# Malicious Event Detection

Rodrigo S. Agredo rsierraa@unal.edu.co

Elkin Ramirez N. elramirez@unal.edu.co

David Martinez T. dasmartineztri@unal.edu.co

Facultad de Ciencias

Departamento de Matemáticas Universidad Nacional de Colombia. Noviembre de 2023

Bogotá. Colombia.
Repositorio:

https://github.com/rsierraa/Machine-Learning-for-Cybersecurity

Abstract—Effective detection of malicious events in computing environments is an ongoing challenge in cybersecurity. This paper presents an innovative approach that combines the use of decision trees and context-based detection to improve the accuracy and efficiency of malicious event identification. The methodology focuses on the integration of relevant contextual information to strengthen the model's decision-making capabilities.

Palabras claves—Adware, Bots, Bugs, Ransomware, Rootkit, Spyware, Trojan horses, Viruses, Worms, Decision tree, Random forest.

## I. Introducción: Ciberseguridad, Datos y Aprendizaje de Máquina

N el ámbito de la ciberseguridad,la amenaza constante y evolutiva del malware ha generado la necesidad crítica de desarrollar enfoques avanzados para la detección y mitigación de eventos maliciosos. Este capítulo se sumerge en la exploración de estrategias innovadoras que combinan dos componentes clave: el poder de los árboles de decisión y la sensibilidad contextual para mejorar la capacidad de discernir entre actividades benignas y maliciosas.

El paisaje actual de ciberseguridad se enfrenta a diversas formas de malware, cada una con su conjunto único de actividades dañinas, desde adware y bots hasta ransomware y rootkits. Además, las inyecciones maliciosas, como las SQL injections y manipulaciones en redes inalámbricas, han ampliado la superficie de ataque. En este contexto, nuestro enfoque busca no solo identificar las amenazas conocidas, sino también adaptarse a nuevas variantes y técnicas evasivas.

En este trabajo, se implementa el capítulo 7 del libro Hands-On Machine Learning for Cybersecurity[1]. A lo largo de este capítulo, se explora un caso de uso práctico utilizando el conjunto de datos KDD Cup 1999, con el objetivo de distinguir entre conexiones de red benignas y maliciosas. Esta aplicación concreta nos permitirá evaluar la eficacia de la aplicación de árboles de decisión y la consideración del contexto en la detección de eventos maliciosos.

En última instancia, este capítulo pretende no solo ofrecer un análisis detallado de los enfoques propuestos, sino también proporcionar una perspectiva práctica y aplicada para aquellos inmersos en la seguridad cibernética y el aprendizaje automático.

#### II. MARCO TEÓRICO

### A. Machine Learning en Ciberseguridad

Hoy día los protocolos tradicionales de prevención e identificación de amenazas están siendo rápidamente reemplazados por agentes inteligentes.

Utilizados para identificar nuevo malware, ataques zeroday y amenazas de persistencia avanzada (APT), estos nuevos productos de Machine Learning a manera de soluciones tecnológicas están tomando el control, en las áreas de User Behavior, Network Traffic, Endpoint Solutions y SIEM.

### B. Los Datos en el Machine Learning

Los datos son el alimento de los sistemas de aprendizaje que, mediante técnicas para detectar patrones, conocidas como "minería de datos" y otros procesos de preparación exhaustiva, convierten registros de eventos, incidentes, configuraciones, etc, tomados durante períodos específicos en contextos específicos, en modelos que son capaces de clasificar información de manera autónoma y, en última instancia, tomar o ayudar a tomar decisiones de una manera que un conjunto de seres humanos no podría tomar al menos a una velocidad remotamente cercana.

Dentro del mundo de la Ciencia de Datos, existen distintas maneras de estandarizar las muestras; hay muestras estructuradas y no estructuradas, marcadas y no marcadas (labelled and unlabelled); esto dependiendo de la organización y la clasificación previa que se le haya dado a los datos que las componen.

