritmos estándar de envoltura (wrapper) Permutation Importance y Boruta, utilizando diversos clasificadores para probar su eficacia. Los resultados obtenidos sugieren que la precisión de los algoritmos considerados no varía al seleccionar distintos subconjuntos de atributos del dataset CICIDS 2017.

Palabras clave: Detección de intrusos, aprendizaje máquina, selección de atributos.

Feature Selection with a Wrapper Method for Intrusion Detection System: Case CICIDS-2017

Abstract. Intrusion Detection Systems (IDS) monitor the events in a network system, being responsible for distinguishing between normal and abnormal traffic which can be considered as an attack. IDS commonly handle a large amount of data traffic, which involves irrelevant and redundant features. Consequently, feature selection is one of the most prominent factors that affect the IDS performance. In this work, IDS performance is analyzed considering feature selection with two standard wrapper algorithms Permutation Importance and Boruta, using various classifiers to test its efficiency. The obtained results suggest that the accuracy of considered algorithms does not vary when selecting different subsets of attributes from the CICIDS 2017 dataset.

Keywords: Intrusion detection, machine learning, feature selection.

1. Introducción

El internet es una herramienta presente en nuestra vida diaria, como resultado de su introducción a diversos ámbitos; entretenimiento, educación, finanzas, etc., un mayor número de usuarios, y en especial empresas, invierten en plataformas de seguridad informática como son antivirus, cortafuegos y otras soluciones para hacerle frente a las amenazas informáticas. Existen antecedentes históricos sobre grandes pérdidas económicas y de datos sensibles debido a amenazas cibernéticas como lo son ataques de denegación de servicio, accesos no autorizados, suplantación de identidad entre muchas otras más amenazas cibernéticas.

Una conocida solución de seguridad informática en un entorno de red es el sistema de detección de intrusos (IDS, por sus siglas en inglés), el cual puede ser un dispositivo o programa que tiene como finalidad detectar amenazas cibernéticas [1]. Un IDS monitorea el tráfico y lo clasifica en dos posibles tipos: normal y malicioso [2] siendo el tráfico malicioso las amenazas cibernéticas y el normal las actividades bien intencionadas. También etiqueta el tráfico y permite a los encargados de la seguridad realizar acciones cuando se presenten ataques cibernéticos [3]. Sin embargo, el IDS funge como un elemento secundario, de apoyo, para otras herramientas orientadas a la seguridad informática como cortafuegos, antivirus, etc. La detección dentro de un IDS se lleva a cabo de dos maneras: mediante firmas que identifican el ataque basándose en patrones previamente establecidos o a través de las características del tráfico de red, ya que el tráfico normal será diferente del anómalo [4].

La detección de tráfico anómalo dentro de los IDS se realiza utilizando diferentes metodologías entre las que sobresale la inteligencia artificial (IA), la cual además es una alternativa útil para detectar ataque desconocidos [5]. La IA utiliza algoritmos de clasificación motivo por el cual la selección de características es un procedimiento útil que evita el sobre entrenamiento, mejora la velocidad de procesamiento y en general el rendimiento de los algoritmos de clasificación [6].

Se han realizado diferentes propuestas que utilizan algoritmos de IA enfocadas a detectar tráfico anómalo en los IDS, para la implementación de éstas destacan los dataset KDD-99 y Kyoto. DARPA propuso KDD-99 y contiene datos sintéticos creados en un entorno militarizado, en este conjunto de datos figuran ataques y tráfico benigno. A pesar de su antigüedad todavía sigue siendo un punto de partida para probar metodologías orientadas a la implementación de IDS, además de existir una versión mejorada NSL-KDD 99 que elimina redundancias y equilibra mejor la presencia de ataques haciéndola más equitativa. Algunas propuestas realizadas involucran la generación de reglas para detección de amenazas utilizando IA con algoritmos genéticos (AG) [7], otras aplican algoritmos evolutivos para mejorar la clasificación ya sea mediante la selección de características [8] o a través de la mejora de los parámetros de clasificadores figurando SVM junto PSO [9-10]. Pawar utiliza NSL-KDD, y realiza la selección de carcterísticas considerando la información de ganancia para mejorar el clasificador [7].

