

The background of the slide is a dark blue field filled with a complex, glowing network of lines and nodes. The lines are primarily blue and red, creating a sense of depth and connectivity. The nodes are small, bright points where the lines intersect. The overall effect is a futuristic, digital network visualization.

# **GRAPH CONVOLUTIONAL NETWORKS**

**Por Jesús y Carlos**



## 1. CONCEPTOS PREVIOS



## 2. INTRODUCCIÓN A GCN



## 3. CÓDIGO BÁSICO DE GCN



## 4. DETECCIÓN DE ANOMALÍAS: D.O.M.I.N.A.N.T



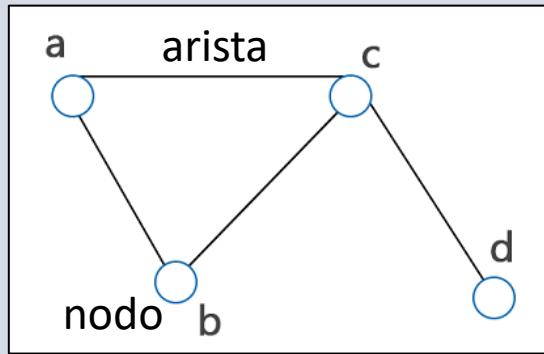
## 5. ALGORITMO D.O.M.I.N.A.N.T: IMPLEMENTACIÓN Y RESULTADOS

# ÍNDICE

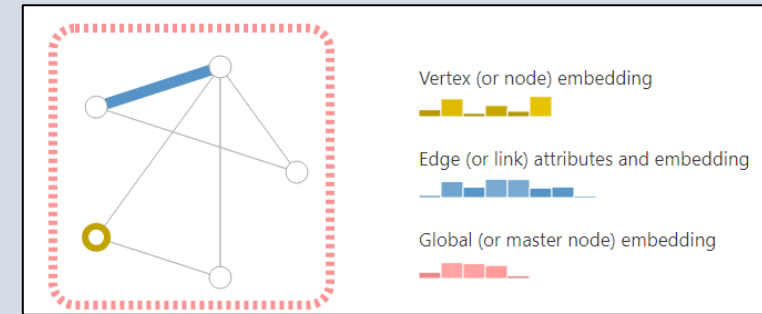


# CONCEPTOS PREVIOS:

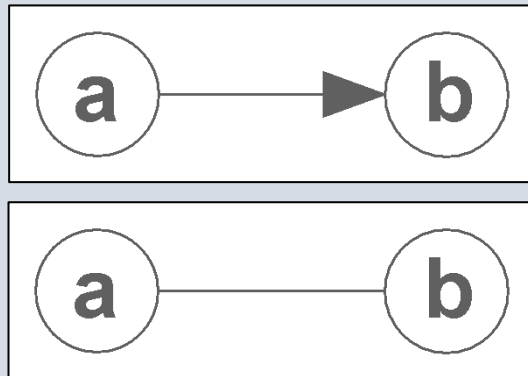
## GRAFOS



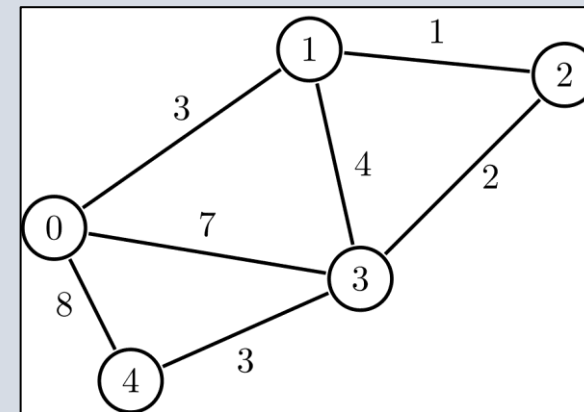
**Figura 1.** Representación de un grafo.



