

Diseño de un modelo neuronal para la detección y la clasificación de intrusiones en redes informáticas

Hugo López Álvarez

Tutor: D. Diego García Álvarez



Universidad de Valladolid

- 1. Introducción
- 2. Enfoques
- 3. Planificación y costes
- 4. Entendimiento del problema
- 5. Entendimiento de los datos
- 6. Modelado
- 7. Evaluación
- 8. Despliegue
- 9. Conclusiones

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Contexto

- ▶ Intrusiones en sistemas informáticos.

- ▶ Evolución de los ataques a las redes informáticas como consecuencia del uso de IA.

- ▶ Defensa ante posibles intrusiones.



Objetivos del proyecto

1. Diseñar e implementar un modelo capaz de detectar intrusiones en redes informáticas y proporcionar una clasificación previa de la intrusión.
2. Desarrollar modelos de detección que han de ser modelos neuronales.
3. Evaluar y comparar los modelos generados con un *dataset* real y complejo.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Metodología

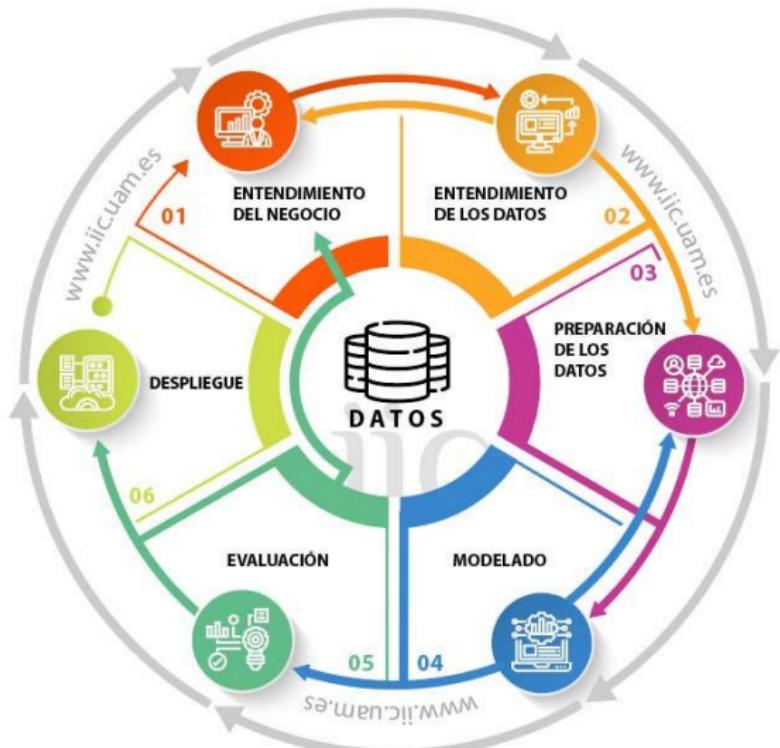


Figura: Esquema del ciclo CRISP-DM estándar.

Marco de trabajo

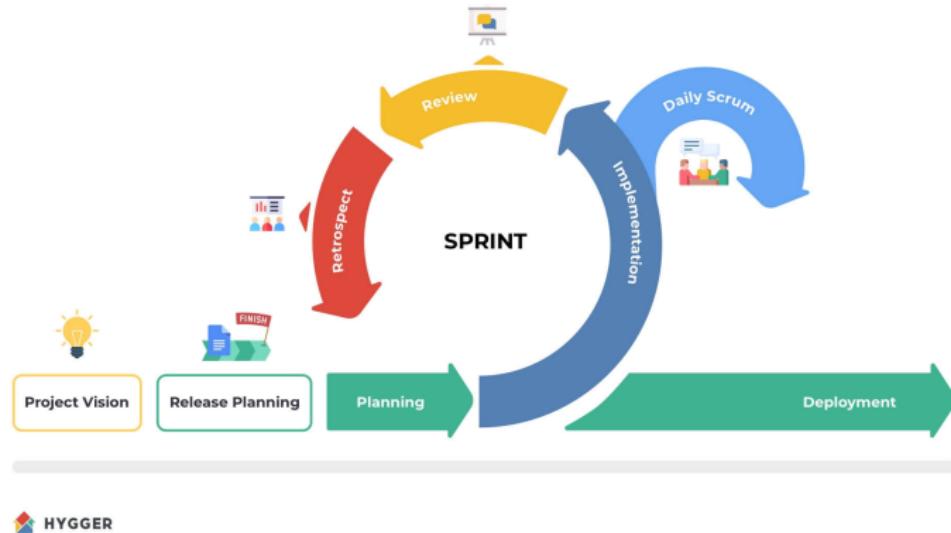


Figura: Esquema del marco de trabajo ágil SCRUM.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

Planificación

Costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Planificación del proyecto

