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. Planificación y costes
3. Metodología
4. Entendimiento del problema
5. Entendimiento de los datos
6. Modelado
7. Evaluación
8. Despliegue
9. Conclusiones

Índice

1. Introducción
2. Planificación y costes
3. Metodología
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

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

Objetivos del académicos

- Comprender el funcionamiento de los modelos neuronales a través de PyTorch y las métricas de evaluación.
- Asimilar las características de varios tipos de modelos neuronales existentes.
- Descubrir el potencial de las redes neuronales para optimizar y mejorar las tecnologías de la información, incluyendo la ciberseguridad de los sistemas.

Índice

1. Introducción

2. Planificación y costes
   - Planificación
   - Costes

3. Metodología

4. Entendimiento del problema

5. Entendimiento de los datos

6. Modelado

7. Evaluación

8. Despliegue

9. Conclusiones

Planificación del proyecto

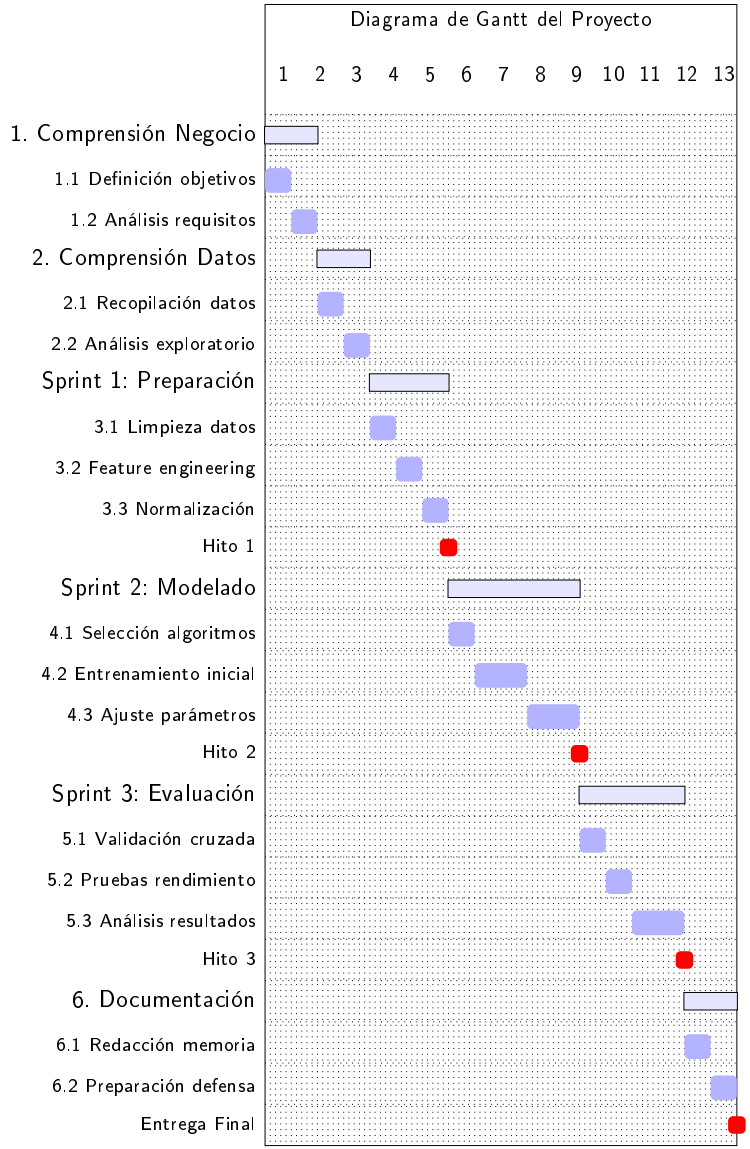

Figura: Diagrama de Gantt para la planificación del proyecto.

Costes de la puesta en marcha del proyecto

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

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	Jupyeter 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 de Software.

Profesional	Coste Total (€)
Ingeniero ML	5 754,0
Científico Datos	3 898,5
Analista Datos	925,0
Coste Total MOD	**10 577,5**

Cuadro: Coste Total de la mano de obra directa..

