Detección de Intrusiones en Redes Industriales

Evaluación Experimental de Algoritmos de Aprendizaje de Máquina

Aldo Insfrán[1,4], Fabio López-Pires[2], Benjamín Barán[3], Eustaquio Martínez[1]

[1]Facultad Politécnica, Universidad Nacional del Este
[2]Facultad de Posgrados, Universidad Internacional Tres Fronteras
[3]Facultad de Informática, Universidad Comunera
[4]División de Ingeniería Electrónica y Sistemas de Control, Dirección Técnica, ITAIPU Binacional

100113 Ciudad del Este, Paraguay
{aldo.insfran, amartinez}@fpune.edu.py
fabio.lopez@uninter.edu.py
bbaran@ucom.edu.py
ajid@itaipu.gov.py

Resumen. Ataques cibernéticos a sistemas industriales de infraestructura crítica son una realidad en la actualidad y sus consecuencias constituyen un riesgo a la continuidad de los negocios, la economía y el bienestar de la población. En este sentido, este trabajo presenta un análisis de implementaciones de sistemas de detección de intrusiones para sistemas industriales y una evaluación experimental de un conjunto de algoritmos, utilizados en dicho tipo de sistemas, aplicando un conjunto de datos obtenido de un sistema industrial de infraestructura crítica. Dicho análisis da énfasis a cuestiones como, algoritmos y conjuntos de datos de evaluación utilizados, parámetros de entrenamiento, ataques ensayados y métricas de evaluación. La evaluación experimental se lleva a cabo sobre un conjunto nueve algoritmos de aprendizaje de máquina utilizando un conjunto de datos con siete tipos de ataques cibernéticos a la red de un sistema industrial del tipo gasoducto en el que se utiliza el protocolo de comunicaciones *modbus* para la supervisión y el control. Los resultados experimentales mostraron que los algoritmos basados en árboles de decisión arrojan los mejores resultados de clasificación para la métrica de *F1-Score*.

Palabras Claves: *Sistemas de Control Industrial*, *Sistemas de Detección de Intrusiones*, *Modbus*, *Aprendizaje de Máquina*, *Ciberseguridad*.

1 Introducción

Las plantas de generación de energía, las subestaciones eléctricas y ciertas manufacturas, consideradas infraestructuras críticas, tienen como componentes principales a los sistemas de control industriales (*Industrial Control Systems - ICS*).

Originalmente, la ciberseguridad en este tipo de sistemas era confiada al hecho de permanecer desconectados (*airgapped*) de los demás sistemas. En la actualidad, la integración cada vez mayor entre ambientes empresariales (*Information Tecnologies - IT*) e industriales (*Operational Tecnologies - OT*) y la necesidad de conexiones remotas representan desafíos al mantenimiento de la ciberseguridad.

En este escenario, considerando los daños y las consecuencias que podría acarrear el comprometimiento de una infraestructura crítica, numerosas iniciativas han surgido para proteger los mencionados *ICS*. Una de las estrategias propuestas es la defensa en profundidad [1] cuya implementación se establece en el estándar ISA/IEC 62443 a través de la creación de zonas y conductos. En las zonas correspondientes al proceso industrial se sugiere el uso de sistemas no invasivos de detección de intrusiones, que a diferencia de los sistemas de control de código malicioso convencionales, como los antivirus, no ejercen una acción posterior a la detección de una anomalía. Acorde con esta filosofía, varios trabajos han propuesto sistemas de detección de anomalías (*Anomaly Detection Systems - ADS*) para redes industriales basados en el análisis del tráfico de red, los llamados sistemas de detección de intrusiones (*Intrusion Detection Systems - IDS*), los cuales han mostrado su potencial en la detección de ciberataques a los *ICS*. Estos *IDS* utilizan normalmente algoritmos de aprendizaje de máquina (*Machine Learning - ML*) entre los que se diferencian los de aprendizaje clásico (e. g., árboles de decisión), los basados en redes neuronales (*Neural Networks - NN*) y otros de tipo combinado.

En el presente trabajo se lleva a cabo primeramente una revisión y análisis de trabajos relacionados a *IDS* utilizados en sistemas de supervisión y control industriales del tipo *Supervisory Control and Data Adquisicion* (*SCADA*). La composición genérica de este tipo de sistemas comprende: servidores principales de aplicación (*Main Terminal Units – MTU*), unidades terminales de agregación y control local (*Remote Terminal Units – RTU*), interfaces humano-máquina (*Human Machine Interfaces – HMI*), estaciones de ingeniería, y los dispositivos de planta como controladores de máquina, los sensores y actuadores. Seguidamente se realiza la evaluación experimental de un grupo de algoritmos de *ML* utilizando un conjunto de datos (*dataset*) extraído de una red en un sistema de control de un gasoducto (*pipeline*) [2]. El *dataset* es obtenido en condiciones de operación normal del sistema así como durante la ejecución de una serie de ataques de distintos tipos.