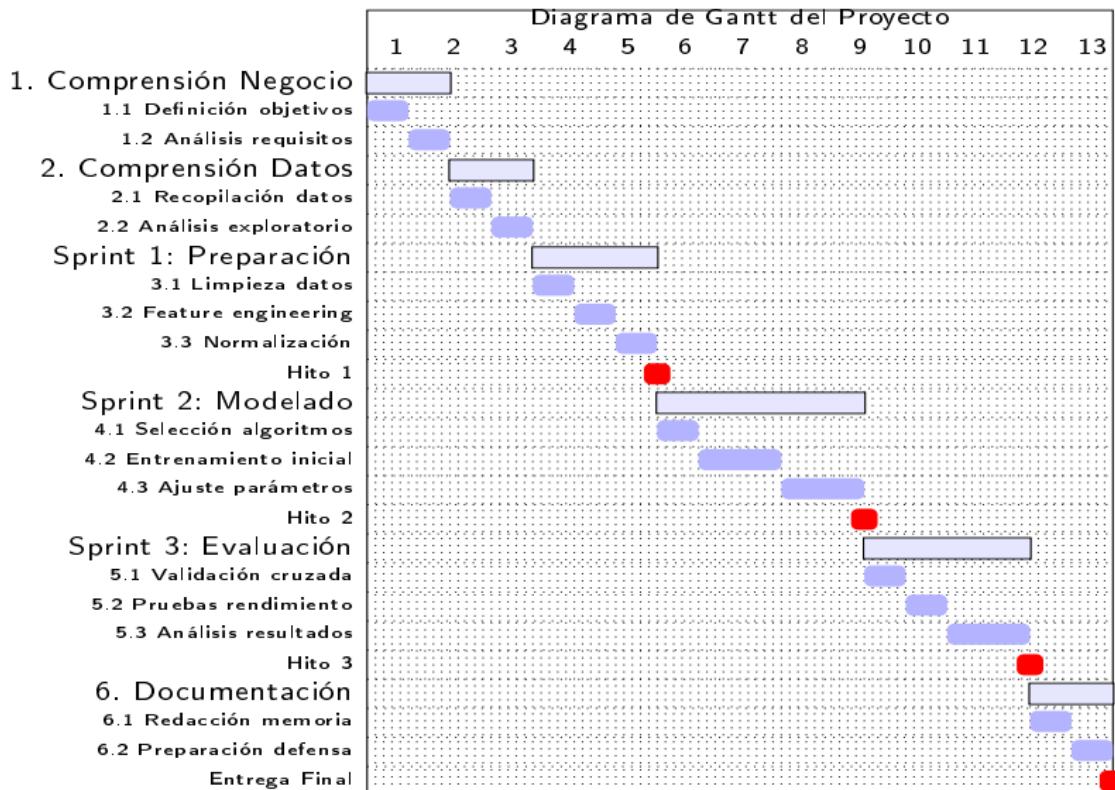


Figura: Diagrama de Gantt para la planificación del proyecto.

Costes de la puesta en marcha del proyecto

$$\text{Amortización del Hardware} = 829 \text{ €} \cdot \frac{1}{8} \cdot \frac{3,5}{12} = 30,22 \text{ €}$$

Funcionalidad	Software	Coste Mensual	Duración	Coste Total
Sistema Operativo (SO)	Kubuntu 24.10 x86_64	0 €	3,5 meses	0 €
Lenguaje (memoria)	Latex	0 €	3 meses	0 €
Editor latex	TexMaker	0 €	3 meses	0 €
IDE	MS Visual Studio Code	0 €	1 mes	0 €
Lenguaje (modelos)	Python	0 €	1 mes	0 €
IDE de Python	Jupyter Notebooks	0 €	1 mes	0 €
Plataforma MLOps	Weights&Biases	0 €	1 mes	0 €
Control de versiones	GitHub	0 €	1 mes	0 €
IA generativa (código)	DeepSeek	0 €	1 mes	0 €
IA generativa (memoria)	Gemini	0 €	2 mes	0 €
Comunicación 1	MS Outlook	0 €	3,5 mes	0 €
Comunicación 2	MS Teams	0 €	3,5 mes	0 €

Cuadro: Costes del Software.

Costes de la puesta en marcha del proyecto

Profesional	Horas Totales	€/hora	Total (€)
Ingeniero ML	137	42	5 754,0
Científico de Datos	113	34,5	3 898,5
Analista de Datos	50	18,5	925,0
Costes MOD	300		10 577,5

Cuadro: Costes de la mano de obra directa.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Conceptos básicos

- ▶ ¿Qué es un ataque a un sistema informático?
 - ▶ Tipos de ataque más comunes.
 - ▶ TCP/IP.

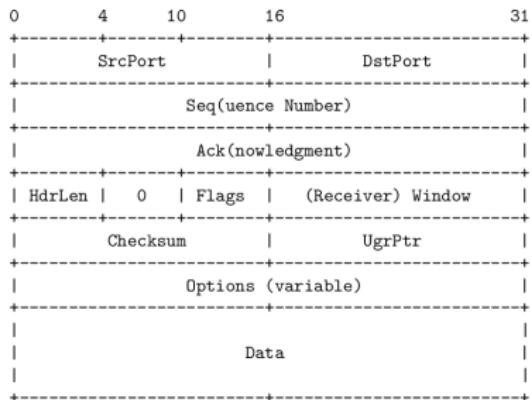


Figura: Esquema segmento TCP.