Índice

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

CRISP-DM

- Diseñada para guiar proyectos de minería de datos y aprendizaje automático.
- Su estructura cíclica y flexible, la hace aplicable en diversos dominios, desde el marketing hasta la ciberseguridad.

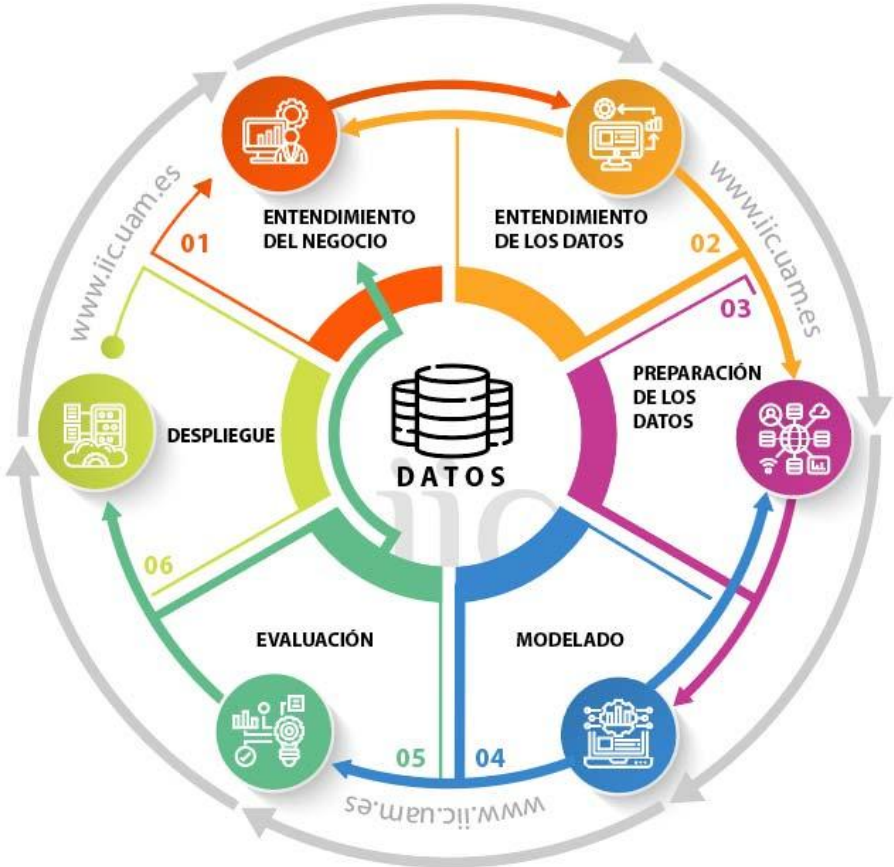

Figura: Esquema del ciclo CRISP-DM estándar.

Índice

1. Introducción
2. Planificación y costes
3. Metodología
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.

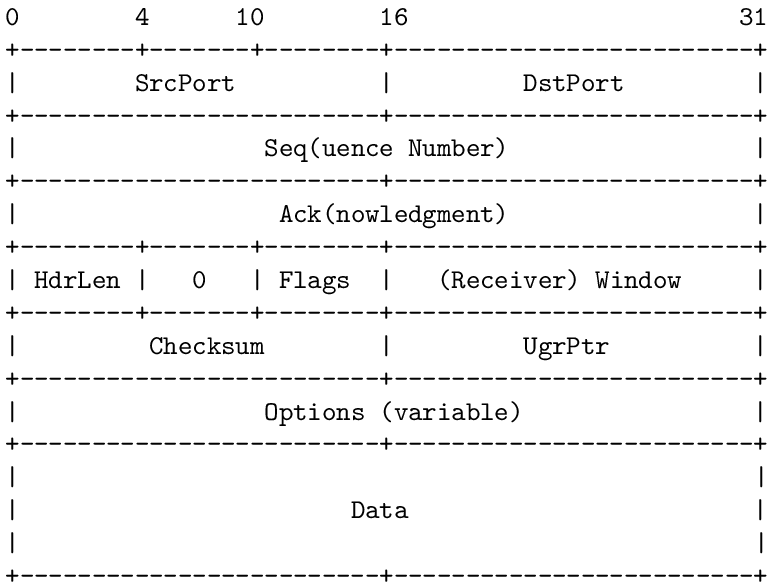

Figura: Esquema segmento TCP.