La organización del documento es como sigue: la Sección 1 introduce los conceptos básicos que explican de manera general el contexto y el propósito del trabajo; la Sección 2 presenta el análisis de los trabajos relacionados, los cuales se toman como base en la selección del *dataset* utilizado, los algoritmos a evaluar y la métrica de evaluación; la Sección 3 describe los componentes del análisis experimental; el procedimiento de generación de *datasets*, las pruebas y los resultados obtenidos; así también se señalan los principales hallazgos relacionados a dichos resultados planteando una discusión acerca de las características del *dataset* y las condiciones de parametrización de los algoritmos en cada caso. Finalmente, en la Sección 4 se presentan las conclusiones generales y se sugieren posibles trabajos futuros.

2 Trabajos Relacionados

En una revisión de la literatura sobre trabajos relacionados a *IDS* para sistemas industriales fueron encontradas varias propuestas de implementación de variantes de algoritmos de *ML*, las cuales son evaluadas utilizando *datasets* en ocasiones capturados específicamente para tal propósito y en otros casos obtenidos de otros trabajos, cuyo objetivo era el de crearlos con las debidas consideraciones [3] que permitan su uso para el entrenamiento/evaluacion de los *IDS*. Otros trabajos presentan estudios comparativos de estas propuestas, con los que, se orientan futuros esfuerzos de investigación.

En los siguientes apartados se analiza un conjunto de trabajos tanto de implementación de *IDS*, como de desarrollo de *datasets*. Dicho análisis delimita la selección de algoritmos, *datasets*, parámetros de entrenamiento (*features*) y métricas utilizadas en este trabajo.

2.1 *Datasets* para Evaluación de *IDS*

En cuanto a los protocolos de comunicaciones, directamente relacionado con la aplicación industrial del sistema del cual se genera el *dataset*, se ha encontrado que en la mayor parte de los trabajos, como en [2], [4] y [5], se utilizan variantes del protocolo *modbus* (*RTU* y *Transport Control Protocol – TCP*) ampliamente aplicado en varios sectores industriales como manufactura, *utilities* y energía. En la selección de los *features* que finalmente conforman estos *datasets,* se observa que todos presentan los campos de las unidades de datos de protocolo (*Protocol Data Unit – PDU*) además de etiquetas que identifican los mensajes maliciosos. Otros trabajos como [6] y [7] tratan sobre sistemas eléctricos en los que se hace uso de protocolos IEC 60870-5-104, donde los *datasets* están compuestos también por algunos campos de las PDU de las capas 2, 3, 4, y 7 de dichos protocolos, además de las etiquetas.

Entre los ataques que son ensayados, el más frecuente es el de tipo reconocimiento a nivel de aplicación, también conocido como ataque de *fingerprinting*, como se muestra en [2], [4], [6] y [7]. Como parte de este tipo de ataque se determinan las direcciones, códigos de función utilizados y el contenido de memoria (*COILs* y *Registers*) de los nodos. Otras variantes encontradas, relacionadas al reconocimiento, son el escaneo de direcciones de capa de red y de puertos de capa de transporte como en [6] y [7]. También, en [2], [4], [6] y [7], se han encontrado ataques de inyección de comandos y respuestas, todos llevados a cabo a través de una estrategia *man-in-the-middle* (*MITM*) con excepción de [4], en el que se ataca primeramente un nodo pivote (controlador) al que se carga (*load*) código malicioso. Los ataques de reconocimiento en [6] y [7] se llevan a cabo haciendo *spoofing* mediante la inyección de mensajes de lectura. En [2], [5], y [6] se ejecutan ataques de denegación de servicio (*Denial of Service – DOS*). En [2] se fabrican mensajes con valores incorrectos del código de redundancia (*Cyclic Redundancy Check – CRC*) de *modbus*, en [5] y [6] tales ataques se llevan a cabo usando *icmp*[1] y *tcp-syn flooding*, en [5] se utiliza además *modbus-query flooding*. Todos los trabajos describen básicamente la forma en la que son llevados los

[1] *Internet Control Message Protocol – ICMP*

ataques: [2] a través del uso de un *datalogger*; [4], [5], [6] y [7] muestran comandos de consola y en ocasiones otras herramientas utilizadas como *metasploit*[2].

Vale destacar la clasificación realizada en [8], en la que se diferencian cuatro clases de ataques: reconocimiento (*reconnaissance* - *RECON*), inyección de respuestas (*response injection* - *RI*), inyección de comandos (*command injection* - *CI*) y denegación de servicios (*denial-of-service* - *DOS*). También se muestran subclases para los *RI*; *Naive Malicious RI* (*NMRI*) y *Complex Malicious RI* (*CMRI*); y, subclases para los *CI*; *Malicious State CI* (*MSCI*), *Malicious Parameter CI* (*MPCI*), *Malicious Function-Code CI* (*MFCI*). Esta clasificación se utilizará como referencia para las discusiones sobre los tipos de ataque en este trabajo.

De todos estos trabajos que presentan *datasets*, en [2] se considera la mayor diversidad de ataques practicados a un *IDS*. Los 35 escenarios de ataque ensayados corresponden a los 7 tipos que forman parte de la clasificación llevada a cabo en [8]: *NMRI, CMRI, MSCI, MPCI, MFCI, RECON* y *DOS*. Todos estos corresponden a ataques llevados a cabo directamente sobre el protocolo industrial *modbusRTU*. Los datos capturados son: valores de campos del *PDU modbus*, estampas de tiempo y etiquetas de tipo de ataque. Tales características motivaron a la selección de este *dataset* como base para la evaluación experimental en este trabajo.