**Figura 3.** Información incrustada en los elementos.



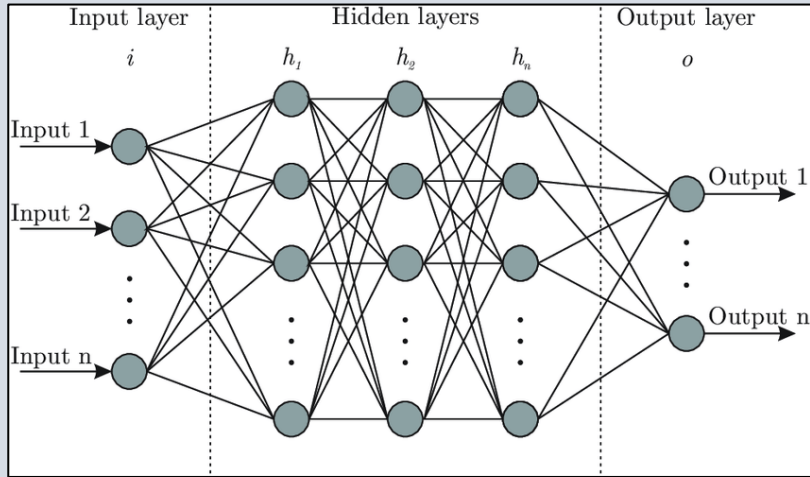
**Figura 2.** Enlace dirigido / Enlace no dirigido.



**Figura 4.** Grafo ponderado.



# CONCEPTOS PREVIOS: REDES NEURONALES



**Figura 5.** Esquema básico de una red neuronal.

## FUNCIONAMIENTO BÁSICO

Paso hacia delante  
(*Forward Pass*)

$$x_j^{l+1} = f \left( \sum_i w_{ij}^l x_i^l \right)$$

*Loss function  
evaluation*

$$C(y, \hat{y})$$

Paso hacia atrás  
(*Backward Pass*)

$$\frac{\partial C}{\partial \omega_i}$$



# CONCEPTOS PREVIOS: REDES NEURONALES

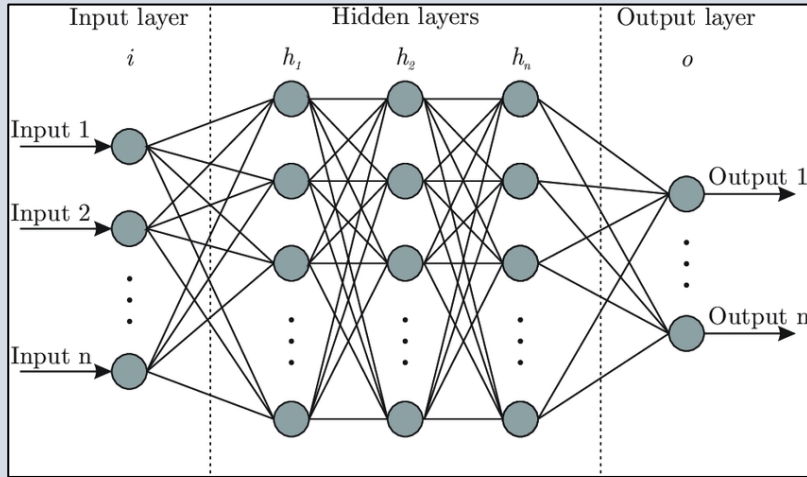


Figura 5. Esquema básico de una red neuronal.

## TIPOS DE REDES NEURONALES

Muchísimos...

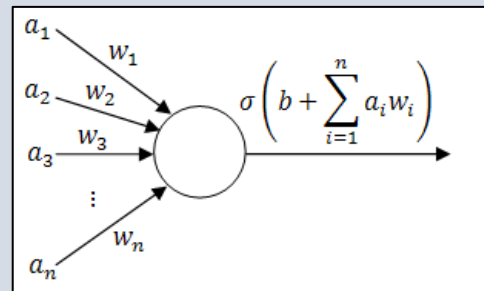


Figura 6. Perceptrón simple

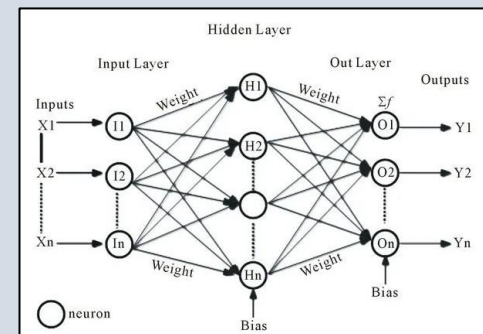


Figura 7. Perceptrón multicapa.

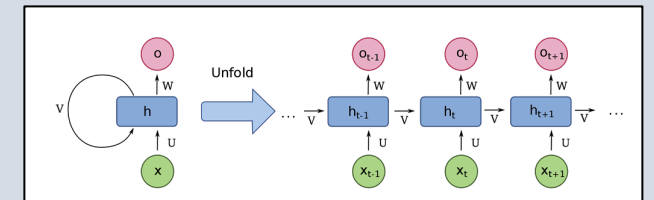


Figura 8. Redes recurrentes.