## C. Fases del Aprendizaje de Máquina (Machine Learning)

- Fase de Análisis: Los datos recopilados son analizados para detectar patrones en ellos, que ayudan a definir qué características y parámetros serán relevantes para ser usados al entrenar el modelo.
- Fase de Entrenamiento: Los parámetros generados en la fase anterior se utilizan para crear el modelo de aprendizaje, que es el resultado de un proceso iterativo, donde a medida que las iteraciones avanzan, mejora la calidad de predicción.
- Fase de Pruebas: El modelo obtenido en la fase anterior son puestos a prueba; es entrenado con más datos y su

desempeño es evaluado con métricas. Se suele probar con datos que no fueron usados en la fase de entrenamientos para ver cómo se comporta.

 Fase de aplicación: El modelo ya mejorado, habiendo pasado por un control de calidad, es finalmente alimentado con datos reales del contexto real en el que se planea utilizar. Se dice que el modelo entra entonces al entorno de producción.

Las propiedades, colecciones, mapas, sustituciones y predeterminados no son definidos directamente por el estándar AIML, sino más bien hacen parte del conjunto de parámetros que personalizan el producto final.

## D. Tipos de Aprendizaje de Máquina

- Algoritmos de Aprendizaje Supervisado: El aprendizaje supervisado es aquel en el que se utiliza un conjunto de datos conocido para clasificar o predecir con datos en mano
- Algoritmos de Aprendizaje Sin Supervisión: La técnica consiste en no etiquetar los datos iniciales. Las conclusiones se dan procesando datos cuya estructura no se conoce de antemano.
- Aprendizaje por Refuerzo: es un tipo de programación dinámica en la que el software aprende de su entorno para producir una salida que maximice la recompensa.

#### E. Algoritmos en ML

- Máquinas de vectores soporte: Son algoritmos de aprendizaje supervisado que se usan en la clasificación lineal y no lineal. Estas crean un hiper plano en un espacio óptimo en espacios dimensionales altos.
- Redes bayesianas: Son modelos probabilísticos usados para realizar predicciones y ayuda en la toma de decisiones. Se basan en los principios de la teoría de la probabilidad.
- Árboles de decisión: Es una técnica de aprendizaje automático predictivo. Estos usan el análisis de decisiones e intentan predecir el valor objetivo, son implementaciones de problemas de clasificación.
- Random Forests: Son una extensión de los árboles de decisión. Para este caso, su usan en conjunto para realizar predicciones. Por ser un colectivo son más estables y fiables.
- Algoritmos genéticos: Son algoritmos meta heurísticos usados en problemas de optimización con y sin restricciones.

## F. URL Spoofing [3]

URL (Uniform Resource Locator) es el formato utilizado universalmente en la Internet para representar la dirección de un recurso que se encuentra en ella (una página HTML, un documento CSS, una imagen, etc). [2]

La alteración de URLs (o URL spoofing) es una práctica fraudulenta que consiste en alterar la apariencia de una URL original que apunta al recurso oficial que se está buscando, para redirigir de alguna manera a la víctima y hacerla acceder a algún recurso no legítimo diseñado para atacarle.

A veces es suficiente solo con dar click a una URL maliciosa para infectar un dispositivo. Otras, se trata de un ataque más sofisticado que conlleva más pasos y artefactos diseñados y puestos en marcha para obtener algún beneficio (robar una identidad, obtener credenciales, obtener información sensible, etc).

## G. Recopilación y Análisis de los Datos

#### Dataset:

Las fuentes de datos provienen principalmente del informe de detección de intrusiones de DARPA de 1998. Programa de evaluación del MIT Lincoln Labs. Este conjunto de datos contiene una variedad de eventos de red que se han simulado en el entorno de la red militar. Los datos son TCP Dump que se ha acumulado desde la red de área local de un entorno de la Fuerza Aérea. Los datos están plagados de múltiples ataques.