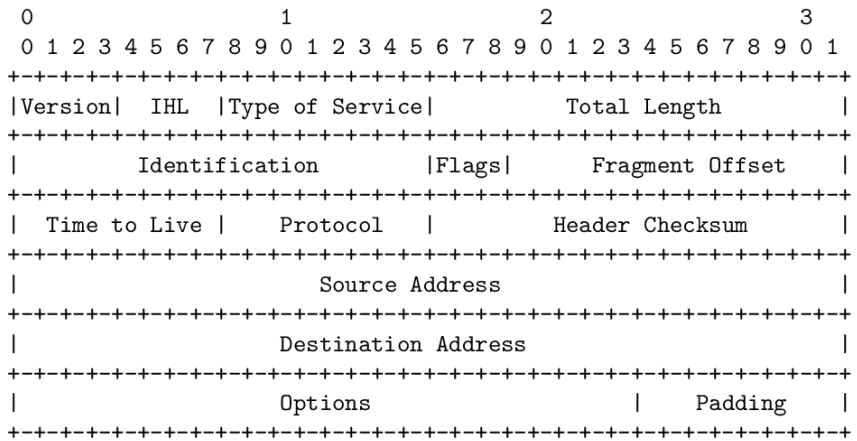

Figura: Formato de la cabecera IPv4.

Soluciones actuales

- Cortafuegos de próxima generación (NGFW).
- Sistema de detección de intrusiones (IDS).
- Sistema de prevención de intrusiones (IPS).

Índice

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

Origen de los datos

- El dataset utilizado en este trabajo es `NF-UNSW-NB15-v3`, desarrollado como parte de un análisis realizado en la Universidad de Queensland, Australia.

- Contiene 53 atributos que describen características del tráfico de red y que permiten clasificar las muestras de tráfico en nueve clases de ataques o como conexiones benignas.

- Algunos de los datos que recoge, son:
  - La duración de la conexión.
  - Los bytes enviados y los recibidos.
  - Versiones de los protocolos utilizados.

Tipos de ataques registrados en el conjunto de datos

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

Cuadro: Clasificación de amenazas de seguridad del *dataset* `NF-UNSW-NB15-v3`.

Atributos

Los atributos utilizados para desarrollar los modelos de este trabajo son los 53 atributos del conjunto de datos original, excepto los siguientes que han sido eliminados:

- `IPV4_SRC_ADDR`
- `IPV4_DST_ADDR`
- `FLOW_START_MILLISECONDS`
- `FLOW_END_MILLISECONDS`

Etiquetas

- `Label`: Indica si se trata de una conexión benigna o maligna.
- `Attack`: Indica el tipo de ataque al que corresponde esa conexión.