## FUNCIONAMIENTO BÁSICO

Paso hacia delante  
(Forward Pass)

$$x_j^{l+1} = f \left( \sum_i w_{ij}^l x_i^l \right)$$

Loss function  
evaluation

$$C(y, \hat{y})$$

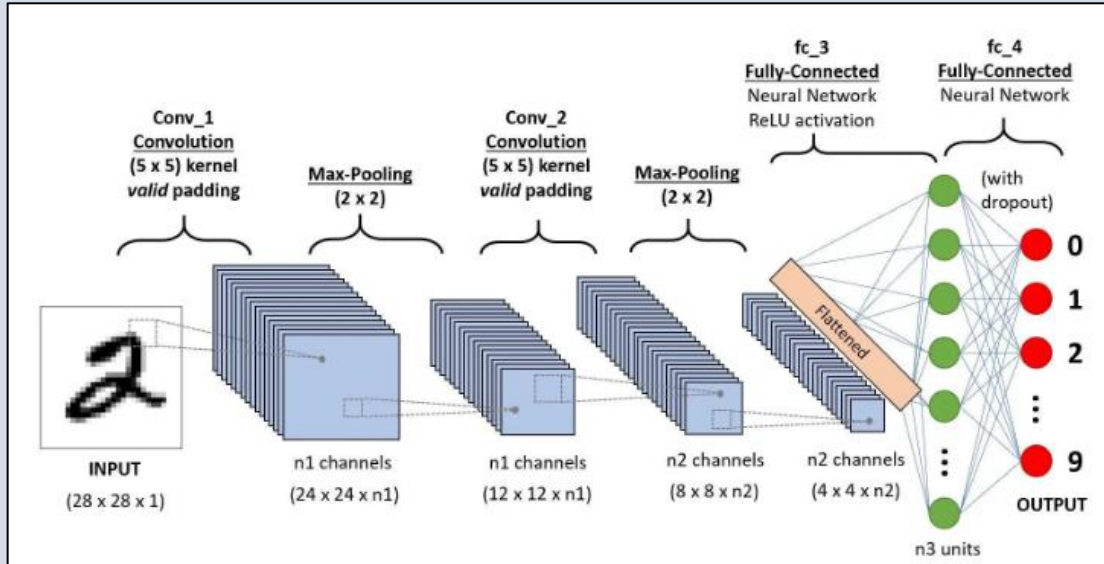
Paso hacia atrás  
(Backward Pass)

$$\frac{\partial C}{\partial \omega_i}$$



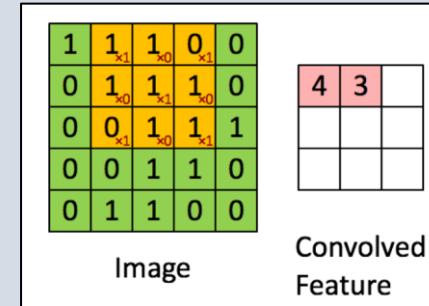
# CONCEPTOS PREVIOS:

## CNN



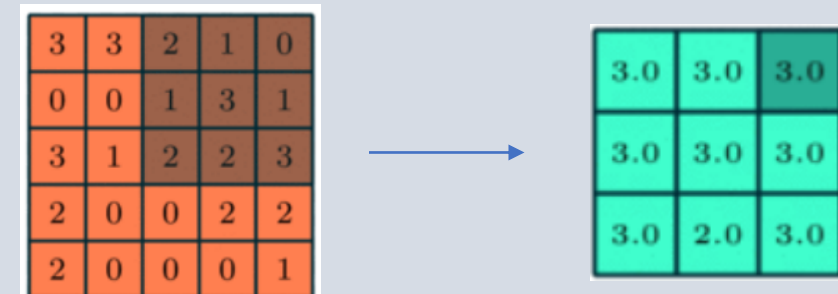
**Figura 9.** Arquitectura de una red convolucional simple.

Increíblemente buenas para  
extraer características de imágenes.



Los valores de los filtros  
van a ser autoaprendidos  
por la red.