1100:05.727% mm ) : size19-40-36-177.ist.tr.ibs.ost.125 : dui-usr-nat-110.irstarens.oss.mm ; 365:36000 mi: 1469 mi: 3186 0
(200, Almestano 1307771 114849487) (DF)
\$500;07,00787; pag0 < dal-use-rust-120,iretarora.com.ess > slis139-92-26-177,ist.tr.ibs.ret.1221; , 1449;2897(1448) ack 342 sin 318
(10p./up.timestamp 1568/8027 1247771) (SF)
[35:08:07.70754 geo ( del-use-cust-130.jestarena.com.vev ) slis139-92-26-177.ist.tr.jbs.ret.1221; , 2997:4595(1445) ack 342 win 318 Oran.ren.linestamo: 1506/58/17 1507771 (UP)
H508cW.707822 dec0 ) dist39-00-06-77.ist.tr.im.ret.1221 ) del-use-cust-110.inetarena.com.eec; 362:36200 ack 4365 ein 31856 o loren.timestaren 1507801 158696077 UFF1
/cm./institute ion/for ion/for institute insti
1200:00,74790 neo ( dul un-out-10,interea.com.aw ) dist19-30-30-17,ist.tr.jbs.ce.1221; P 43455735(144) ack 502 sin 118
R fingerless,or (299)
13:00:09.151742 gep0 ) slig139:92-26-177.ist.tr.jbs.net.1221 ) dsl-usw-cust-110.inetarena.com.ww: F 342:34200) adx 6290 win 31856 0
/mp./isentamp 138112 [16880350 0F] 1709125 [1700] mm0 / dml-um-usut-10.isentamema.com.vev ) dis179-90-25-177.ist.tr.jms.ret.[121] , 6299529000 mc 343 von 3295
1370137.13703 880 ( 01-45-041-13),043-041-13),043-04.06,060 (01-13-46-17),141,0 (18-24-123) ( 02-34-200) 80 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13),043-041-130 (01-13-46-13-46-13),043-041-130 (01-13-46-13-13-46-13-13-46-13-13-13-46-13-13-46-13-13-13-13-13-13-13-13-13-13-13-13-13-
p.co.(140588 1140478 124012 (F) 115955 MBS10 sec > slat19-50-5-17.ist.tr.ib.ret.122 > dal-up-nat-110.inetarena.coou (S 9009726:9009726:0 up 312
has 340 tack of the control of the c
13:09:03.1027; ppp0 ) slip139-90-26-177.ist.tr.;he.ret.122 ) dsl-use-cust-110.iretarene.coe.eve; P 1:322(321) ack 1 vin 32120 (nos
ptimotame 125907 [1499525] (F) 1109105 [14911] mm0 (mil-um-umi-13).iestarema.com.com ) slim[F9-90-25-177.int.tr.jim.com.1223] , [1100 mm 22 vin 1286 (mm.com.
INONIO,14741 ppo < dal-use-out-127,instarera.com.com > starIFF-50-26-177,ist.tr.tim.ret.1222) , 1/100 acc 522 con 31856 Occ.n. Lacotmer 1485500 2253007 (EF)
Liebting 1980000 200000 (p) (a) (a) (a) (a) (a) (a) (a) (a) (a) (b) (a) (a) (b) (a) (a) (a) (a) (a) (a) (a) (a) (a) (a
337750 /30100 MBO 1 00 100 0000 10 100 0000 10 100 0000 10 1
[379704.507754 see0 ) slis [37970-00-177.jst.tr.jbs.net.1222 ) dsl van-oust-110.jnetsrens.com.see1 . 322132000 ack 1449 sin 31856 0
Transconstance (1485949) (253640 (0F)
13/99/05/35/760 ppe0 < dbl-uss-cust-12),iretarera.com.com.com > slip139-92-26-177,ist.tr.ikm.ret.1222; , 2997(4565(166)) ack 222 win 318
Pop./op.timestamp 114855491 12536407 (DF)
33/95/95/95/958 ppe0 > slip135/95/95/177.ist.tr.jbs.net.1222 > dal-usa-cust-110.inetarene.com.ess; , 322:322/00 ed: 4345 win 31856 /
popularentary 150793 (1465641) (BF) 150866 STORY mad ( delicarymentil) (retorne on any ) six(Th-500Ke)77 (at to the out 1500 EP 4585-570(1447) art 30 win 30
1379706/70150 880 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,08,380 ( 01/48-041/13),1846/884,080 ( 01/48-041/13
12/9705, 0/1907 rest) : 12/12/200 acc 5793 sin 11856 0

Fig. 1: [1]

Este conjunto de datos representa en su mayoría estas 4 categorías de ataques a red comúnmente conocidos:

- Ataques de denegación de servicio (DOS): una forma más avanzada de este ataque se llama ataque de denegación de servicio distribuido (DDoS).
- Ataques de fuerza bruta: son accesos no autorizados desde una máquina remota.
- Ataques Buffer overflow: son ataques a memoria que dan accesos no autorizados a privilegios de superusuario local (root).
- Ataques de reconocimiento: se ocupan de la vigilancia de sondeo y el escaneo de puertos.