S. Mukherjee utiliza los datos de NSL-KDD e implementa un clasificador bayesiano junto con reducción de características, considerando la correlación entre los atributos seleccionados con la clase objetivo, pero baja entre los mismos atributos, además implementó Information Gain y Gain Ratio [11].En este sentido, otro dataset utilizado para implementar algoritmos de detección es el Kyoto, que tiene como ventaja utilizar

atributos implementados en el dataset KDD-99 pero capturando tráfico real en un amplio periodo de tiempo [13]. Además, Najafabadi clasificó el dataset de Kyoto considerando como entrenamiento el primer día y los demás para la evaluación del modelo, la selección de características la realizó con chi-cuadrada y para clasificar empleó tres algoritmos: k-NN, Decision Tree J48 y Naïve Bayes [13]. Finalmente, Amar sugiere una clasificación utilizando la nomenclatura de -1, 1 y -2. Él dataset lo divide en tres partes para selección de atributos, para aprendizaje y para validación, utiliza PCA y redes neuronales para la selección de atributos [14].

Aunque los dataset mencionados se han utilizado para la implementación de algoritmos de detección, cabe señalar que incluyen ataques cibernéticos desactualizados además de tráfico completamente sintético para el caso de KDD 99 y NSL-KDD, por ello es importante considerar otras alternativas a los dataset mencionados. El dataset CIC IDS 2017 presenta características importantes entre las que figuran un conjunto de ataques variados, un mapeo completo de la red, tráfico de red real, lo cual lo hace ideal debido a que este tipo de tráfico es el que se desea monitorear. En este documento, se analiza el efecto de la selección de características con los algoritmos Boruta y Permutation Importance en el dataset CIC IDS 2017 y se evalúa el desempeño de los subconjuntos de atributos obtenidos con diferentes clasificadores. En la sección dos se describen los métodos de selección de características, así como los algoritmos de clasificación utilizados. Tanto el dataset como el procedimiento de este documento se describen en la sección tres. Los resultados y conclusiones se abordan tanto las secciones cuatro y cinco, respectivamente.

2. Marco teórico

2.1. Selección de atributos

La selección de características es el proceso de seleccionar un subconjunto de características relevantes para construir modelos de aprendizaje robustos. La selección de atributos se clasifica en tres tipos: wrapper, filter e híbridos. El primero emplea un algoritmo de aprendizaje automático para evaluar la fiabilidad de un conjunto de características; entre estos destacan dos algoritmos a mencionar Boruta e Importancia de permutación (Permutation importance). Mientras que el método filter utiliza las características de los datos para evaluar su importancia o rango por medida de distancia, medidas de correlación, medidas de consistencia y medida de información. Finalmente, los métodos híbridos son una combinación de los dos anteriores, siendo útiles cuando existe un gran número de atributos para usar algoritmos del tipo wrapper y el rendimiento del enfoque basado en filtros no es satisfactorio. Este tipo de algoritmo pone en marcha un filtrado para medir la importancia de cada atributo. Posteriormente, dentro del conjunto total se seleccionan diferentes subconjuntos de atributos y se evalúan desde el punto de vista de wrapper.

Boruta. Es un algoritmo de tipo wrapper el cual funciona alrededor de Random Forest y es capaz de trabajar con cualquier método de clasificación que pueda aplicar medidas de importancia de la variable. A continuación, se describe el funcionamiento del algoritmo Boruta:

CONJUNTO ORIGINAL			
Atributo 1	Atributo 2	Atributo 3	Atributo 4
1	2	3	4
9	7	5	4

CONJUNTO SOMBRA			
Atributo 1	Atributo 2	Atributo 3	Atributo 4
2	1	4	3
7	9	4	5

Fig. 1. Conjunto de datos sombra y original.