**Figura 10.** Intuición de lo que sucede en una capa convolucional.



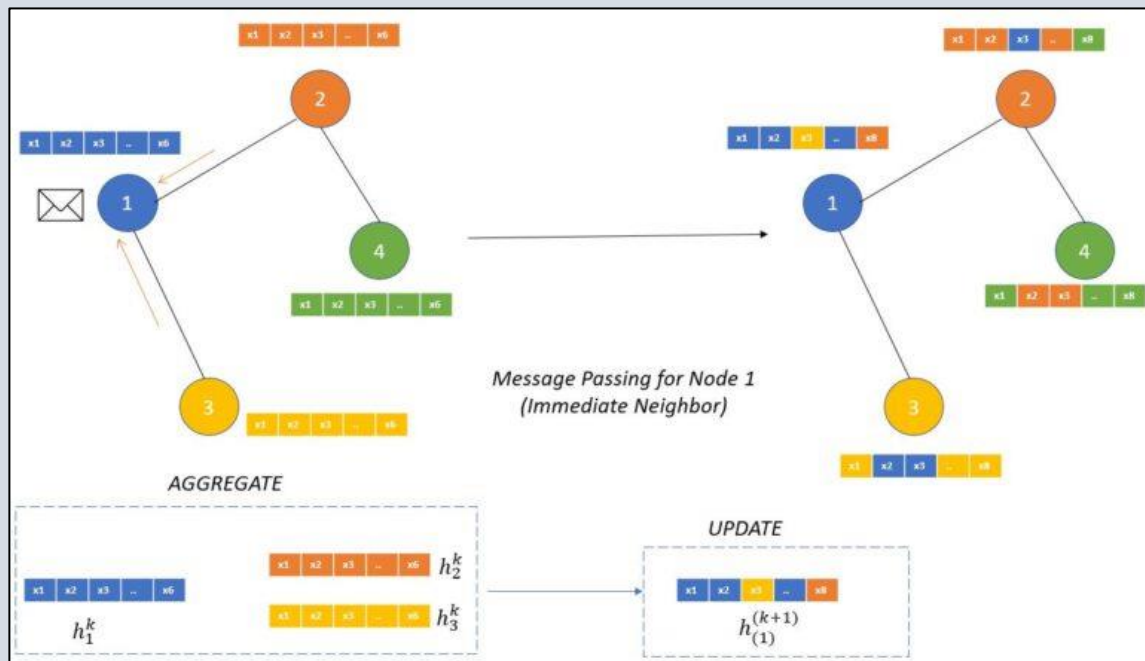
**Figura 11.** Intuición de lo que sucede en una capa de *pooling*.





# INTRODUCCIÓN A GCN

“Matrimonio” entre Grafos y redes convolucionales



**Figura 12.** Concepto de propagación de mensajes.

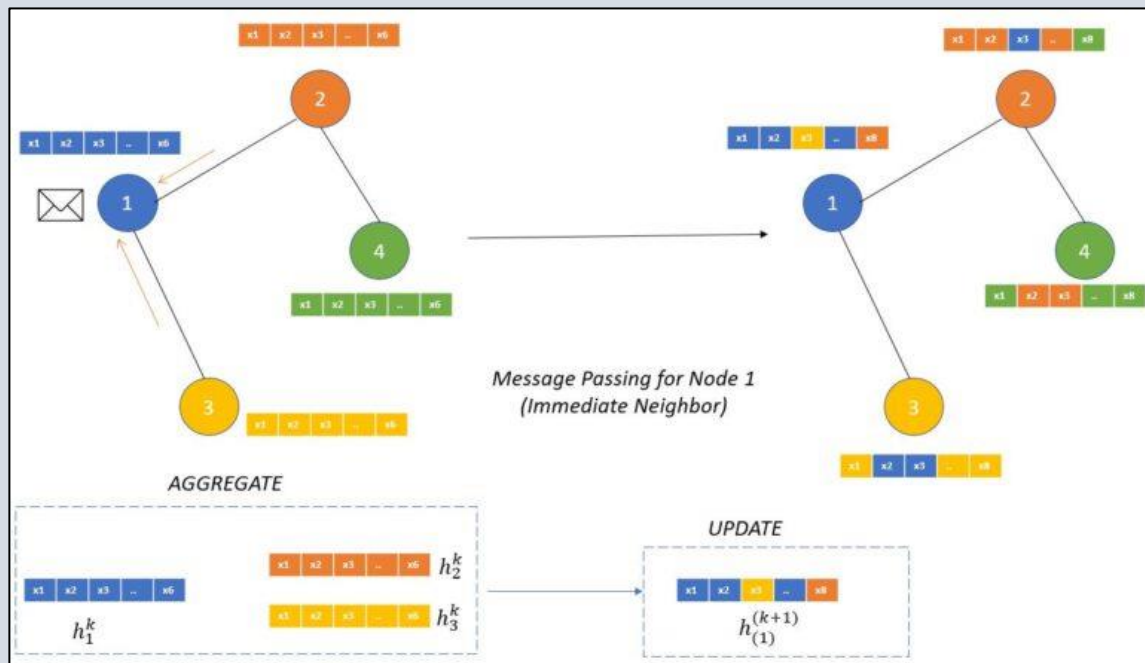
$$h_i = \sum_{j \in \mathcal{N}_i} \mathbf{W} x_j$$