Índice

1. Introducción

2. Planificación y costes

3. Metodología

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

Origen de los modelos neuronales

- Ramón y Cajal
- McCulloch/Pitts
- Rosenblatt

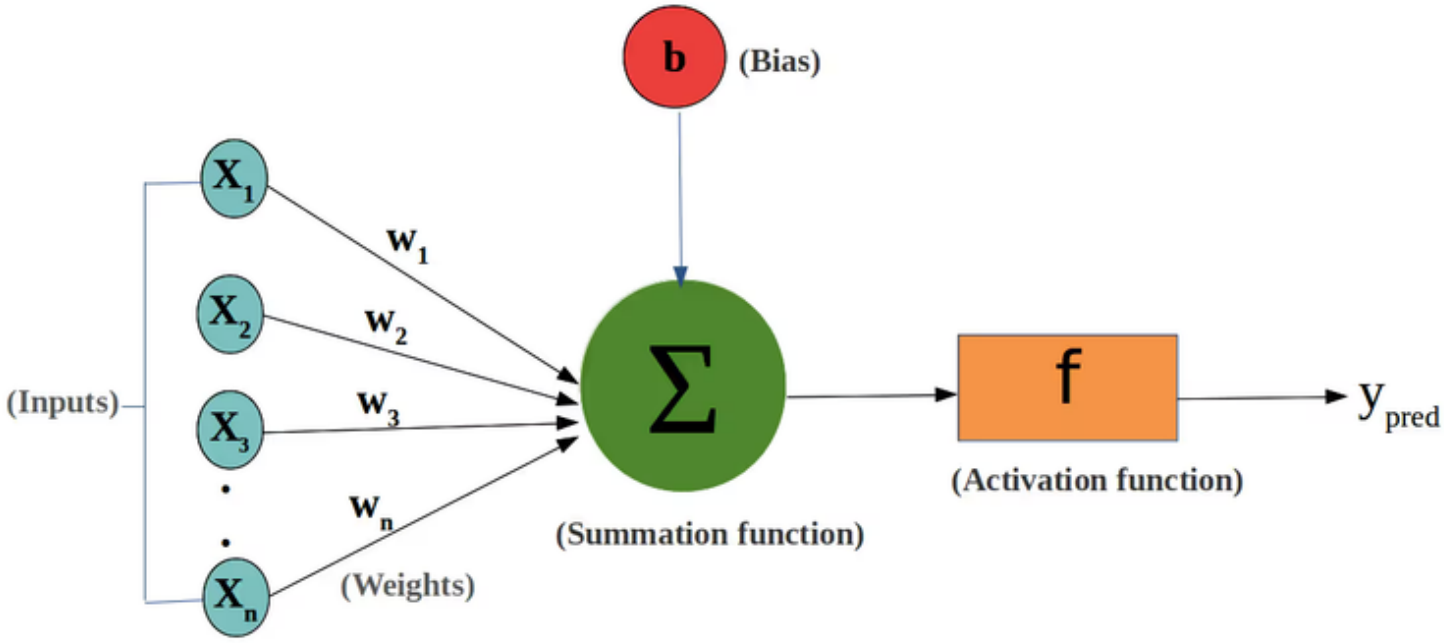

Figura: Esquema del funcionamiento de una neurona artificial.

Parámetros e hiperparámetros

- Los parámetros se ajustan a través del algoritmo de optimización, utilizando los cálculos de la función de pérdida durante la fase de entrenamiento de los modelos. Los parámetros de una red neuronal son:
  - Pesos.
  - Bias o sesgos.
- Los hiperparámetros son valores que se configuran antes de la fase de entrenamiento y 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*).

Matrices de confusión para el MCB

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

	Predicción Positiva	**Predicción Negativa**
Real Positivo	VP	FN
Real Negativo	FP	VN

Cuadro: Matriz de confusión para modelos de clasificación binaria.

Matriz de confusión para el MCM

	Predicción Clase 1	Predicción Clase 2	Predicción Clase 3	Predicción Clase 4	Predicción Clase 5	Predicción Clase 6	Predicción Clase 7	Predicción Clase 8	Predicción 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 multiclase.

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.

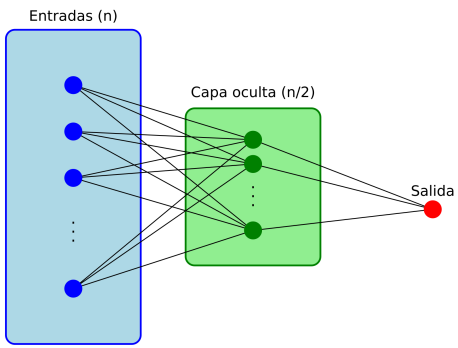

Figura: MCB25.

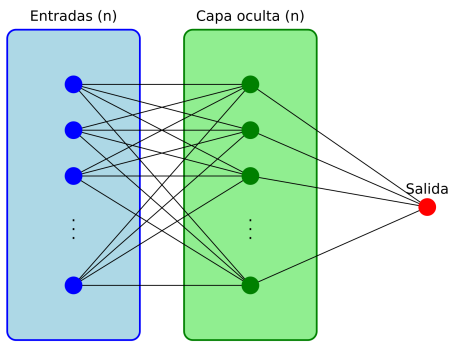

Figura: MCB49.