Atributos		
A	B	C
1	4	3
2	5	4
3	6	5

Atributos		
A	B	C
1	0	3
2	0	4
3	0	5

Fig. 2. Conjunto de atributos, la tabla izquierda contiene el conjunto original y la derecha el conjunto con la variable B permutada.

1. En la Fig. 1 se observa que el algoritmo añade aleatoriedad al conjunto de datos al crear copias barajadas de un número determinado de características (denominadas características sombra u ocultas), por ejemplo, el atributo 1 con valores originales de 1 y 9, son asignados con valores sombra de 2 y 7.
2. El siguiente paso es entrenar un clasificador de Random Forest en el conjunto de datos extendido y aplica una medida de importancia de la característica (el valor predeterminado es Precisión de disminución media) para evaluar la importancia de cada característica, cuando más reducción exista mayor significancia tendrá.
3. En cada iteración, verifica si una característica real tiene mayor importancia que la mejor de sus características sombra (es decir, si la característica tiene una puntuación *Z* más alta que la puntuación *Z* máxima de sus características sombra) y elimina constantemente las características que se consideran poco importantes.
4. Finalmente, el algoritmo se detiene cuando se confirman o rechazan todas las características o cuando alcanza un límite especificado de Decision Tree.

Importancia de la permutación. Es calculada una vez que un algoritmo de clasificación es entrenado. Determina la importancia de los atributos computando el error obtenido en ausencia o ruido de una variable (ver Fig. 2). Si el error es menor o la diferencia es poco significativa cuando se permuta una variable en comparación a cuando no está permutada, entonces la variable no es (muy) importante. Este proceso se lleva a cabo con cada una de las variables para poder calcular la importancia de cada atributo y crear subconjuntos. Los atributos con importancia nula o muy baja pueden ser descartados de manera automática.

2.2. Algoritmos de clasificación

Naïve Bayes. El clasificador ingenuo de Bayes opera bajo un fuerte supuesto de independencia entre los atributos, lo cuales pueden ser tanto continuos como discretos o una combinación de ambos. La independencia de los atributos implica que la probabilidad de un atributo no afecta a la probabilidad del otro. El funcionamiento del clasificador Bayesiano es el siguiente:

1. Los datos de entrenamiento son particionados basándose en las etiquetas de cada clase.

2. Después del proceso de entrenamiento, cuando entre un dato desconocido se buscará la probabilidad a posteriori para cada una de las clases posibles para la tupla desconocida, la clase con mayor posibilidad de existir para una tupla será seleccionada.

Decision Tree. Este tipo de algoritmos crea una estructura en forma de árbol para un determinado conjunto de datos, donde cada nodo representa un atributo y las hojas representan los resultados o clases. Cada nodo debe pertenecer a datos de tipo categórico y en caso de ser continuos se debe discretizar antes del proceso de categorización. La construcción de un árbol se lleva a cabo mediante el método se divide y vencerás, donde se comienza con un nodo principal que contiene el conjunto total de datos y posteriormente se van dividiendo los elementos del nodo en subconjuntos que se localizarán dentro de otros nodos y estos a su vez se dividirán de manera iterativa hasta lograr homogeneidad en los nodos hoja.

Decision Tree tipo CART. Es un tipo de Decision Tree binario que se construye mediante la división de un nodo en dos nodos hijo repetidamente, comenzando con la raíz que contiene todas las muestras para el aprendizaje. Consiste en tres pasos que se repiten de manera iterativa:

1. Buscar el mejor punto para dividir cada atributo.
2. Encontrar el mejor nodo con base en el aumento de la homogeneidad en los nodos hijo.
3. Si las condiciones no son satisfechas, dividir de nuevo comenzando con el paso uno.