2.2 Propuestas de *IDS* Estudiados

De todos los trabajos estudiados que presentan implementaciones de *IDS* para *ICS*, el protocolo *modbus* es el más utilizado. En menor medida fueron encontrados casos en los que se analizan otros protocolos como los del IEC 60870-5-104 o del IEC 61850, correspondientes a aplicaciones en el sector eléctrico. Así mismo, en la mayor parte de los trabajos, al tiempo de proponer algoritmos o modelos de *IDS*, se generan *testbeds* y *datasets* propios. De los trabajos analizados, sólo [9], [10] y [11] utilizan datos generados en otros trabajos como los presentados en la Sección 2.1.

En cuanto a los *features* analizados en el proceso de entrenamiento de los algoritmos o modelos utilizados para la detección, se observa que en la mayoría de los trabajos se generan flujos de comunicaciones agrupando los mensajes obtenidos en los *datasets*. Tales flujos contienen *features* que son valores calculados a partir de los campos de los mensajes intercambiados en la red. En [10] y [11] se utilizan datos correspondientes a la capa de red como la cantidad y el tamaño de los paquetes, en [12] son utilizados valores obtenidos de los campos del protocolo de capa de aplicación del IEC 60870-5-104 (*Application Protocol Data Unit* - *APDU*). Adicionalmente, en [11] se considera la secuencia de transición de los mensajes. Otros trabajos calculan estos *features* directamente del conjunto de mensajes, sin agruparlos en flujos. En [13] se calculan longitudes e intervalos entre paquetes, en [14] además se hace un recuento de paquetes del tipo *ARP* (*Address Resolution Protocol*), con el mismo origen o destino. En [15] se analizan tiempos de respuesta y de retransmisión, mientras que en [16] se utilizan, entre otros, los tiempos de ida y vuelta. Algunos trabajos se centran exclusivamente en *features* calculados a partir de, valores obtenidos del protocolo industrial de capa de

[2] *https://www.metasploit.com/*

aplicación, parámetros del sistema de automatización (e. g., registros de un *Programmable Logic Controller – PLC*) y de mediciones de la planta. En [9] que hace uso del *dataset* de [2], estos *features* se seleccionan de entre los parámetros *modbus.* En [16] se utilizan valores de medición de voltaje mientras que en [17] se utiliza el código de función. Todos estos trabajos utilizan datos de red. En [18] sin embargo se utilizan datos obtenidos a partir del *Windows Performance Monitor* (*WPM*) en la estación de ingeniería.

Parte de los trabajos presenta escenarios de ataque a protocolos no industriales, o de capas inferiores a la de aplicación, en ellos se llevan a cabo ataques de tipo *MITM* con los que se hace *spoofing* de *ARP* para hacer *fingerprint* de los activos de red industriales como en [18]**,** o escaneos de red (*RECON*) con el uso de mensajes *TCP-FIN* [14]. También se observan ataques de tipo *DOS* a través de inundación con *TCP-SYN*. Otro conjunto de trabajos presenta además, ataques directamente ejecutados sobre los protocolos industriales. Algunos de ellos también ejecutados con una estrategia de *MITM* y *spoofing*; *fingerprinting* de los controladores a través de comandos de lectura de *modbus*, fabricación e inyección de comandos (*MFCI*) mediante comando de escritura *modbus* [17]; y, alteraciones de los valores leídos de campo a través de inyección de respuestas maliciosas (*NMRI* y *CMRI*) [16]. También se consideran ataques de exfiltración de datos [12] manipulando el campo *COT* del *ASDU* del IEC 60870-5-104. A diferencia del enfoque *MITM*, en [4] y [13] se ensayan cargas de código malicioso a un activo de red, desde el cual posteriormente se practican otros ataques.

2.3 Algoritmos de *IDS* Estudiados

La mayor parte de los trabajos revisados utiliza variantes de *Support Vector Machine* (*SVM*) como algoritmo base para sus implementaciones de *IDS* como por ejemplo [13]. También se utiliza en conjunto con procesos de optimización en la obtención de *features* como en [17], y en *clusters* con sus resultados agregados mediante otros algoritmos como el *k-means* en [14]. Otro tipo de algoritmos muy utilizados y que muestran buen desempeño son los árboles de decisión, encontrado en variantes con el uso de poda como *Reduced Error Pruning* (*REPTree*) en [15], o con métodos de agregación como *bagging* (*Random Forest – RF*) en [18] y *boosting* en [10] y [15]. También se encuentran algoritmos bayesianos, *Multinomial Naive Bayes* en [15] y de redes bayesianas en [9]. Por otro lado, [10] y [16] hacen uso de redes neuronales de tipo *Multi-Layer Perceptron* (*MLP*) y [17] además prueba funciones del tipo *Radial Basis Function* (*RBF*). A diferencia de estos trabajos, en [12] se propone la construcción de un autómata probabilístico y en [11] la definición de un modelo ciber-físico.

La Tabla 1 resume los algoritmos encontrados en las implementaciones de *IDS* examinadas. La Tabla 2 relaciona tales algoritmos con los trabajos donde fueron encontrados.