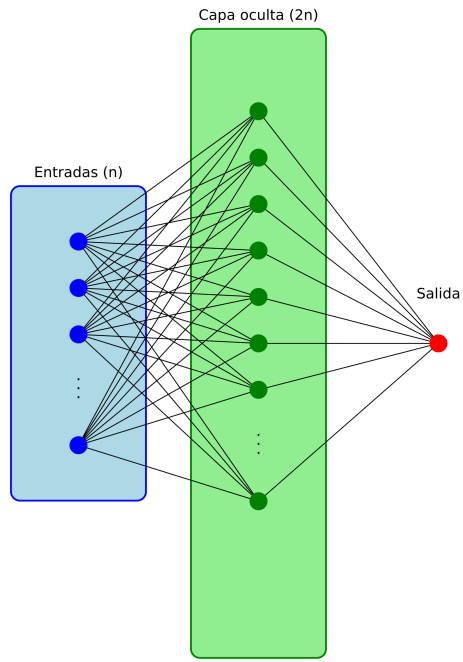

Figura: MCB98.

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.

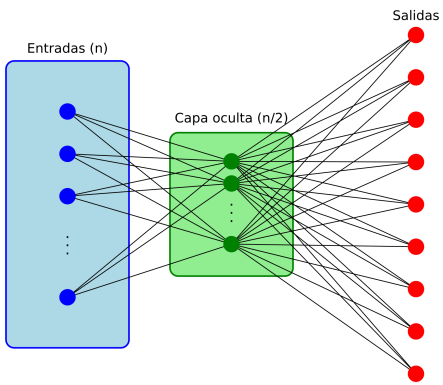

Figura: MCM25.

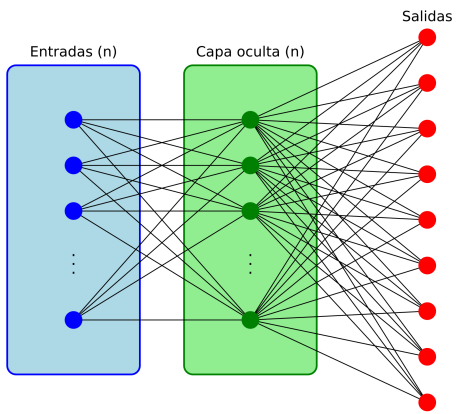

Figura: MCM49.

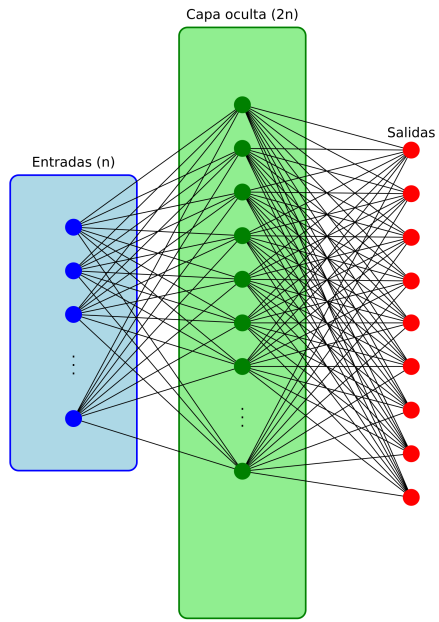

Figura: MCM98.

Implementación del MCB

- **Función de pérdida**: `BCEWithLogistLoss`
- **Algoritmo de optimización**: `AdamW`

```python
class Modelo(nn.Module):
    def __init__(self, input_dim, ventanaOculta):
        super().__init__()
        self.capa1 = nn.Linear(input_dim, ventanaOculta)
        self.capa2 =  nn.Linear(ventanaOculta, 1)

    def forward(self,  X):
        X = self.capa1(X)
        X = self.capa2(X)
        return X
```

Figura: Definición de la clase del modelo de clasificación binaria.

Hiperparámetro	Posibles valores
Batch size	[2000, 10000, 15000, 20000]
Learning rate	$[10^{-2}, 10^{-3}, 10^{-4}]$
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`

```python
class ModeloMulticlase(nn.Module):
    def __init__(self, input_dim, ventanaOculta, numClases):
        super().__init__()
        self.capa1 = nn.Linear(input_dim, ventanaOculta)
        self.bn1 = nn.BatchNorm1d(ventanaOculta, momentum=0.01)
        self.capa2 =  nn.Linear(ventanaOculta, numClases)

    def forward(self,  X):
        X = torch.relu(self.bn1(self.capa1(X)))
        X = self.capa2(X)
        return X
```

Figura: Definición de la clase del modelo de clasificación multiclase.