El algoritmo continuará si todavía no existe la suficiente homogeneidad dentro de las muestras en los nodos hijo, esto se puede calcular con diversas medidas como lo es la entropía y el índice Gini. También es posible detener el algoritmo si el número de muestras que queda por dividir no es muy grande.

Random Forest. Este algoritmo es un meta clasificador, es decir, ocupa un determinado número de clasificadores débiles para construir uno fuerte. En especial, los Random Forest implementan un conjunto de Decision Tree. Para propósitos de clasificación, al haber un dato entrante cada Decision Tree existente realiza de manera individual un proceso de clasificación votando por una clase en especial, la clase con un mayor número de votos es la ganadora.

AdaBoost. Es un meta clasificador de tipo boosting que permite a clasificadores débiles mejorar su rendimiento mediante el uso de varios clasificadores, ya que adapta el error de clasificación de los algoritmos básicos asignándoles un peso a cada uno.

3. Materiales y métodos

3.1. Dataset

La base de datos CIC IDS 2017 se creó en la Facultad de Ciencias de la Computación de la Universidad de Nueva Brunswick en 2017. Contiene un total de 2,827,876 paquetes con 79 atributos que en conjunto alcanzan 5 días de datos reunidos en un periodo mayor a 7 días de actividad de red. Este dataset contiene tráfico benigno y ataques, los cuales representan datos del mundo real, además tener datos de ataques relativamente recientes como HeartBleed, por ejemplo.

Entre los atributos que contiene se destacan la dirección IP, puerto, protocolo, tipo de servicio, etc. También contiene resultados de análisis de red obtenidos mediante la herramienta CICFlowMeter con sus respectivos atributos dentro de un archivo CSV. Para este dataset su utilizó la representación de la actividad perteneciente a 25 usuarios con protocolos HTTP, HTTPS, FTP, SSH, y protocolos de correo electrónico. Se capturaron datos en diferentes días de la semana y a diferentes horas del día. Cada fecha de captura contuvo diferentes ataques además de tráfico normal de red. Los ataques que se incluyen son: Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS [15].

Los criterios implementados para la realización de esta base de conocimientos incluyeron una amplia variedad de circunstancias las cuales involucran una arquitectura de red con switches, enrutadores, módem, cortafuegos además de múltiples sistemas operativos: MacOS, Windows y Ubuntu. La interacción entre los dispositivos de red fue amplia además de que el tráfico resultante fue mapeado completamente y la cantidad de ataques llevada a cabo fue diversa [15].

3.2. Utilerías

Se utilizó una laptop HP modelo 17z-db000, con sistema operativo Windows 10, procesador AMD Ryzen 5 2500U con 16 GB de memoria RAM (DDR4) y un disco duro de 916 GB. Además, se programó en Python 3.7 para 64 bits. Para el preprocesamiento, división de dataset y clasificación se empleó la librería Scikit-learn. Para implementar el selector de características Boruta se usó la librería BorutaPy, la cual a su vez envolvía al algoritmo Random Forest (perteneciente a Scikit-learn), también, se aprovechó la librería eli5 para el selector de atributos Permutation Importance.

3.3. Procedimiento

El procedimiento propuesto involucró los siguientes pasos: normalización, división de la base de datos, selección de características, entrenamiento y validación tal y como se muestra en la Fig. 3.