La métrica de evaluación más utilizada es la precisión de detección (*Accuracy*), seguida por los valores de *Precision*, *Recall* y la media armónica de estos valores *F1-Score*. En [18] se adopta directamente los valores de la matriz de confusión *TP* (*True Positives*), *TN* (*True Negatives*), *FP* (*False Positives*), *FN* (*False Negatives*), mientras

que [17] y [14] dan mayor importancia a los falsos positivos utilizando *false positive rate*. Adicionalmente en [17] se evalúa el tiempo de clasificación y en [15] el tiempo de creación del modelo.

FA[3]	**Cod.**	**Nombre**	**Cod.**	**Nombre**
***Decision Trees* (*DT*) [FA1]**	A1	*C4.5*	A2	*REPTree*
	A3	*Decision Stump Tree*	A4	*Ripple Down Rule*
	A5	*Degged Decision Stump*		
***Instance-Based* [FA2]**	A6	*KNN*	A7	*Linear SVM*
	A8	*One Class SVM*	A9	*Logically-Deep SVM*
***Probabilistic* [FA3]**	A10	*Logistic Regression*	A11	*Multinomial Naive Bayes*
	A12	*Bayes Net*	A13	*Bayes Point Machine*
***Dimensionality Reduction* [FA4]**			A14	*Linear Discriminant Analysis*
***Ensemble* [FA5]**	A15	*Decision Forest*	A16	*Bagged REPTree*
	A17	*Decision Jungle*	A18	*Boosted DT*
	A19	*k-means + SVM*		
***Neural Netw.* [FA6]**	A20	*MLP*	A21	*RBF-MLP*
	A22	*Averaged Perceptron*		
***Otros* [FA7]**	A23	*Finite State Machine*	A24	*Deterministic Probabilistic Automata*

Tabla 1. Algoritmos encontrados en las implementaciones de *IDS* estudiadas

Con respecto a los mejores resultados de detección obtenidos, [15] y [10] indican a los algoritmos de tipo *ensemble* de árboles como los de mejor precisión de detección. A su vez [9] muestra un resultado de 100% en todas sus métricas con el uso de redes bayesianas analizando, al igual que los anteriores un *dataset modbus*. En [18] se muestra mejores resultados para el algoritmo *K-Nearest Neighbors* (*KNN*) pero con un *dataset* conformado por parámetros del *WPM*. Con lo anterior es difícil hacer una comparativa de resultados dado que los *features* utilizados en la construcción de los modelos que son evaluados son distintos.

Entre otras cuestiones, se tiene que solo [9] realiza experimentos de clasificación multiclase, supervisada y no supervisada. Todos los demás trabajos llevan a cabo básicamente clasificaciones binarias. Solo [10] y [16] muestran experimentalmente el efecto de la variación del desbalance entre las clases objetivo de detección (anomalías) y la clase mayoritaria (normal).

En la segunda parte del presente trabajo se lleva a cabo un análisis experimental de un conjunto de algoritmos de clasificación y aprendizaje supervisado cuya selección ha sido inspirada en la Tabla 1. Han sido seleccionados representantes de cinco de las familias de algoritmos presentadas: *C4.5* y *REPTree*, de la FA1; *KNN* y *SVM*, de la FA2; *Multinomial Naive Bayes* (*MNB*) y *Logistic Regression* (*LR*), de la FA3; *MLP*, de la FA6; y, *RF* junto con *Gradient Boosted Tree* (*GBT*), como algoritmos de tipo *ensemble* (FA5). Se han omitido los algoritmos de reducción dimensional, cuyo uso

[3] FA: Familia de Algoritmos

podría aprovecharse mejor en clasificaciones no supervisadas, y a los modelos de máquinas de estado y autómatas determinísticos.

	[17]	[14]	[15]	[13]	[9]	[10]	[18]	[6]	[16]	[12]	[11]
A1			FA1		FA1						
A2			FA1								
A3			FA1								
A4			FA1								
A5			FA1								
A6							FA2	FA2			
A7				FA2		FA2		FA2			
A8	FA2	FA2									
A9						FA2					
A10			FA3			FA3					
A11			FA3								
A12					FA3						
A13						FA3					
A14								FA4			
A15						FA5	FA5				
A16			FA5								
A17						FA5					
A18						FA5					
A19		FA5									
A20						FA6			FA6		
A21	FA6										
A22						FA6					
A23									FA7		FA7
A24										FA7	

Tabla 2. Relación de Familias de Algoritmos e implementaciones de *IDS*

3 Evaluación Experimental Propuesta

3.1 Preparación de *datasets*

El *dataset* propuesto en [2], consta de 35 tipos diferentes de ataque pertenecientes a las 7 categorías descritas en [8]. La preparación de los datos para la evaluación experimental consiste en la creación de subconjuntos de datos (*subset*), a partir del *dataset* original, para una clasificación binaria. De esta manera se crean 7 *subsets*, cada uno conteniendo una sola categoría de ataque, sin discriminación entre escenarios individuales de ataques dentro de la misma categoría. Por tanto, en los *subsets*, cada fila está marcada con una de dos posibles etiquetas; "0" para tráfico normal, y "1" para tráfico anómalo. Adicionalmente, para evaluar los efectos del desbalance entre las clases (definido aquí como el cociente entre la cantidad de filas etiquetadas como

ataque y el número total de filas) se divide nuevamente cada *subset* de detección binaria en tres *subsets* con desbalances diferentes. El primero de ellos conserva el desbalance que se obtuvo al separar el *dataset* original en cada uno de los 7 *subsets*. Para la creación del segundo *subset*, del primero se van eliminando de forma aleatoria filas etiquetadas como tráfico normal hasta conseguir el mismo desbalance que se tiene en el *dataset* global multiclase provisto en [2], que corresponde a aproximadamente un 22%. El tercer *subset* se obtiene del primero aumentando el número de filas eliminadas hasta conseguir un desbalance del 50%. Finalmente, cada *subset* se divide en dos, con una relación de 70/30, para el entrenamiento y las pruebas respectivamente.