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

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

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
epochs	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.

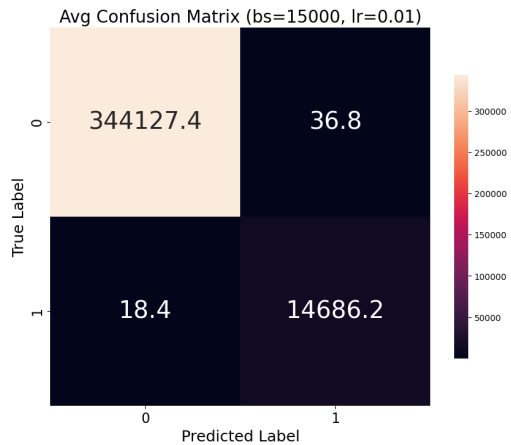

Figura: Matriz de confusión 1º MCB98.

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-MCM98	5º-MCM98
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.

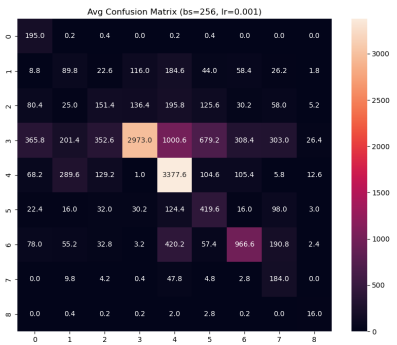

Figura: Matriz de confusión 1º MCM98.

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 multiclase.

Índice

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

Análisis de la 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 la 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 multiclase.

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.