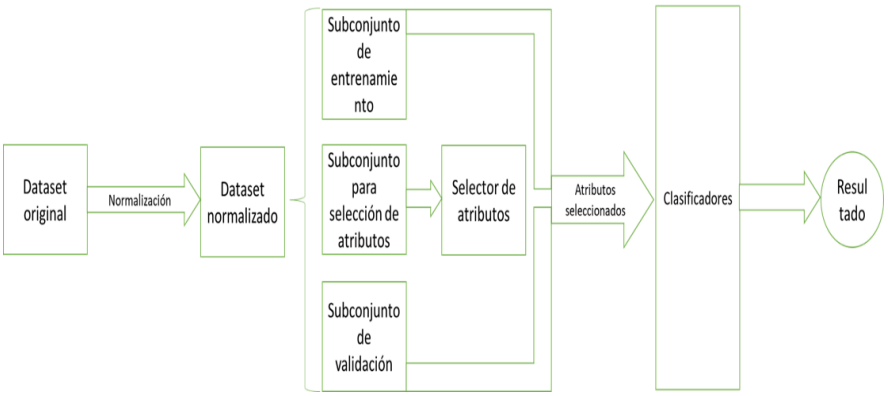

Fig. 3. Descripción del método propuesto.

Tabla 1. Precisión de los algoritmos de clasificación señalados considerando todos los atributos del CIC IDS 2017.

Clasificador	Precisión
Decision Tree	99.81%
Random Forest (64 clasificadores)	99.82%
AdaBoost (100 perceptrón)	85.33%
Multinoamial Naïve Bayes	84.85%
Gaussian Naïve Bayes	70.57%
AdaBoost (100 árboles)	85.69%

El dataset se dividió en tres subconjuntos, 20% para entrenamiento, 20% para selección de características y 60% para pruebas. Para la selección de características se consideraron los algoritmos de Boruta y Permutation Importance, descritos anteriormente. Los clasificadores implementados fueron Decision Tree, AdaBoost, Random Forest y Naïve Bayes. Cabe mencionar que el desempeño de cada algoritmo se analizó con y sin selección de atributos, es decir, con el total de atributos del dataset. Se llevaron a cabo pruebas con los mejores atributos partiendo desde el primer atributo, segundo y de manera iterativa hasta obtener el conjunto total de atributos, excepto con el algoritmo Boruta, ya que éste entrega un conjunto de características óptimo.

4. Resultados

En esta sección se muestran los resultados obtenidos al aplicar los algoritmos previamente descritos al dataset CIC IDS 2017. En la Tabla 1 se puede observar la precisión obtenida en cada uno de los clasificadores al considerar todos los atributos del dataset, dado que la precisión es una métrica utilizada en la eficacia de los

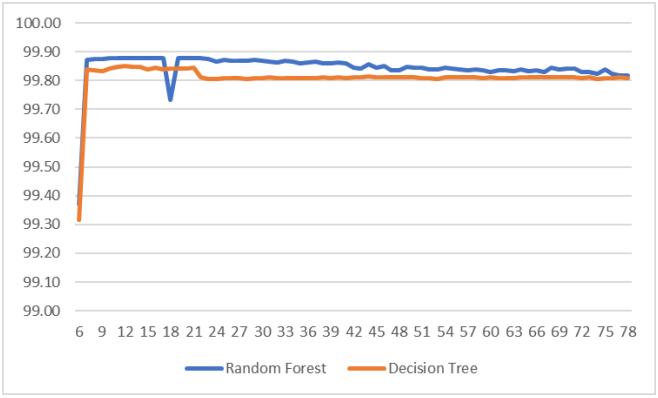

Fig 4. Comparación de la precisión entre algoritmos de Random Forest y Decision Tree.

Tabla 2. Precisión de diferentes algoritmos considerando la cantidad de mejores atributos indicados.

	Atributos						
Clasificadores	20	30	40	50	60	70	Todos
Random Forest	99.88 %	99.87 %	99.86 %	99.85 %	99.83 %	99.84 %	99.82 %
Decision Tree	99.84 %	9981%	99.81 %	99.81 %	99.81 %	99.81 %	99.81 %
Gaussian Naïve Bayes	71.42 %	72.24 %	73.04 %	71.23 %	70.52 %	69.53 %	70.57 %
Multinomial Naïve Bayes	86.47 %	87.12 %	87.04 %	86.56 %	86.60 %	86.35 %	84.84 %
AdaBoost (100 árboles)	85.70 %	85.69 %	85.69 %	85.69 %	85.69 %	85.69 %	85.69 %
AdaBoost (100 Perceptrones)	92.20 %	91.17 %	89.45 %	98.23 %	98.26 %	96.69 %	96.48 %

clasificadores. Los resultados indican que Random Forest arrojó la mejor clasificación seguido por Decision Tree. En estos dos casos la precisión supera el 99%.