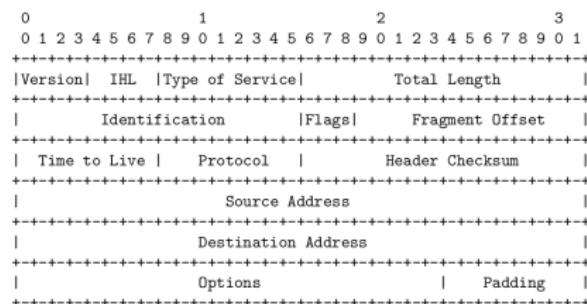


Figura: Formato de la cabecera IPv4.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Conjunto de datos: NF-UNSW-NB15-v3

Atributos (53)								Etiquetas (2)	
#	Dur	Proto	SBytes	DBytes	...	Label	Attack		
0	4,7	TCP	102	300	...	0	Benign	Conecciones	Conecciones
1	7,3	UDP	215	540	...	1	Fuzzers		
2	0,9	TCP	98	188	...	0	Benign		
3	3,6	TCP	155	222	...	1	Generic		
4	6,4	UDP	300	310	...	1	Exploits		
5	1,8	ICMP	65	120	...	0	Benign		
6	5,5	TCP	180	390	...	1	Backdoor		
7	2,9	TCP	143	305	...	1	DoS		
8	9,1	UDP	260	470	...	0	Benign		
9	1,2	TCP	97	190	...	0	Benign		
10	4,3	ICMP	120	240	...	1	Exploits		
:	:	:	:	:	...	:	:		
2 365 424	3,2	TCP	245	510	...	1	DoS		

Número de muestras por clase del conjunto de datos

#	Dur	Proto	SBytes	DBytes	...	Atributos (53)		Etiquetas (2)	
						Label	Attack		
0	4,7	TCP	102	300	...	0	Benign		
1	7,3	UDP	215	540	...	1	Fuzzers		
2	0,9	TCP	98	188	...	0	Benign		
3	3,6	TCP	155	222	...	1	Generic		
4	6,4	UDP	300	310	...	1	Exploits		
5	1,8	ICMP	65	120	...	0	Benign		
6	5,5	TCP	180	390	...	1	Backdoor		
7	2,9	TCP	143	305	...	1	DoS		
8	9,1	UDP					Benign		
9	1,2	TCP					Benign		
10	4,3	ICMP					Exploits		
:	:								
2 365 424	3,2	TCP							

Conexiones

Clase	Cantidad
Benigno	2 237 731
Fuzzers	33 816
Analysis	2 381
Backdoor	1 226
DoS	5 980
Exploits	42 748
Generic	19 651
Reconnaissance	17 074
Shellcode	4 659
Worms	158

Partición de los datos

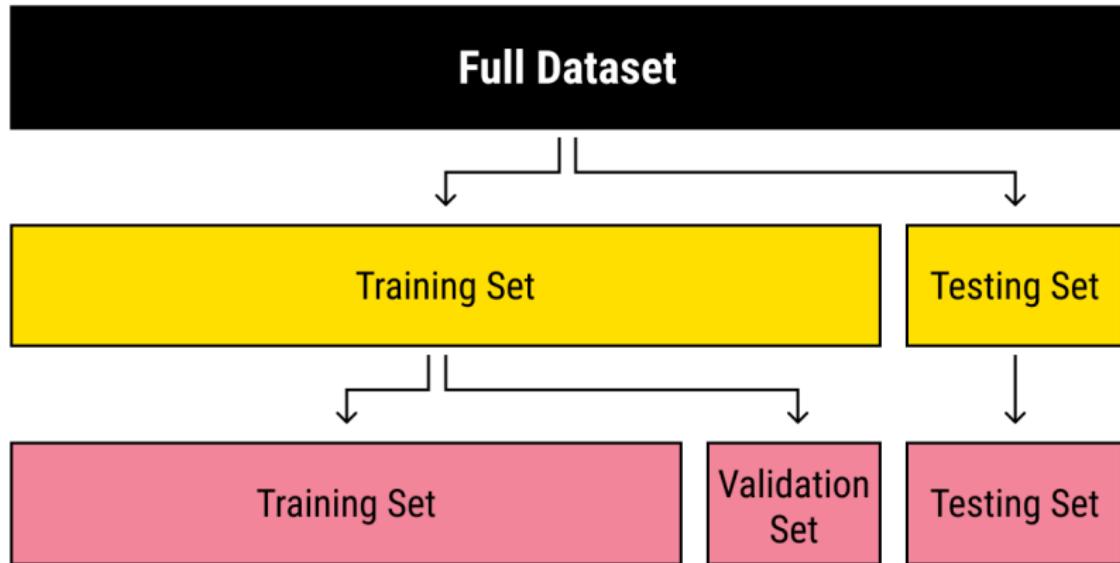


Figura: Partición de los datos para cada fase del desarrollo del modelo.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

Origen de los modelos neuronales

Clasificación con redes neuronales

Arquitecturas desarrolladas

Implementación de los modelos

Selección de las configuraciones
de los modelos

7. Evaluación

8. Despliegue

9. Conclusiones

Parámetros e hiperparámetros

Los hiperparámetros son los responsables de controlar el comportamiento del proceso de entrenamiento. Algunos de los hiperparámetros más importantes son:

- ▶ Tasa de aprendizaje (*Learning rate*).
 - ▶ Épocas (*Epochs*).
 - ▶ Tamaño de lote (*Batch size*).