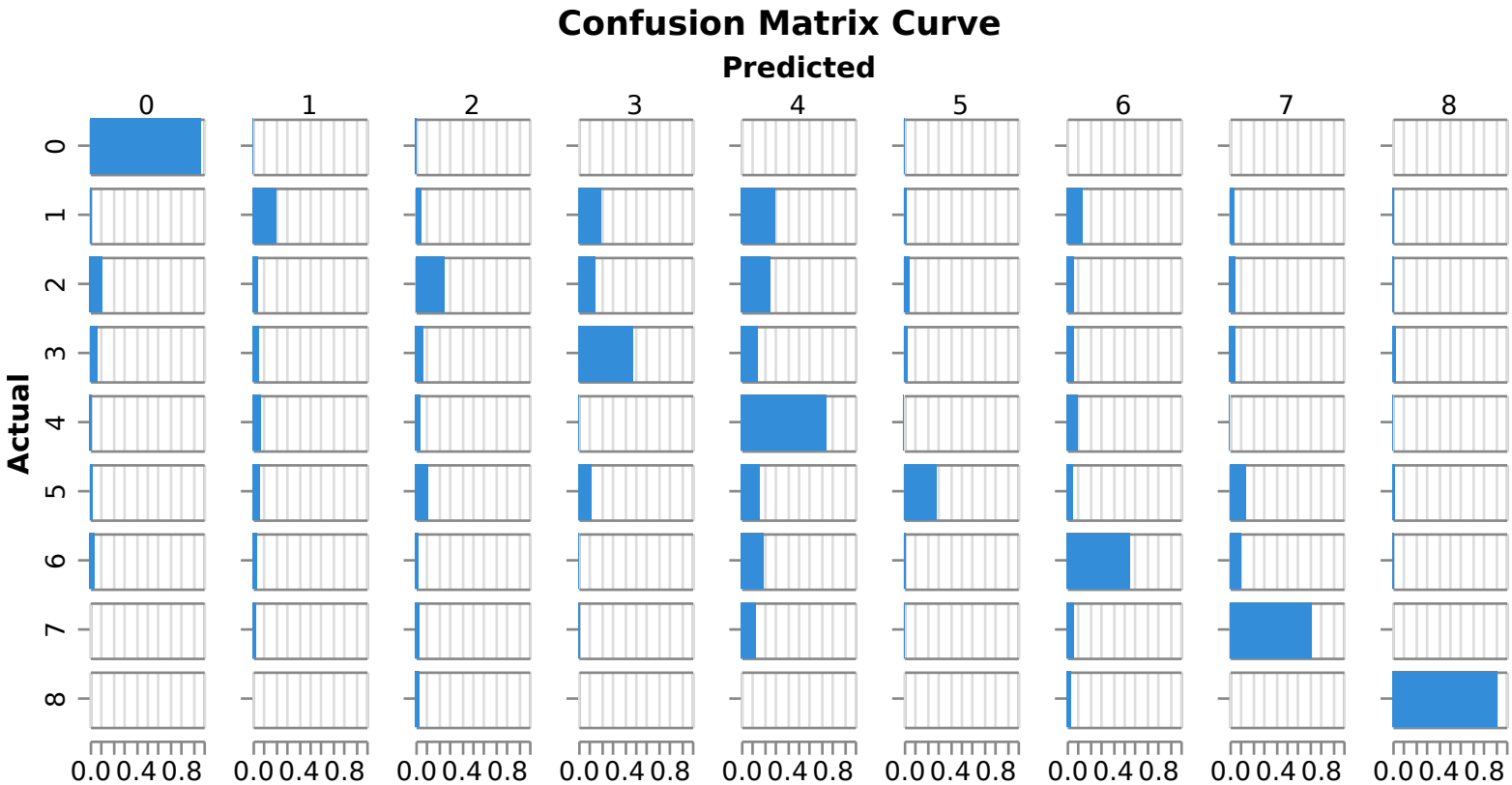

Figura: Matriz de confusión normalizada de la mejor configuración de hiperparámetros obtenida en el MCM98 durante la fase de evaluación.

Comparación de los resultados con otros modelos

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. Planificación y costes
3. Metodología
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 reducidir el número de falsos positivos y negativos.

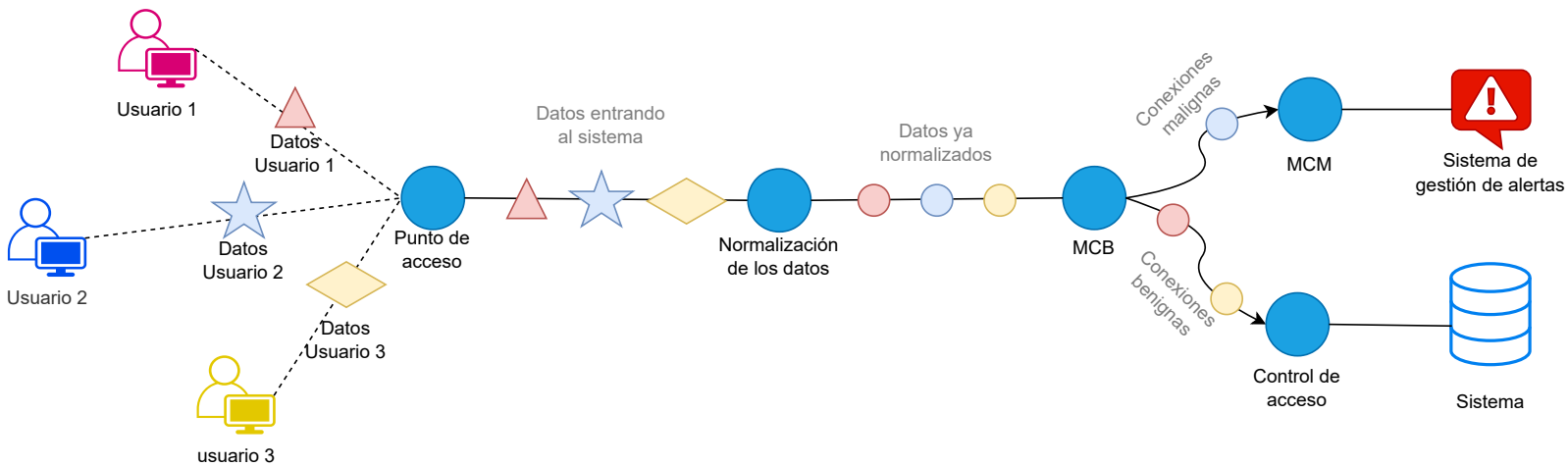

Figura: Ejemplo de despliegue de los modelos en un sistema informático.

Índice

1. Introducción
2. Planificación y costes
3. Metodología
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 clasifiación binaria (MCB) y un modelo de clasificación multiclase (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