Librerías importadas : Usamos paquetes de machine learning/ciencia de datos como numpy, sklearn, pandas y matplotlib para la visualización:

```
from time import time
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import cross_val_score
```

Fig. 2: [1]

Para implementar isolation forest, utilizamos el paquete sklearn.ensemble

from sklearn.ensemble import IsolationForest  $Fig. \ 3 \colon \lceil 1 \rceil$ 

El siguiente código importa los paquetes relevantes y carga los datos (KDD):

```
from &klearn.metrics import roc_curve, auc
from &klearn.datasecs import fetch_bddcupy9
startplcnib inited
startplcnib inited
dataset = fetch_bddcup99(subset=None, shuffle=True, percentiO=True)
# http://www.ddd.org/ddd-cup/view/kdd-cup-1999/Tasks
X = dataset.data
y = dataset.target
```

Fig. 4: [1]

#### Características de los datos:

Los datos almacenados en nuestro dataset conservan las siguientes características.

Feature name	Description	Type
duration	Length (number of seconds) of the connection	continuous
protocol_type	Type of the protocol, for example, tcp, udp, and so on	discrete
service	Network service on the destination, for example, http, telnet, and so on	discrete
src_bytes	Number of data bytes from source to destination	continuou
dst_bytes	Number of data bytes from destination to source	continuou
flag	Normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	Number of wrong fragments	continuous
urgent	Number of urgent packets	continuou

Fig. 5: [1]

En el dataset se representan como.

```
feature_null = ['intraino', 'protect.htgp', 'servise', 'flap',
'mm_falled_logins', 'logo_recongingens', 'servise', 'flap',
'mm_falled_logins', 'logo_recongingens', 'rocm_fasty', 'rocm_falled_logins', 'mm_falled_logins', 'mm_falled_logins', 'mm_falled_logins', 'mm_falled_logins', 'logo_reconsise', 'rocm_falled_logins', 'logo_reconsise', 'mm_falled_logins', 'mm_falled_logins', 'mm_falled_logins', 'serv_consise', 'serv_consise_logins', 'serv_consise_logins',
```

Fig. 6: [1]

El código anterior mostrará las primeras filas de la tabla con todos los nombres de las columnas. Luego convertimos las columnas en flotantes para un procesamiento eficiente.

```
for col in X.columns:
    try:
        X[col] = X[col].astype(float)
    except ValueError:
        pass
```

Fig. 7: [1]

Convertimos lo categórico en variables ficticias o indicadores:

```
X = pd.get_dummies(X, prefix=['protocol_type_', 'service_', 'flag_']
drop_first=True)
X.head()
```

Fig. 8: [1]

Ya con todo esto es posible generar los recuentos.

Al ejecutarse, el código anterior muestra alrededor de 5 filas × 115 columnas:

Ajustamos un árbol de clasificación con max\_profundidad = 7 en todos los datos de la siguiente manera

El resultado del ajuste del modelo anterior es el siguiente: Análisis: Se implementó una técnica de minería de datos (KDD: Knowledge Discovery in Databases), en donde el objetivo fue aislar observaciones seleccionando features de la data al azar, y dividiendo en un valor entre el máximo y el mínimo. El resultado de este proceso es un árbol, donde el número de divisiones que llevan a aislar una muestra equivale a la longitud desde la raiz hasta el nodo terminal. Esta longitud, promediada en el conjunto de los random trees, es una medida de normalidad: Es nuestra función de decisión.

Se organizan los datos, dividiéndolos en train y test, y se aplican estándares como verificar la precisión de nulidad.

y.value_	counts()	
Out:		
smurf.		280790
neptune.		107201
normal.		97278
back.		2203
satan.		1589
ipsweep.		1247
portsweep	p.	1040
warezcli	ent.	1020
teardrop		979
pod.		264
nmap.		231
guess_pa	sswd.	53
buffer_o	verflow.	30
land.		21
warezmas	ter.	20
imap.		12
rootkit.		10
loadmodu		9
ftp_writ	e.	8
multihop		7
phf.		4
perl.		3
spy.		2
dtype: i	.nt64	
	Fig. 9: [1]	
from sklearn.tree im	port DecisionTreeClassifie	r, export_graphvi:
treeclf = DecisionTr	eeClassifier(may denth=7)	