En la Tabla 2 se exhibe la precisión de los diferentes clasificadores utilizando las *n* mejores características obtenidas con Permutation Importance. Tanto Decision Tree como Random Forest arrojan resultados de clasificación del 99% a partir de utilizar las 6 mejores características. Bayes Multinomial muestra un desempeño del 87% al considerar los primeros 38 atributos. Mientras que AdaBoost con árboles muestra una precisión constante desde los 27 atributos considerados para la clasificación y manteniéndose sin cambios posteriormente sin importar el número de atributos que se agreguen.

Al agregar cada vez más características los clasificadores tuvieron una tendencia a incrementar su precisión, pero ocasionalmente presentaron intervalos con pequeños pero significativos decrementos y/o incrementos que contenían máximos globales.

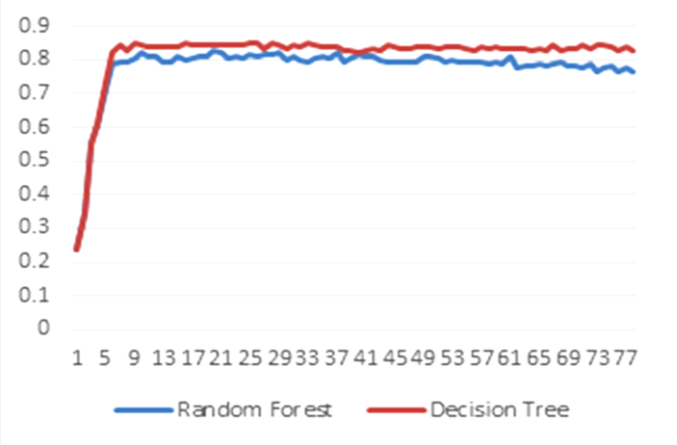

Fig 5**.** Comparación de la sensibilidad entre algoritmos de Random Forest y Decision Tree.

Tabla 3**.** Numero de atributos óptimos para cada clasificador.

Clasificador	**Permutation Importance Precisión, N atributos**	**Boruta**
Decision Tree	99.85%, N=12	99.81%
Random Forest (64 árboles)	99.88%, N=13	99.82%
AdaBoost (100 perceptrón)	92.56%, N=45	74.51%
Multinomial Naïve Bayes	87.32%, N=38	86.00%
Gaussian Naïve Bayes	75.19%, N=26	71.97%
AdaBoost (100 árboles)	85.70%, N=21	85.69%

Como se puede verificar en la Fig. 4, los algoritmos de Decision Tree y Random Forest superaron el 99% de precisión. Ambos exhibieron un desempeño relativamente estable, el cual no variaba demasiado al incrementar el número de atributos. También se puede observar una tendencia por parte del Random Forest a superar levemente a Decision Tree. Por otro lado, el Decision Tree posee una mayor velocidad de procesamiento debido a que Random Forest al ser un meta algoritmo contiene múltiples Decision Tree, por lo cual realiza el proceso de clasificación con menor rapidez.

Por otra parte, Decision Tree mostró una mayor sensibilidad que Random Forest prácticamente en todo momento como se muestra en la Fig. 5.

Por su parte el algoritmo Boruta hizo una valoración de los atributos y únicamente los mejores se toman en cuenta para su uso. En este caso, se obtuvieron 62 atributos. La Tabla 3 muestra el número óptimo de atributos con cada algoritmo, haciendo el uso de Permutation Importance y Boruta.