Entre las columnas (*features*) de los *subsets* se tienen, campos del *PDU* de *modbus*, un indicador de que el mensaje se trata de un comando o respuesta y una estampa de tiempo. La Tabla 3 muestra los mismos junto con una breve descripción. A diferencia de otros trabajos como [10], en el presente, los mensajes intercambiados en la red no son previamente agrupados en flujos. Como paso siguiente a la formación de los *subsets* se trata cada una de las columnas estableciendo para ellas tipos específicos de datos, en general, diferenciándolas entre valores numéricos y etiquetas.

Ord.	***Feature***	**Descripción**	**Ord.**	***Feature***	**Descripción**
1	*Address*	Dirección del esclavo *modbus*	10	*System Mode*	Oper. manual/ automática
2	*Function*	Código de función *modbus*	11	*Control Scheme*	Control por Bomba/ Válvula
3	*Lenght*	Longitud de mensaje	12	*Pump*	Activ. manual de bomba
4	*Setpoint*	Punto de operación de bomba	13	*Solenoid*	Activ. manual de válvula
5	*Gain*	Parámetros de ajuste del controlador PID (Proporcional Integral Derivativo)	14	*Pressure measurement*	Medición de Presión
6	*Reset rate*		15	*CRC rate*	Cod. chequeo de error *modbus*
7	*Deadband*		16	*Command/ Response*	Bandera de identificación
8	*Cylce time*		17	*TimeStamp*	Estampa de tiempo
9	*Rate*		18	*Categ. Result*	Result. de Clasif.

Tabla 3. *Features* para el entrenamiento de algoritmos evaluados

Otra cuestión importante que surge en el tratamiento de los datos para la formación de los *subsets* es el manejo de los valores desconocidos o "*missing values"*. Las tablas que forman el *dataset* tienen como columnas a cada uno de los *features* seleccionados (campos del *PDU modbus*). De esta manera, cada fila constituye un mensaje *modbusRTU*. Existen varios tipos de mensajes *modbus*, el tipo de un mensaje particular es determinado por el campo de código de función. No todos los tipos de mensajes transportan los mismos campos *modbus*, por esta razón, no todas las filas de la tabla

contendrán valores para todas las columnas (*features*). Este es el motivo por el cual se originan los "*missing values*".

Pudo observarse experimentalmente que la presencia de tales valores desconocidos empobrecía los resultados de predicción de los algoritmos. Por esta razón, se optó por una estrategia de relleno. Cabe destacar que de todos los trabajos explorados en la Sección 2, sólo [6] aborda el tema proponiendo también el relleno.

Todos los valores faltantes están relacionados con registros de memoria de esclavos *modbus*. Estos valores solo pueden cambiar en respuesta a comandos enviados desde el maestro o por decisión de una lógica de control interna del *PLC*. Los comandos donde se modifican registros del esclavo están representados por mensajes en los que se identifican campos *modbus* con los nuevos valores para tales registros. Las modificaciones que eventualmente sean realizadas por la propia lógica de control del *PLC* son finalmente reportadas en las respuestas a los mensajes de *polling* del maestro. Por lo anterior, para este trabajo, la estrategia de relleno fue la de mantener el último valor del registro del esclavo hasta leer un mensaje con un nuevo valor para el mismo.

La manipulación del *datset*, la creación de los *subsets*, la parametrización y evaluación de los algoritmos se llevan a cabo con las funcionalidades ofrecidas por la aplicación *knime*[4].

3.2 Entrenamiento y Evaluación de los Algoritmos

Para la evaluación de los algoritmos del conjunto, se crean flujos de trabajo en *knime*, en los que primeramente se normalizan los valores que luego pasan a entrenar a cada uno de los algoritmos. Como se comenta más adelante, según sea el caso, se utilizan dos tipos de normalizaciones lineales; calculadas a partir del mínimo y máximo del conjunto analizado y a partir de una distribución gaussiana con media 0 y desviación estándar 1. Al igual que con el desbalance, se llevaron a cabo pruebas cambiando la cantidad de *features*, utilizando u obviando las estampas de tiempo. La razón de esto es apreciar la dependencia de los resultados con la secuencia de ocurrencia o aparición de los mensajes en el proceso (industrial). Tanto los flujos de trabajo creados en *knime* como el *dataset* utilizado se han hecho de acceso público[5]. A continuación se presentan las parametrizaciones para cada uno de los algoritmos evaluados.