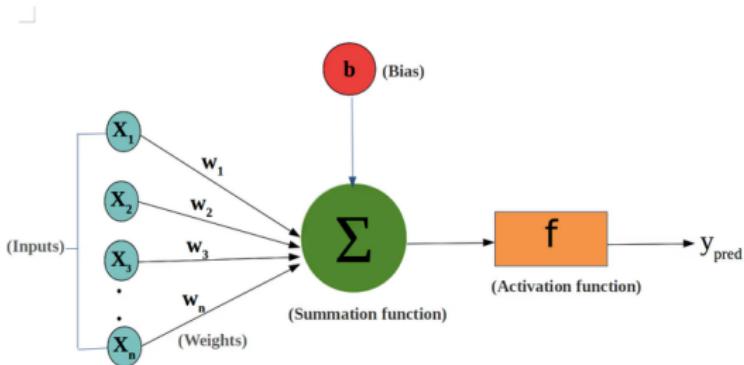


Figura: Esquema del funcionamiento de una neurona artificial.

Matrices de confusión para el MCB

- ▶ Verdaderos positivos (VP).
- ▶ Falsos positivos (FP).
- ▶ Falsos negativos (FN).
- ▶ Verdaderos negativos (VN).

	Predictión Positiva	Predictión Negativa
Real Positivo	VP	FN
Real Negativo	FP	VN

Cuadro: Matriz de confusión para modelos de clasificación binaria.

$$\text{Accuracy} = \frac{VP + VN}{VP + FP + VN + FN}$$

$$\text{Precision} = \frac{VP}{VP + FP}$$

$$\text{Recall} = \frac{VP}{VP + FN}$$

Matriz de confusión para el MCM

	Predictión Clase 1	Predictión Clase 2	Predictión Clase 3	Predictión Clase 4	Predictión Clase 5	Predictión Clase 6	Predictión Clase 7	Predictión Clase 8	Predictión Clase 9
Real Clase 1	VP ₁	FP ₁₂	FP ₁₃	FP ₁₄	FP ₁₅	FP ₁₆	FP ₁₇	FP ₁₈	FP ₁₉
Real Clase 2	FP ₂₁	VP ₂	FP ₂₃	FP ₂₄	FP ₂₅	FP ₂₆	FP ₂₇	FP ₂₈	FP ₂₉
Real Clase 3	FP ₃₁	FP ₃₂	VP ₃	FP ₃₄	FP ₃₅	FP ₃₆	FP ₃₇	FP ₃₈	FP ₃₉
Real Clase 4	FP ₄₁	FP ₄₂	FP ₄₃	VP ₄	FP ₄₅	FP ₄₆	FP ₄₇	FP ₄₈	FP ₄₉
Real Clase 5	FP ₅₁	FP ₅₂	FP ₅₃	FP ₅₄	VP ₅	FP ₅₆	FP ₅₇	FP ₅₈	FP ₅₉
Real Clase 6	FP ₆₁	FP ₆₂	FP ₆₃	FP ₆₄	FP ₆₅	VP ₆	FP ₆₇	FP ₆₈	FP ₆₉
Real Clase 7	FP ₇₁	FP ₇₂	FP ₇₃	FP ₇₄	FP ₇₅	FP ₇₆	VP ₇	FP ₇₈	FP ₇₉
Real Clase 8	FP ₈₁	FP ₈₂	FP ₈₃	FP ₈₄	FP ₈₅	FP ₈₆	FP ₈₇	VP ₈	FP ₈₉
Real Clase 9	FP ₉₁	FP ₉₂	FP ₉₃	FP ₉₄	FP ₉₅	FP ₉₆	FP ₉₇	FP ₉₈	VP ₉

Cuadro: Matriz de confusión para los modelos de clasificación multiclas.

$$F1_{\text{weighted}} = \sum_{c=1}^C \frac{N_c}{N} \cdot F1_c$$

Donde:

- ▶ C es el número total de clases.
- ▶ N_c es el número de muestras de la clase c .
- ▶ N es el número total de muestras.
- ▶ $F1_c$ es la puntuación F1 de la clase c , calculada como:

$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

Perceptrón multicapa (MLP)