Vector de representación de un nodo  
como combinación del de sus vecinos



# INTRODUCCIÓN A GCN

“Matrimonio” entre Grafos y redes convolucionales



**Figura 12.** Concepto de propagación de mensajes.

$$h_i = \sum_{j \in \mathcal{N}_i} \mathbf{W} x_j$$

Vector de representación de un nodo  
como combinación del de sus vecinos

$$h_i^{l+1} = f \left( \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{W} h_j^l \right)$$

Expresión general para capas  $l$ -ésimas

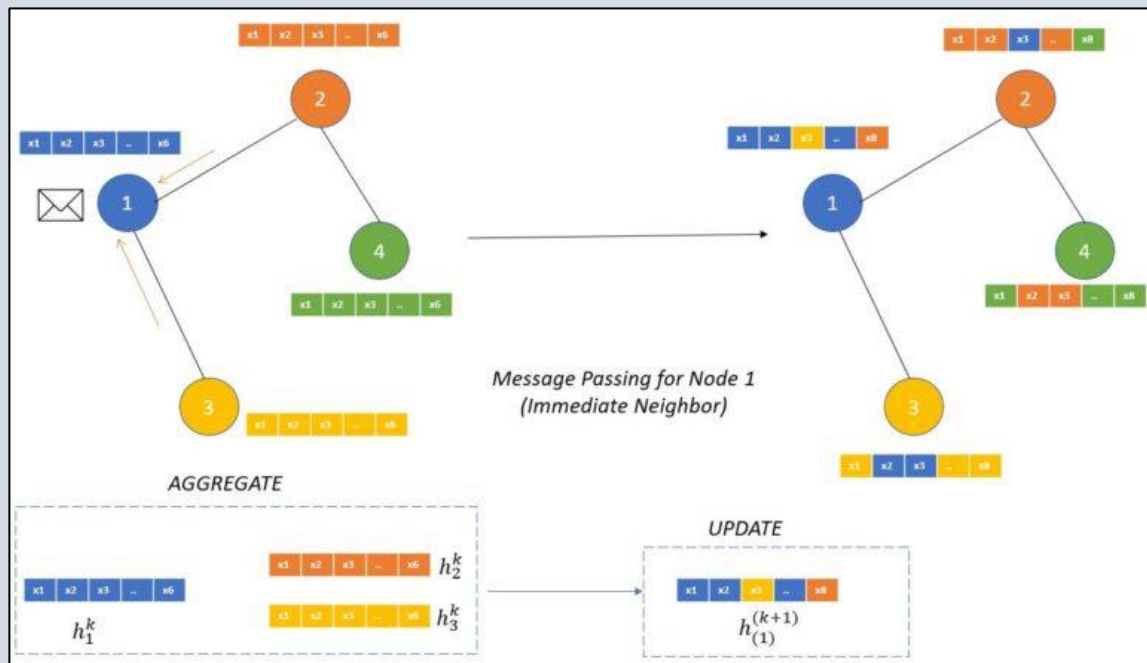
$$H^{l+1} = f(\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} H^l \mathbf{W}^l)$$





# INTRODUCCIÓN A GCN

“Matrimonio” entre Grafos y redes convolucionales



**Figura 12.** Concepto de propagación de mensajes.

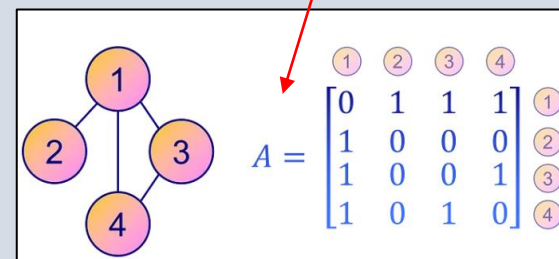
$$h_i = \sum_{j \in \mathcal{N}_i} \mathbf{W} x_j$$