La matriz de confusión de la Fig. 6 y la Fig. 7 muestran los resultados obtenidos al implementar Boruta junto a Decision Tree y Random Forest.

Fig. 6. Matriz de confusión resultante de la implementación de Decision Tree y Boruta.

5. Conclusiones

Esta investigación ha encontrado que la eficacia de los algoritmos de clasificación en la detección de ataques cibernéticos no está, necesariamente, influenciada a la complejidad de estos.

Los resultados demuestran una mayor eficacia en la clasificación del Decision Tree CART en relación con algoritmos más complejos como AdaBoost y Naïve Bayes, superando a la mayoría de los algoritmos implementados en este estudio, en términos de velocidad de clasificación (excepto de Multinomial Naïve Bayes) y a casi todos en precisión, siendo superado levemente por Random Forest por lo cual se puede argumentar que el Decision Tree CART es un candidato excelente para utilizarse en IDS basados en la detección de actividad anómala al que considera el dataset CIC IDS 2017.

Es importante señalar que se necesitan investigar más selectores de características, ya que no se incluyen los de tipo filter ni híbridos en este trabajo.

Fig 7. Matriz de confusión resultante de la implementación de Random Forest y Boruta.

Referencias

1. Stampar, M., Fertalj, K.: Artificial intelligence in network intrusion detection. In: Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1318–1321 (2015)
2. Gharaee, H., Hosseinvand, H.: A new feature selection IDS based on genetic algorithm and SVM. In: International Symposium on Telecommunications (IST), pp. 139–144 (2016)
3. Kanika, U.: Security of network using ids and firewall. International Journal of Scientific and Research Publications, 3(6), pp. 1–4 (2013)
4. Kanimozhi, V., Jacob, D.T.P.: Calibration of various optimized machine learning classifiers in Network Intrusion Detection System on the Realistic Cyber Dataset Cse-Cic-Ids2018 Using Cloud Computing. International Journal of Engineering Applied Sciences y Technology, 4(6), pp. 209–213 (2019)

5. Li, Z., Qin, Z., Shen, P., Jiang, L.: Zero-shot learning for intrusion detection via attribute representation. Neural Information Processing, Lecture Notes in Computer Science, pp. 352–364 (2019)
6. Zuech, R., Khoshgoftaar, T.M.: A survey on feature selection for intrusion detection. In: Proceedings of the 21st ISSAT International Conference on Reliability and Quality in Design, pp. 150–155 (2015)
7. Pawar, S.N., Bichkar, R.S.: Genetic algorithm with variable length chromosomes for network intrusion detection. International Journal of Automation and Computing, 12(3), pp. 337–342 (2015)
8. Li, J., Zhao, Z., Li, R., Zhang, H.: AI-based two-stage intrusion detection for software defined IOT networks. IEEE Internet of Things Journal, 6(2), pp. 2093–2102 (2019)
9. Gao, F.: Proceedings Qingdao Oulu. Academy Publ., pp. 21–22 (2009)
10. Chakir, E.M., Khamlichi, Y.I.: An effective intrusion detection model based on SVM with feature selection and parameters optimization. Journal of Theoretical and Applied Information Technology, 96(12) (2018)
11. Mukherjee, S., Sharma, N.: Intrusion detection using naive Bayes classifier with feature reduction. Procedia Technology, 4, pp. 119–128 (2012)
12. Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BIONETICS) (2016)
13. Najafabadi, M.M., Khoshgoftaar, T.M., Seliya, N.: Evaluating feature selection methods for network intrusion detection with kyoto. International Journal of Reliability, Quality and Safety Engineering, 23(1), pp. 1650001 (2016)
14. Ammar, A.: Comparison of feature reduction techniques for the binominal classification of network traffic. Journal of Data Analysis and Information Processing, 3(2), pp. 11–19 (2015)
15. University of New Brunswick: Datasets. https://www.unb.ca/cic/datasets/ids-2017.html (2020)