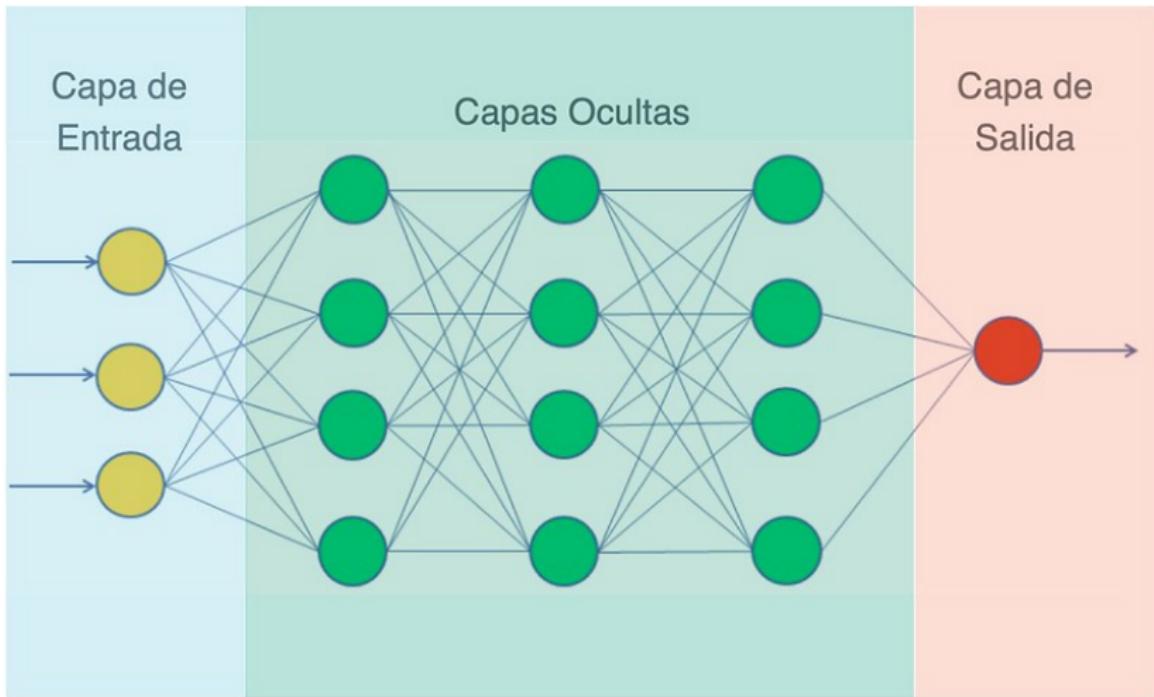


Figura: Esquema del perceptrón multicapa.

Patrón pizarra y Weights&Biases

The screenshot shows the Weights&Biases interface with the following details:

- Project:** TFG_Hugo / Projects / Alv-lop-hugo's workspace
- Table:** Runs (108)
- Filter:** Name (108 visualized)
- Columns:** batch, epoch, hidden, learning, avg_recall, avg_precision, avg_accuracy, avg_f1, avg_f2, avg_fn, avg_fp, avg_lo, avg_ro, avg_sp
- Data:** A list of 108 training runs, each with a unique color-coded icon. The first few rows are:
 - Avg_bs(20000)_lr(0.01)_hs(98)_e(10) - 20000, 10, 98, 0.01, 0.99675, 0.9975, 0.99985, 0.99812, 0.9985, 18.4, 36.8, 0.0039172, 0.99978, 0.99989
 - Avg_bs(20000)_lr(0.01)_hs(49)_e(10) - 20000, 10, 49, 0.01, 0.99867, 0.99713, 0.99983, 0.9979, 0.99836, 19.6, 42.2, 0.0040364, 0.99977, 0.99988
 - Avg_bs(10000)_lr(0.001)_hs(49)_e(30) - 10000, 30, 49, 0.001, 0.99865, 0.99716, 0.99983, 0.99791, 0.99835, 19.8, 41.8, 0.0040818, 0.99978, 0.99988

Figura: Ejemplo de utilización de Weights&Biases como pizarra.

Arquitecturas desarrolladas para el MCB

- ▶ **MCB25:** Número de neuronas en la capa oculta igual a la mitad del número de atributos o entradas que recibe el modelo.
- ▶ **MCB49:** Número de neuronas en la capa oculta igual al mismo número de atributos o entradas que recibe el modelo.
- ▶ **MCB98:** Número de neuronas en la capa oculta igual al doble del número de atributos o entradas que recibe el modelo.

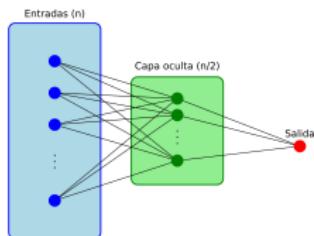


Figura: MCB25.

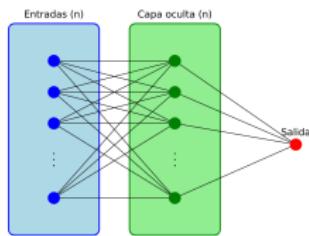


Figura: MCB49.

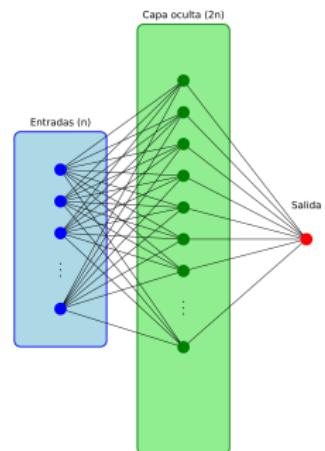


Figura: MCB98.

Arquitecturas desarrolladas para el MCM

- ▶ **MCM25:** Número de neuronas en la capa oculta igual a la mitad del número de atributos o entradas que recibe el modelo.
- ▶ **MCM49:** Número de neuronas en la capa oculta igual al mismo número de atributos o entradas que recibe el modelo.
- ▶ **MCM98:** Número de neuronas en la capa oculta igual al doble del número de atributos o entradas que recibe el modelo.

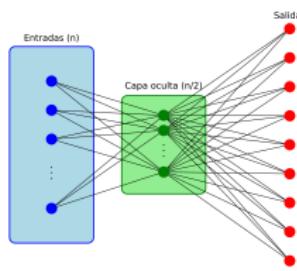


Figura: MCM25.