Para los algoritmos de tipo árbol de decisión, que incluyen los de tipo *ensemble*; *C4.5*, *REP* de *C4.5*, *RF*, y *GBT*; los valores de los *subsets* pasaron previamente por un proceso de normalizados lineal de tipo mínimo-máximo. Para el caso de *C4.5*, *REP*, y *RF* fueron utilizadas dos variantes de medidas de "calidad" de la información, el *Gini Index* y el *Gain Ratio* [19].

Para las variantes *MNB* y *LR* de los clasificadores estadísticos y *KNN* de aprendizaje basado en instancias, también fue utilizada normalización lineal de mínimo-máximo. Adicionalmente, para *LR* fueron ensayados dos métodos propuestos en *knime*; pequeños cuadrados con modificación iterativa de pesos (*Iteratively Reweighted Least Squares – IRLS*), y gradiente de media estocástica (*Stochastic Average Gradient –*

[4] *https://www.knime.com/*
[5] *https://github.com/AldoJavInsD/eval-IDS-ICS.git*

SAG). Para la selección de *K* en *KNN*, como en [20] se consideran valores impares (evitando empates) entre 1 y $\sqrt{N}$ (raíz cuadrada el número de filas en el *subset* de entrenamiento). Dado el desbalance en dichos *subsets* (llegando al 1% para el *subset DOS*), fueron tomados valores pequeños de *K* para evitar la predominancia de la clase mayoritaria [21]. De este modo fueron realizados cálculos para distancias a 3 y 5 vecinos. Cabe comentar que en el desarrollo de los experimentos, probando valores mayores de *K*, cercanos a $\sqrt{N}$, no se observaron grandes diferencias en los resultados de las predicciones.

En el caso del *MLP*, fue utilizada la implementación *RProp* de *knime* [22]. La normalización fue llevada a cabo con las dos variantes lineales previamente mencionadas; con valores en distribución gaussiana (*z-score*) y con valores mínimo-máximo. La incorporación del segundo tipo se debe a que en las pruebas se observaron mejoras en las métricas usando esta normalización para algunos casos. De igual forma se llega a una configuración de 5 capas, 17 neuronas por capa y 750 iteraciones, posterior a ensayos en los cuales se observaron los efectos del cambio de cada uno de los valores, seleccionando finalmente los que ofrecían mejor compromiso entre el tiempo de generación del modelo (usando el *subset* de entrenamiento) y los resultados obtenidos al evaluar el *subset* de pruebas.

Finalmente para el algoritmo *SVM*, fueron también utilizadas las dos formas de normalización, y además se exploraron las tres opciones de *kernel* que ofrece *knime*: *Polynomial*, *Radial Basis Function* (*RBF*) e *HyperTangent*.

	C4.5	REP	RF	GBT	KNN	SVM	MNB	LR	MLP
S1	**0,998**	0,996	0,997	0,997	0,985	0,975	0,994	0,997	0,994
S2	0,903	0,902	0,937	**0,977**	0,898	0,906	0,905	0,900	0,905
S3	0,916	0,916	0,938	**0,976**	0,897	0,906	0,902	0,901	0,904
S4	0,819	0,818	0,987	**0,988**	0,975	0,972	0,814	0,783	0,928
S5	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
S6	0,902	0,904	0,988	**0,993**	0,951	0,844	0,734	0,789	0,934
S7	0,835	0,839	**0,978**	0,963	0,949	0,823	0,828	0,807	0,940

Tabla 4. Mejores valores de *F1-Score* para los algoritmos evaluados con los 7 *subsets*

Los resultados obtenidos en los ensayos para todo el conjunto de algoritmos y *subsets* pueden verse en la Tabla 4, en la que por comodidad, se han nominado a los *subsets* como S1 (*RECON*), S2 (*NMRI*), S3 (*CMRI*), S4 (*MSCI*), S5 (*MFCI*), S6 (*MPCI*) y S7 *(DOS*), respectivamente.

3.3 Principales Hallazgos (PH)

- *PH1*: En el conjunto de algoritmos relacionados con árboles de decisión, como se esperaba, los mejores rendimientos fueron para los de tipo *ensemble* (*RF* y *GBT*), prácticamente para todos los tipos de ataques. En general para todos estos algoritmos los resultados se mantuvieron por encima del 93,7%, dándose los resultados más bajos para los ataques de tipo *RI* (inyección de respuestas

maliciosas), y los mejores para el *MFCI*, con el cual la mayoría de los algoritmos presentó buenos resultados. Lo anterior condice con la forma de ejecución del ataque, los códigos de función utilizados en los mensajes malicioso no son encontrados en comandos normales. Fuera del caso del *MFCI*, el mejor valor obtenido fue 99,74% para *GBT* (con un límite de 4 niveles para los árboles componentes) con *RECON* (con un desbalance del 50%, sin el uso de estampas de tiempo) y el peor 81,77% para *REP* (utilizando *Gini Index*) en *MSCI* (50% de desbalance, sin el uso de estampas de tiempo).