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz

treecif - DecisionTreeClassifier(max_depth=7)

scores = cross_val_score(treecif, X, y, scoring='accuracy', cv=5)

print np.mean(scores)

treecif.fit(X, y)
```

Fig. 10: [1]

## 0.9955204407492013

Fig. 11: [1]

#### H. Entrenamiento

El modelo fue entrenado con una base de datos del Defense Advances Research Projects Agency - DARPA conteniendo una gran variedad de eventos de red simulados en un entorno militar. Este conjunto de datos es analizado con arboles de decisión los cuales identifican los datos recolectados en los TCP Dump entre maliciosos y no maliciosos.

Existen dos categorías de arboles, estas van en función de su objetivo

• Árbol de decisión de variable categórica: Las variables categóricas, a diferencia de las numéricas, representan atributos que no pueden expresarse en términos de cantidades o números continuos. En lugar de ello, estas variables representan distintas categorías o grupos. Al emplear un árbol de decisión para analizar conjuntos de datos con variables categóricas, cada nodo de decisión se divide con base en una categoría específica. Esta división es guiada por preguntas del tipo "¿pertenece el dato a la categoría A o a la categoría B?". La respuesta a cada una de estas preguntas es binaria, es decir, sí o no. Esta estructura de decisión, representada por los nodos del árbol, permite clasificar eficientemente los datos en

función de las distintas categorías, lo que simplifica el proceso de toma de decisiones.

• Árbol de decisión de variable continua: las variables continuas representan atributos que pueden expresarse en términos de cantidades o números continuos. Cuando nos adentramos en el análisis de árboles de decisión que involucran variables continuas, la dinámica de toma de decisiones se modifica para adaptarse a la naturaleza numérica de estas características. En lugar de simplemente dividir los nodos en categorías, las divisiones se realizan a lo largo de un rango numérico. Cada nodo ahora se pregunta: "¿El valor de la variable continua está por encima o por debajo de un determinado umbral?". La respuesta sigue siendo binaria, pero el proceso de determinar la ubicación del umbral óptimo es crucial y se basa en criterios específicos como la ganancia de información o la reducción de la impureza de Gini.

## I. Coeficiente de Gini

La reducción de la impureza de Gini es utilizada por el algoritmo de árboles de decisión para evaluar qué división resulta en una mayor homogeneidad en los nodos hijos. Una reducción mayor indica una mejor división, ya que se está logrando una mayor separación entre las clases en los nodos resultantes. En cada paso, el árbol seleccionará la división que maximice esta reducción de la impureza de Gini. Este proceso se repite recursivamente durante la construcción del árbol hasta que se cumple un criterio de parada.

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2$$

Donde:

- D es el conjunto de datos del nodo.
- c es el numero de clases en el conjunto de datos.
- $p_i$  es la proporción de elementos en el conjunto de datos que pertenecen a la clase i.

En el siguiente caso se usará un clasificador de árbol de clasificación. El objeto que se definió anteriormente recibe los

DecisionTreeClassifier(class\_weight+None, criterion='gial', max\_depth+7,
max\_feature=None, max\_leaf\_node=None,
min\_puputty\_decreas=00, min\_puputry\_pilt+None,
min\_semples\_leaf=0, min\_semples\_pilt+2,
min\_semp

Fig. 12: [1]

siguientes parámetros:

- class\_weight: Define el peso de las clases en el modelo. Si es None, todas las clases tienen el mismo peso.
- **criterion:** Especifica la función para medir la calidad de una división. En este caso, se utiliza la impureza de Gini, ya que el valor es 'gini'.
- max\_depth: Establece la profundidad máxima del árbol.
   En este caso, la profundidad máxima es 7, lo que significa que el árbol se construirá hasta una profundidad máxima de 7 niveles.

- max\_features: Indica el número de características a considerar al buscar la mejor división. Si es None, se consideran todas las características.
- max\_leaf\_nodes: Limita el número máximo de nodos hoja en el árbol. Si es None, no hay límite.
- min\_impurity\_decrease: Requiere una cierta cantidad de disminución de la impureza para dividir un nodo.
- min\_impurity\_split: Este parámetro ya no se utiliza y se mantiene por razones de compatibilidad. En versiones más recientes de scikit-learn, se ha eliminado.
- min\_samples\_leaf: Establece el número mínimo de muestras requeridas para estar en un nodo hoja.
- min\_samples\_split: Establece el número mínimo de muestras requeridas para dividir un nodo interno.
- min\_weight\_fraction\_leaf: La fracción mínima ponderada de la suma total de pesos (de todas las muestras de entrada) requerida para estar en un nodo hoja.
- **presort:** Este parámetro ya no se utiliza y se mantiene por razones de compatibilidad.
- random\_state: Si se proporciona un valor, asegura reproducibilidad.
- splitter: Especifica la estrategia utilizada para elegir la división en cada nodo. En este caso, se utiliza 'best', lo que significa que la mejor división se elige según la impureza.

Finalmente se realiza la ejecución del siguiente comando:

pd.DataFrame(('feature':X.columns,
'importance':traeelf.feature\_importances\_)).sort\_values('importance',
ascending=False).head(10)

Fig. 13: [1]

Este comando crea un DataFrame que muestra las 10 características con la mayor importancia, ordenadas de mayor a menor importancia, a partir del modelo de árbol de decisión anteriormente mencionado.

El DataFrame obtenido es el siguiente:

Número	Característica	Importancia
20	srv_count	0.633722
25	same_srv_rate	0.341769
9	num_compromised	0.013613
31	dst_host_diff_srv_rate	0.010738
1	src_bytes	0.000158
85	servicered_i	0.000000
84	serviceprivate	0.000000
83	serviceprinter	0.000000
82	servicepop_3	0.000000
75	servicenetstat	0.000000

#### J. Pruebas

Detección de Anomalías con KDD: Uamos datos binarios donde 1 representa un ataque no-normal, y verificamos la precisión.

El output de los cinco primeros elementos se ve así:

Organizamos los datos, declaramos el modelo IsolationForest y ejecutamos model fit

Hacemos una predicción:

```
Out[43]:

0 True
1 True
2 True
3 True
4 True
dtype: boo
```

Fig. 14: Resultado de imprimir el vector y

```
Out[61]:

IsolationForest(bootstrap=False, contamination=0.1, max_features=1.0, max_samples='auto', n_estimators=100, n_jobs=1, random_state=None, verbose=0)
```

Fig. 15

```
y_predicted = model.predict(X_test)
pd.Series(y_predicted).value_counts()
out[62]:
1     111221
-1     12285
dtype: int64
```

Fig. 16: [1]

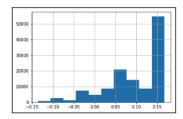


Fig. 17: Histograma de los scores[1]

## K. Aplicación y Clasificación de URLs Maliciosas Carga de datos:

```
from urlparse import urlparse
import pandas as pd
urls = pd.read_json("../data/urls.json")
print urls.shape
urls('string'] = "http://" + urls['string']
(5000, 3)
```

Fig. 18: [1]

Imprimimos las 10 primeras URLs con urls.head(10):

	pred	string	truth
0	1.574204e-05	http://startboyingstocks.com/	0
1	1.840909e-05	http://ggcvk.com/	0
2	1.842080e-05	http://432parkavenue.com/	0
3	7.954729e-07	http://gamefoliant.ru/	0
4	3.239338e-06	http://orka.cn/	0
5	3.043137e-04	Extp://medial.mercola.com/	0
6	4.107331e-37	http://ping.chartbeat.net/ping?h=sltrib.comp=	0
7	1,664497e-07	http://stephensteels.com/	0
8	1.400715e-05	NEEps//kkd-eko.pl/	0
9	2.273991e-05	http://ceakaposta.cz/	0

Fig. 19: [1]

Utilizamos un tokenizer: Una función que genere una lista de tokens a partir de las URLs para poder analizar su composición:

```
from Albarn.frature_astrection.text import CountVectorier import to def count.coemiser(string): final = [] for in integration of the final integra
```

Fig. 20: [1]

## Usamos regresión logística. Importamos los paquetes