Vector de representación de un nodo como combinación del de sus vecinos

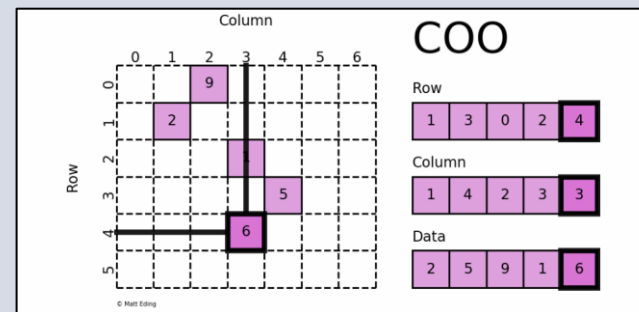
$$h_i^{l+1} = f \left( \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{W} h_j^l \right)$$

Expresión general para capas  $l$ -ésimas

$$H^{l+1} = f(\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} H^l \mathbf{W}^l)$$



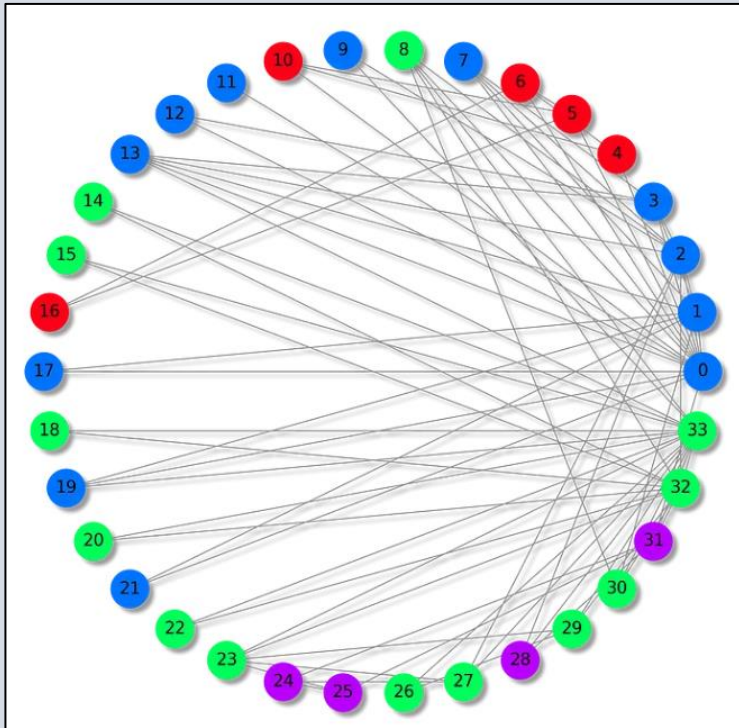
**Figura 13.** Matriz de adyacencia.



**Figura 14.** Formato Coordinate list (COO).



# CÓDIGO BÁSICO DE GCN



```
Dataset: KarateClub():  
=====  
Número de grafos: 1  
Número de características: 34  
Número de clases: 4
```

```
directed = False  
contains_isolated_nodes = False  
contains_self_loops = False
```

**X** = Matriz Identidad  
(no hay info adicional  
incrustada)

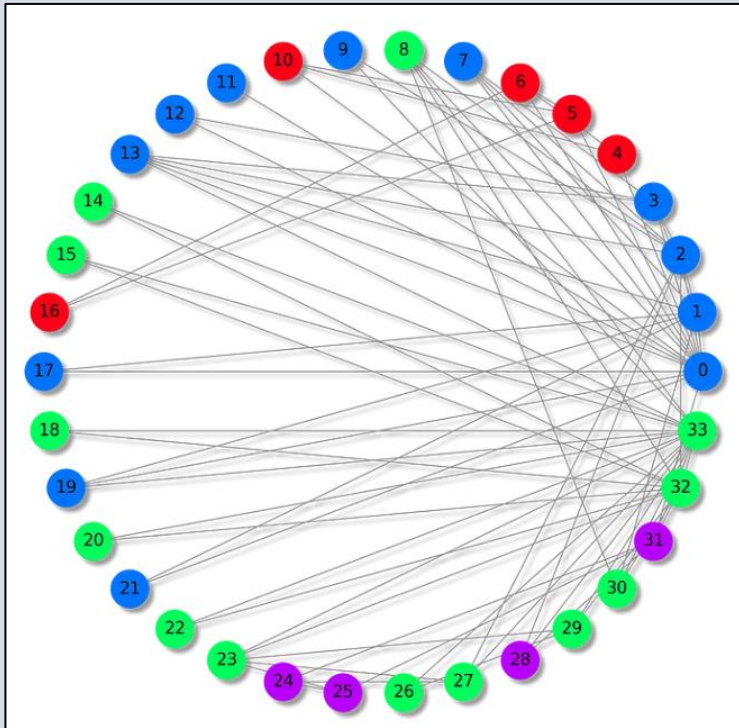
**Figura 15.** *Dataset* utilizado: Club de Karate de Zachary