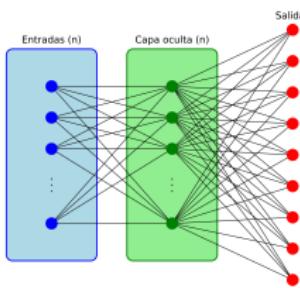


Figura: MCM49.

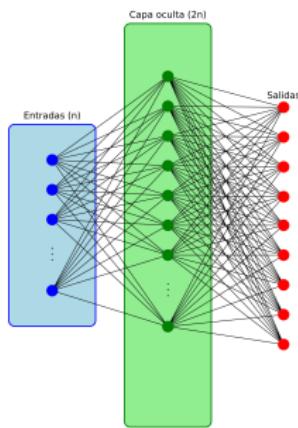
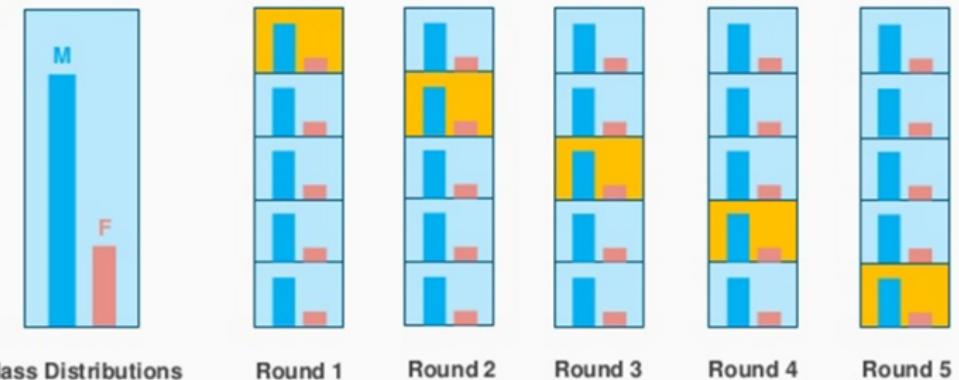


Figura: MCM98.

Validación cruzada



Keep the distribution of classes in each fold

- Training Data (light blue)
- Validation Data (yellow)

Figura: Esquema de la validación cruzada (*Stratificate k-fold (5 folds)*).

Implementación del MCB

- ▶ Función de pérdida: BCEWithLogistLoss
- ▶ Algoritmo de optimización: AdamW

Hiperparámetro	Posibles valores
<i>Batch size</i>	[2000, 10000, 15000, 20000]
<i>Learning rate</i>	$[10^{-2}, 10^{-3}, 10^{-4}]$
Épocas	[10, 20, 30]

Cuadro: Valores de los hiperparámetros utilizados en los experimentos del modelo de clasificación binaria.

Implementación del MCM

- ▶ **Función de pérdida:** CrossEntropyLoss
- ▶ **Algoritmo de optimización:** AdamW

Hiperparámetro	Posibles valores
<i>Batch size</i>	[32, 64, 128, 256, 512]
<i>Learning rate</i>	[10^{-2} , 10^{-3} , 10^{-4} , 10^{-5}]
Épocas	[30, 50, 80, 100]

Cuadro: Valores de los hiperparámetros utilizados en los experimentos del modelo de clasificación multiclas.

Selección de los mejores MCB

Los mejores resultados del MCB los obtuvo la arquitectura con el doble de neuronas en su capa oculta que atributos de entrada tiene el modelo.

Posicion_EXP	1º-MCB98	2º-MCB98	3º-MCB98	4º-MCB98	5º-MCB98
batch_size	20000	10000	15000	15000	10000
epoches	10	30	30	10	20
learning_rate	10 ⁻²	10 ⁻³	10 ⁻³	10 ⁻²	10 ⁻³
avg_f1	0.998124	0.997941	0.997771	0.998015	0.997730
avg_fn	18.4	21.6	22	22.4	22.6
avg_fp	36.8	39	43.6	36	44.2
avg_precision	0.997501	0.997351	0.997039	0.997554	0.996999
avg_recall	0.998749	0.998531	0.998504	0.998477	0.998463
avg_roc_auc	0.999781	0.999777	0.999776	0.999773	0.999777
avg_tn	344127.4	344125.2	344120.6	344128.2	344120
avg_tp	14686.2	14683	14682.6	14682.2	14682

Cuadro: Mejores cinco configuraciones para el MCB98.

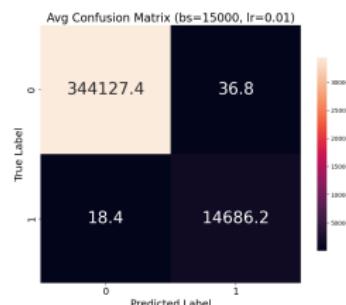


Figura: Matriz de confusión 1º MCB98.

Mejores 5 configuraciones del MCB

En este modelo no se observa que la arquitectura influya especialmente en los resultados obtenidos.