```
from kilearn.pipeline import Ejpeline
from salearn.linear.model import Legistichegression
west = CountVectorizer(skemizer-custom.tckemizer)
ir_pipe = Pipelinen(('west', west), ('model', lr))
ir_pipe = Pipelinen(('west', west), ('model', lr))
from xilearn.model.gelection import cross_val_pore, GriddearchCV.
scores = cross_val_pore(Ir_pipe, X, y, cw=5)
scores.mean() + ont good compole.
```

Fig. 21: [1]

El puntaje obtenido (output de scores.mean()) es:

### 0.980002384002384

```
Fig. 22: [1]
```

Utilizamos random forest para esta detección de URLs maliciosas:

```
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
rf.pipe = Pipeline[[('wect', vect), ('model',
RandomForestClassifier(n_estimator=500)])
scores = cross_val_score(rf_pipe, X, y, cv=5)
scores = cross_val_score(st_pipe, X, y, cv=5)
```

Fig. 23: [1]

El puntaje obtenido es

```
0.981002585002585
```

Fig. 24: [1]

Creamos los datasets de prueba y entrenamiento, y la matriz de confusión

0.981002585002585

Fig. 25: [1]

Realizamos un análisis de sensibilidad variando el threshold

Fig. 26: [1]

Obtenemos el resultado de cada matriz de confusión para cada valor de threshold

La siguiente tabla muestra el índice de importancia de cada parámetro en la clasificación de las URLs

Usamos el clasificador con árbol de decisión así

El nivel de precisión obtenido con el árbol de clasificación es de 0.9822017858017859, mucho más alto que con el tokenizador.

El modelo que se obtiene es un árbol de decisión respecto a los parámetros estandarizados en las muestras (parámetros que son variables categóricas transformadas en números), juzgando sus valores de verdad, donde cada movimiento hacia la izquierda significa verdadero y hacia la derecha falso. [[1190 151 15 30]] [[1201 17 2811 0.3 [[1204 11 [ 22 23]] [[1205 [ 25 20]] [[1205 [ 27 18]] [[1205 01 28 17]] [[1205 0] [ 29 16]] [[1205 29 16]]

Fig. 27: [1]



Fig. 28: [1]

Named 2 - Destination exclanation (as applied)

Vertically a Papal (March 1984), vol. (1984), framed (E.)

Verti - Conditions (insettination-reason, plantation)

Verti - Conditions (insettination-reason, plantation)

Vertical (March 1984), paragraphy (E. y. servings' annuage', one

COL 60 - Memorrow (March 1984), p. 15 - March 1984, p. 1884, p

Fig. 29: [1]

The tree diagram below shows how the decision logic for maliciousness detection works.

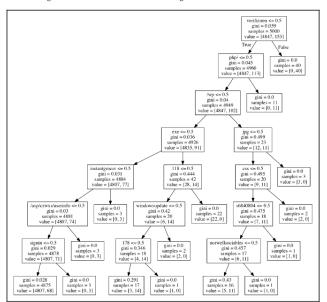


Fig. 30: Figura 1: Diagrama que muestra cómo funciona la lógica de decisión de maliciosidad de nuestro modelo clasificador [1]

#### III. ANÁLISIS Y DESARROLLO

Este trabajo consiste en la implementación de un modelo clasificador de actividad maliciosa en redes de computadores,

y de URLs, mediante el análisis, procesamiento y entrenamiento basado en muestras de tráfico TCP.

#### IV. CONCLUSIONES

Se estudiaron diferentes tipos de datos maliciosos junto con inyecciones maliciosas en sensores inalámbricos. También se abordaron los diferentes tipos de árboles de decisión, que incluyeron árboles de decisión para variables categóricas y continuas. Se concluyó con una aplicación en la detección de URLs maliciosas mediante árboles de decisión.

Lo que sigue, y está un nivel más allá del alcance de este capítulo en el libro [1] es aprender cómo se pueden atrapar a los impostores y hackers con las manos en la masa.

#### REFERENCIAS

- [1] Halder, S., & Ozdemir, S. (2018). *Hands-On Machine Learning for Cybersecurity* (1.ª ed., pp. 159–182). Birmingham: Packt Publishing.
- [2] Mozilla Foundation What is a URL? MDN Web Docs Available at: https://developer.mozilla.org/en-US/docs/Learn/
- [3] NordVPN Foundation Web DocsURL Spoofing: Definition and explanation Available at: https://nordvpn.com/es/blog/url-spoofing/