## OBJETIVO

Asignar al grupo correcto a cada miembro  
(clasificación de nodo)



# CÓDIGO BÁSICO DE GCN



```
Dataset: KarateClub():  
=====  
Número de grafos: 1  
Número de características: 34  
Número de clases: 4
```

```
directed = False  
contains_isolated_nodes = False  
contains_self_loops = False
```

**X** = Matriz Identidad  
(no hay info adicional  
incrustada)

```
class GCN(torch.nn.Module):  
  
    def __init__(self):  
        super().__init__()  
        self.gcn = GCNConv(dataset.num_features, 3)  
        self.out = Linear(3, dataset.num_classes)  
  
    def forward(self, x, edge_index):  
        h = self.gcn(x, edge_index).relu()  
        z = self.out(h)  
        return h, z  
  
model = GCN()
```

Arquitectura de la red

**Figura 15.** Dataset utilizado: Club de Karate de Zachary

## OBJETIVO

Asignar al grupo correcto a cada miembro  
(clasificación de nodo)



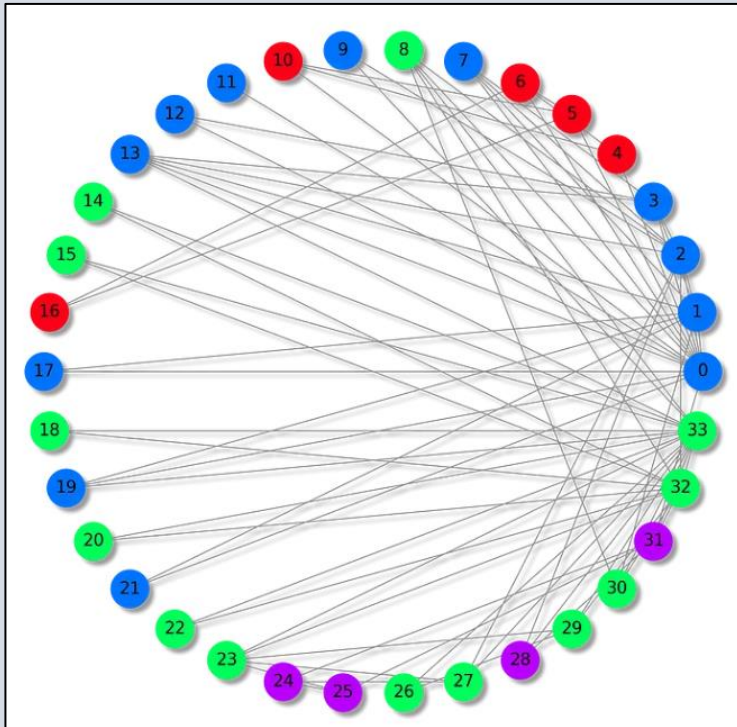
# CÓDIGO BÁSICO DE GCN



PyG



NetworkX  
Network Analysis in Python



```
Dataset: KarateClub():  
=====  
Número de grafos: 1  
Número de características: 34  
Número de clases: 4
```

```
directed = False  
contains_isolated_nodes = False  
contains_self_loops = False
```

$X$  = Matriz Identidad  
(no hay info adicional  
incrustada)

```
class GCN(torch.nn.Module):  
  
    def __init__(self):  
        super().__init__()  
        self.gcn = GCNConv(dataset.num_features, 3)  
        self.out = Linear(3, dataset.num_classes)  
  
    def forward(self, x, edge_index):  
        h = self.gcn(x, edge_index).relu()  
        z = self.out(h)  
        return h, z  
  
model = GCN()
```