- *PH2*: Entre los algoritmos de clasificación probabilísticos, *MNB* y *LR*, el primero presentó los mejores resultados. Para este tipo de algoritmos el ataque más difícil de reconocer fue el de inyección de comando *MPCI* y al igual que los anteriores, el más fácil de detectar fue el de *MFCI*. Obviando este último, el mejor valor obtenido fue 99,70% para *LR* usando *SAG* con *RECON* (2% de desbalance, sin estampas de tiempo) y el peor 73,44% para *MNB* en *MPCI* (50% de desbalance, sin estampas de tiempo).
- *PH3*: Para el caso de los algoritmos basados en redes neuronales artificiales, no se aprecia una diferencia considerable entre el uso de normalización mínimo-máximo y con valores de distribución de probabilidad. El mejor valor obtenido, fuera del *MFCI*, fue 99,43% para normalización mínimo-máximo con *RECON* (desbalance del 2%, con estampas de tiempo) y el peor 89,06% para el mismo tipo de normalización en *MSCI* (50% de desbalance, con estampas de tiempo).
- *PH4*: Para los clasificadores de aprendizaje basado en instancias o memoria, *KNN* y *SVM*, el valor más alto de la métrica fue obtenido por *KNN* (3 vecinos) para el conjunto *RECON* (desbalance del 50%, sin el uso de estampas de tiempo) con 98,46%. A su vez, el menor valor fue para *SVM* (utilizando *RBF*) también con el conjunto *RECON* (50% de desbalance con estampas de tiempo) con 61,47%. Entre las variantes de *kernel* para *SVM*, el de tipo polinomial fue el que en general mejor desempeño mostró. La normalización con valores gaussianos no presentó mejoras respecto de la normalización de valores mínimo-máximo.
- *PH5*: La mayor parte de los mejores valores obtenidos en la métrica de clasificación pertenecen a situaciones en las que fue utilizado el conjunto desbalanceado a un 50% con un 82% de los casos. Cabe destacar que 65% de los mejores valores obtenidos en la métrica fueron para los casos en los que la estampa de tiempo fue utilizada en el conjunto de entrenamiento, siendo los probabilísticos, y en particular *LR* el menos afectado por este *feature*.
- *PH6*: Con respecto al tiempo de entrenamiento de los algoritmos, el *SVM* seguido del *MLP*, son los de mayor tiempo de entrenamiento.
- *PH7*: El algoritmo menos afectado por el desbalance fue el *KNN* para el cual en 3 de las evaluaciones de los 7 *subsets* de ataque, los mejores resultados no fueron para el conjunto balanceado al 50%.
- *PH8*: *GBT* obtuvo los mejores resultados de clasificación para 5 de los 7 *subsets*, los cuales corresponden a ataques de inyección de respuestas maliciosas, simple *NMRI* (S2) y compleja *CMRI* (S3), así como para todos los subtipos de inyección

de comandos, alteración de estado *MSCI* (S4), código de función malicioso *MFCI* (S5), y alteración de parámetros *MPCI* (S6). Si bien en los demás casos dicho algoritmo no obtuvo los valores más altos, los obtenidos fueron bastante cercanos a los mejores.

- *PH9*: Para el ataque de reconocimiento (*RECON*) el algoritmo de árbol simple C4.5 obtiene el mayor valor de *F1-Score* con un 99,8% seguido por *GBT* y *RF* con un 99,7%.
- *PH10*: La mejor clasificación para el *subset* con denegación de servicio (*DOS*) fue para el *RF* con un 97,8% también seguido como segundo mejor valor por el *GBT* con un 96,3%.

4 Conclusiones

En cuanto al análisis de trabajos relacionados, fueron revisados 4 trabajos que proponen *datasets* de evaluación para evaluación de *IDS* de entre los cuales uno fue seleccionado para la evaluación experimental. Así también fueron revisados más de 11 trabajos de implementación de *IDS* para *ICS*, los cuales fueron analizados y comparados aspectos como algoritmos, *datasets* de evaluación (tipos ataques y desbalance de clases), y métricas. De entre los algoritmos utilizados en estos trabajos, un conjunto de 9 algoritmos de 5 familias distintas fue seleccionado para la evaluación experimental.

La evaluación experimental arrojó como mejores resultados, para la métrica *F1-Score*, a los presentados en la Tabla 4. Analizando tales resultados fueron presentados los principales hallazgos en la parte final de la Sección 3. Como resumen de dicho análisis puede concluirse que se encontró que *MFCI*, seguido de *RECON*, son los dos tipos de ataques con mayor facilidad de detección. Así también, los algoritmos basados en árboles de decisión son los que en un mayor número de ocasiones superaron el 99% de *F1-Score* constituyéndose así en los de mejor desempeño de clasificación. En particular, dicho valor fue superado en un 32% de todas las evaluaciones. Los algoritmos probabilísticos y de redes neuronales superaron este umbral en un 29% de sus evaluaciones, y por último los basados en instancias solo superaron este porcentaje en la evaluación del conjunto *MFCI*.

Estas evaluaciones fueron llevadas a cabo con variantes no complejas de los algoritmos de base tomando ventaja de las funcionalidades ya implementadas en la herramienta *knime*. Sin embargo, se considera que los resultados obtenidos permiten apreciar suficientemente el rendimiento de los algoritmos, motivando así a la experimentación de mejoras que optimicen su desempeño con base en alguna dimensión en particular, como la cantidad de falsos positivos y el tiempo de creación de los modelos.