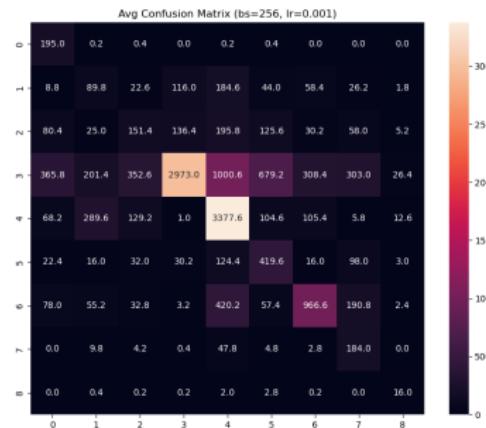
batch_size	epochs	hidden_size	learning_rate	avg_recall
20000	10	98	0.01	0.9987486972
20000	10	49	0.01	0.9986670886
10000	30	49	0.001	0.9986534868
15000	10	25	0.01	0.9986126862
15000	10	49	0.01	0.9985718792

Figura: Mejores cinco configuraciones de hiperparámetros del modelo de clasificación binaria.

Selección de los mejores MCM

Al igual que en el MCB, los mejores resultados del MCM se han obtenido en la arquitectura con el doble de neuronas en su capa oculta que atributos de entrada tiene el modelo.

Posicion_EXP	1º-MCM98	2º-MCM98	3º-MCM98	4º-M CM 98	5º-M CM 98
batch_size	256	256	512	64	256
epochs	100	80	100	80	50
learning_rate	10 ⁻³	10 ⁻³	10 ⁻²	10 ⁻³	10 ⁻³
avg_accuracy	0.569414	0.556057	0.556915	0.556969	0.562083
avg_f1_macro	0.413180	0.406773	0.388808	0.398720	0.397308
avg_f1_weighted	0.583123	0.577553	0.574200	0.571020	0.569831
avg_precision_macro	0.394898	0.385322	0.372836	0.377543	0.387029
avg_precision_weighted	0.681971	0.675326	0.674534	0.669836	0.669688
avg_recall_macro	0.577069	0.564962	0.573083	0.566046	0.550391
avg_recall_weighted	0.569414	0.556057	0.556915	0.556969	0.562083
avg_roc_auc_ovo	0.813320	0.806782	0.809286	0.812789	0.800316
avg_roc_auc_ovr	0.788041	0.784984	0.781135	0.782758	0.777422



Cuadro: Mejores cinco configuraciones para el MCM98.

Figura: Matriz de confusión 1º MCM98.

Mejores 10 configuraciones del MCM

En el caso del MCM, la arquitectura MCM98 ha obtenido en la fase de entrenamiento unos resultados muy superiores a las otras dos arquitecturas desarrolladas.

batch_size	epochs	hidden_size	learning_rate	avg_f1_weighted
256	100	98	0.001	0.5831225815
256	80	98	0.001	0.5775525405
512	100	98	0.01	0.5741997027
64	80	98	0.001	0.5710195986
256	50	98	0.001	0.5698308505
256	50	49	0.001	0.5698160035
128	80	98	0.001	0.5669125716
128	100	98	0.001	0.5657792633
128	100	49	0.001	0.5655814612
128	80	49	0.001	0.5654025284

Figura: Mejores diez configuraciones de hiperparámetros del modelo de clasificación multiclas.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Análisis de los resultados obtenidos en el MCB

- ▶ Los resultados obtenidos durante la fase de evaluación para las arquitecturas del MCB han sido mejores que los obtenidos durante la fase de entrenamiento.
- ▶ Destaca la ausencia de la arquitectura MCB49 entre las cinco configuraciones con mejores resultados del MCB. Este hecho puede atribuirse a factores aleatorios no controlables.

Posicion_EXP	batch_size	learning_rate	epochs	test_recall
3º-MCB98	15000	0.001	30	0.9987487079
3º-MCB25	20000	0.01	10	0.9987487079
5º-MCB98	10000	0.001	20	0.9987487079
1º-MCB98	20000	0.01	10	0.9986943039
2º-MCB25	10000	0.001	30	0.9986398999

Figura: Mejores cinco modelos en la fase de evaluación del modelo de clasificación binaria.

Análisis de los resultados obtenidos en el MCM

- ▶ Los resultados obtenidos durante la fase de evaluación para las arquitecturas del MCM han sido ligeramente peores que los obtenidos durante la fase de entrenamiento.
- ▶ En la fase de evaluación la arquitectura MCB25 ha obtenido un resultado superior respecto al resto de las arquitecturas en comparación con los resultados de la fase de entrenamiento.

Posicion_EXP	batch_size	epochs	learning_rate	test_f1_weighted
4º-MCM98	64	80	0.001	0.5685004671
3º-MCM25	128	80	0.001	0.5384648868
2º-MCM25	512	100	0.001	0.5297063107
5º-MCM49	256	80	0.001	0.5166473814
2º-MCM98	256	80	0.001	0.5090883811

Figura: Mejores cinco modelos en la fase de evaluación del modelo de clasificación multiclas.

Matriz de confusión del MCM98 normalizada

Dado el desbalanceo existente entre el número de muestras de cada clase del conjunto de datos utilizado, es recomendable aplicar la técnica de normalización de la matriz de confusión.

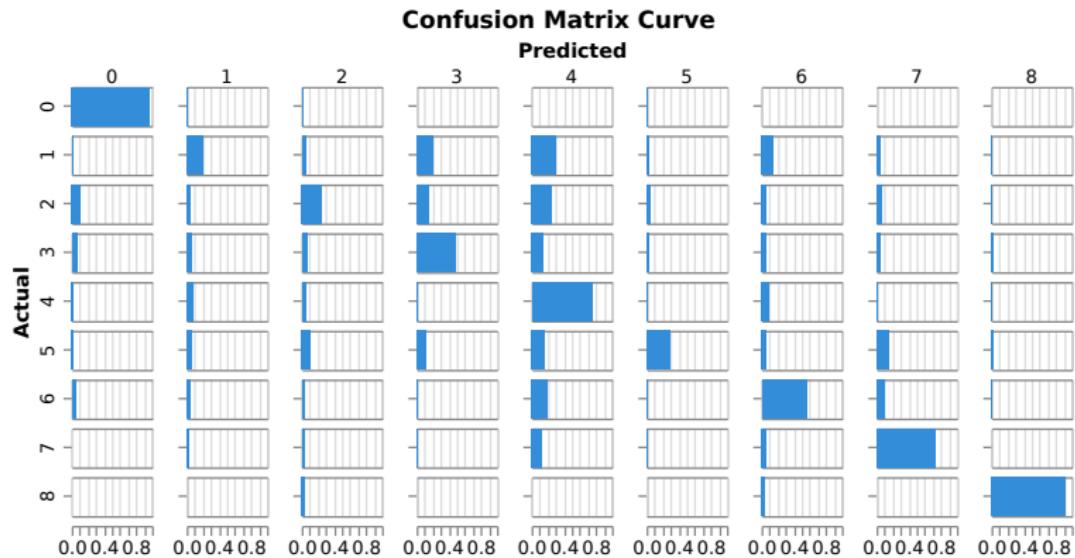


Figura: Matriz de confusión normalizada de la mejor configuración de hiperparámetros obtenida en el MCM98 durante la fase de evaluación.

Comparación de los resultados con el trabajo de M. Sarhan

Los modelos desarrollados en este trabajo presentan mejoras notables en la clasificación binaria y un rendimiento competitivo en la clasificación multiclase frente a otros modelos publicados anteriormente.

Modelo	Accuracy	AUC	F1-Score	Recall	Precision	Specificity
Sarhan et al.	98.62%	0.9485	0.85	-	-	-
3º-MCB98	99.98%	0.9998	0.9979	0.9987	0.9971	0.9999

Cuadro: Comparación de modelos de clasificación binaria.

Clase	F1-Score (Sarhan et al.)	F1-Score (4º MCM98)
Analysis (0)	0.15	0.97
Backdoor (1)	0.17	0.20
DoS (2)	0.41	0.25
Exploits (3)	0.82	0.48
Fuzzers (4)	0.55	0.75
Generic (5)	0.66	0.28
Reconnaissance (6)	0.82	0.56
Shellcode (7)	0.75	0.72
Worms (8)	0.55	0.92

Cuadro: Comparación de los valores obtenidos en F1-Score por clase.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Despliegue de los modelos en un entorno real

El despliegue efectivo de estos modelos contribuye a mejorar la seguridad en redes mediante la automatización de la detección de conexiones malignas, permitiendo respuestas rápidas y reducir el número de falsos positivos y negativos.

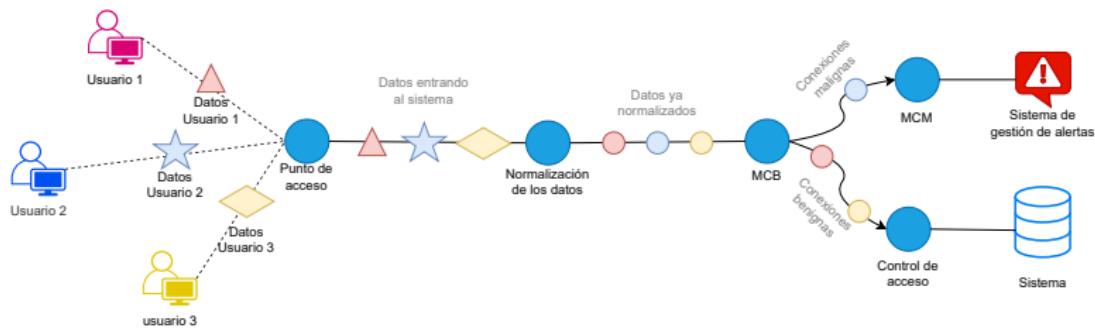


Figura: Ejemplo de despliegue de los modelos en un sistema informático.

Índice

1. Introducción

2. Enfoques

3. Planificación y costes

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Conclusiones del proyecto

- ▶ Se han desarrollado un modelo de clasificación binaria (MCB) y un modelo de clasificación multiclas (MCM) basados en una red neuronal de tipo MLP (Perceptrón multicapa).
- ▶ El MCB y el MCM son capaces de detectar con gran precisión si una conexión a un sistema es benigna o maligna y de dar un clasificación previa del tipo de intrusión que se está produciendo.
- ▶ El entrenamiento y la evaluación de los modelos se ha llevado a cabo utilizando datos que simulan un escenario real, lo que da lugar a que el MCB y el MCM estén preparados para funcionar en un entorno real.

Muchas gracias por su atención.



Universidad de Valladolid