Queda como propuesta para trabajos futuros, la experimentación con protocolos industriales que representen sistemas de otra naturaleza como los especificados en los estándares IEC 62870-5-104 e IEC 61850, ampliamente utilizados en sistemas de generación y subestaciones eléctricas, así como otras variantes de los diferentes algoritmos de aprendizaje de máquina (*ML*).

5 Referencias

[1] Industrial Control Systems Cyber Emergency Response Team, "Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies," 2016.

[2] I. Turnipseed, Z. Thornton and T. Morris, "Industrial Control System Simulation and Data Logging for Intrusion Detection System Research," in *7th Annual Southeastern Cyber Security Summit*, Huntsvile, AL, 2015.

[3] P. Nevavuori and T. Kokkonen, "Requirements for Training and Evaluation Dataset of Network and Host Intrusion Detection System," in *New Knowledge in Information Systems and Technologies*, Springer, 2019, pp. 534-546.

[4] A. Lemay and J. Fernandez, "Providing SCADA Network Data Sets for Intrusion Detection Research," in *9th USENIX Workshop on Cyber Security Experimentation and Test*, Austin, TX, 2016.

[5] I. Frazão, P. Abreu, T. Cruz, H. Araújo and P. Simões, "Denial of Service Attacks: Detecting the Frailties of Machine Learning Algorithms in the Classification Process," in *13th International Conference, CRITIS*, Kaunas, Lithuania, 2018.

[6] M. Egger, E. Gunther and E. Dominik, "Comparison of approaches for intrusion detection in substations using the IEC 60870-5-104 protocol," in *9th DACH+ Conference on Energy Informatics ONLINE*, Sierre, Switzerland, 2020.

[7] P. Maynard, K. McLaughlin and S. Sezer, "An Open Framework for Deploying Experimental SCADA Testbed," in *In 5th International Symposium for ICS & SCADA Cyber Security Research*, University of Hamburg, Germany, 2018.

[8] T. Morris and W. Gao , "Industrial Control System Traffic Data Sets for Intrusion Detection Research," in *Critical Infrastructure Protection VIII*, vol. 441, J. Butts and S. Shenoi, Eds., Berlin, Heidelberg: Springer, 2014, pp. 65-78.

[9] I. Ullah and Q. H. Mahmoud, "A Hybrid Model for Anomaly-based Intrusion Detection in SCADA Networks," in *IEEE International Conference on Big Data (BIGDATA)*, Boston, MSA, USA, 2017.

[10] G. Vasquez, R. S. Miani and B. B. Zarpelao, "Flow-Based Intrusion Detection for SCADA networks using Supervised Learning," in *XVII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg*, Brasília, DF, Brasil, 2017.

[11] C. Sheng, Y. Yao, Q. Fu and W. Yang, "A cyber-physical model for SCADA system and its intrusion detection," *Computer Networks,* vol. 185, 2021.

[12] P. Matoušek, O. Ryšavý, M. Grégr and V. Havlena, "Flow based monitoring of ICS communication in the smart grid," *Journal of Information Security and Applications,* vol. 54, 2020.

[13] A. Terai, S. Abe, S. Kojima and Y. Takano, "Cyber-Attack Detection for Industrial Control System Monitoring with Support Vector Machine based on Communication Profile," in *IEEE European Symposium on Security and Privacy Workshops*, Paris, France, 2017.

[14] L. Maglaras, J. Jiang and T. J. Cruz, "Combining ensemble methods and social network metrics for improving accuracy of OCSVM on intrusion detection in SCADA systems," *Journal of Information Security and Applications,* no. 30, p. 15–26, 01 10 2016.

[15] S. Ponomarev and T. Atkison, "Industrial Control System Network Intrusion Detection by Telemetry Analysis," *IEEE Transactions on Dependable and Secure Computing,* vol. 13, no. 2, pp. 252-260, 2016.

[16] P. Kreimel, O. Eigner, F. Mercaldo, A. Santone and P. Tavolato, "Anomaly detection in substation networks," *Journal of Information Security and Applications,* vol. 54, 2020.

[17] P. Zeng, W. Shang, M. Wan, L. Li and P. An, "Intrusion detection algorithm based on OCSVM in industrial control system," *SECURITY AND COMMUNICATION NETWORKS,* vol. 9, no. 10, p. 1040–1049, 2015.

[18] F. Zhang, H. A. Dias Edirisinghe Kodituwakku, J. W. Hines and J. Coble, "Multilayer Data-Driven Cyber-Attack Detection System for Industrial Control Systems Based on Network, System, and Process Data," *IEEE Transactions on Industrial Informatics,* pp. 4362 - 4369, 07 01 2019.

[19] S. Tangirala, "Evaluating the Impact of GINI Index and Information," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 2, pp. 612-619, 2020.

[20] A. Hassanat, M. Ali Abbadi, G. A. Altarawneh and A. A. Alhasanat, "Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach," *International Journal of Computer Science and Information Security,* vol. 12, no. 8, pp. 33-39, 2014.

[21] I. Chelliah, "An Introduction to K-Nearest Neighbors Algorithm," Medium, 23 November 2020. [Online]. Available: https://towardsdatascience.com/an-introduction-to-k-nearest-neighbours-algorithm-3ddc99883acd. [Accessed 28 07 2022].

[22] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks (ICNN)*, San Francisco, CA, USA, 1993.