Arquitectura de la red

```
criterion = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters(), lr = 0.01)
```

Función de pérdida y optimizador utilizados

```
for epoch in range(totalEpochs):  
  
    # Reset gradientes  
    optimizer.zero_grad()  
  
    # Forward pass  
    h, z = model(data.x, data.edge_index)  
  
    # Calcular la pérdida  
    loss = criterion(z, data.y)  
  
    # Calcular la precisión  
    acc = accuracy(z.argmax(dim=1), data.y)  
  
    # Backward pass  
    loss.backward()  
  
    # Actualizar los pesos  
    optimizer.step()
```

Entrenamiento

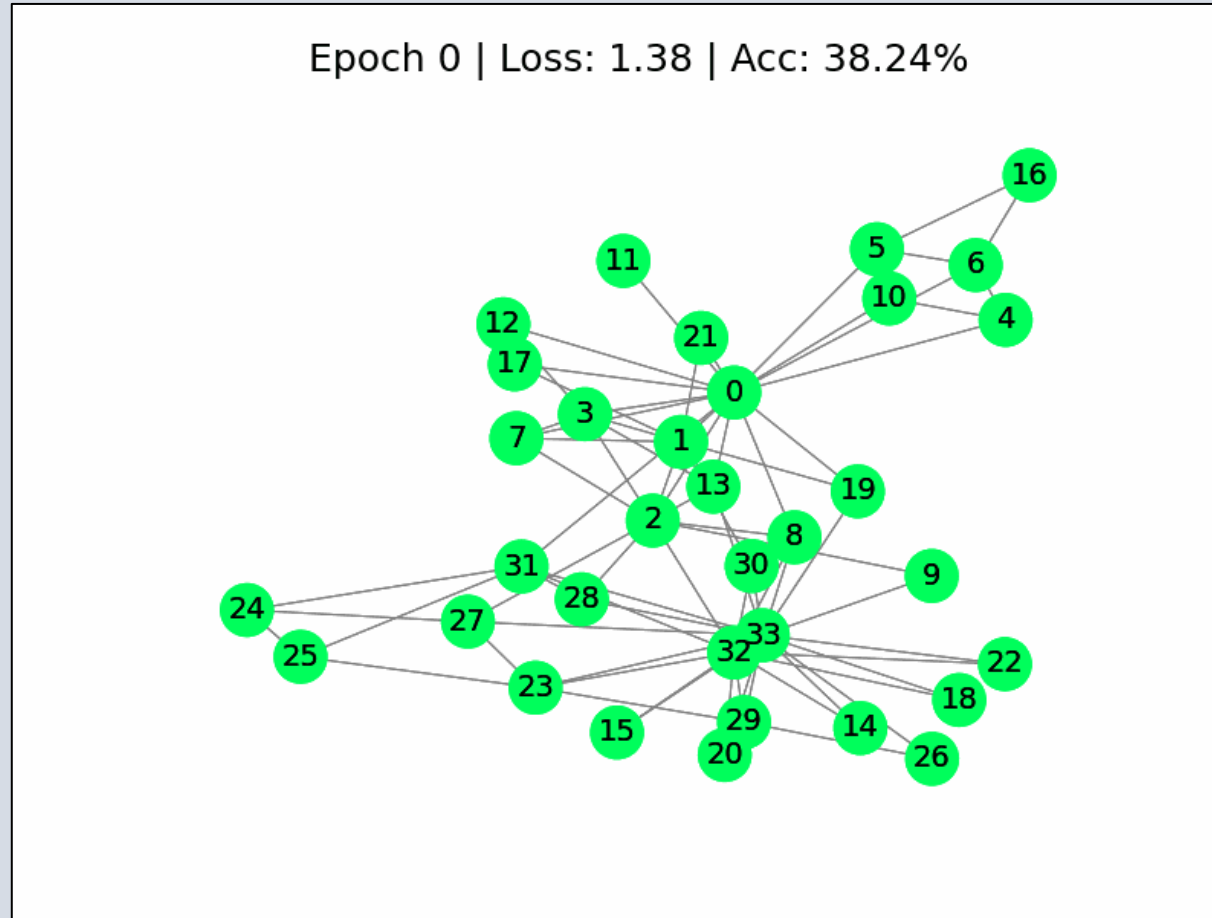
Figura 15. Dataset utilizado: Club de Karate de Zachary

## OBJETIVO

Asignar al grupo correcto a cada miembro  
(clasificación de nodo)



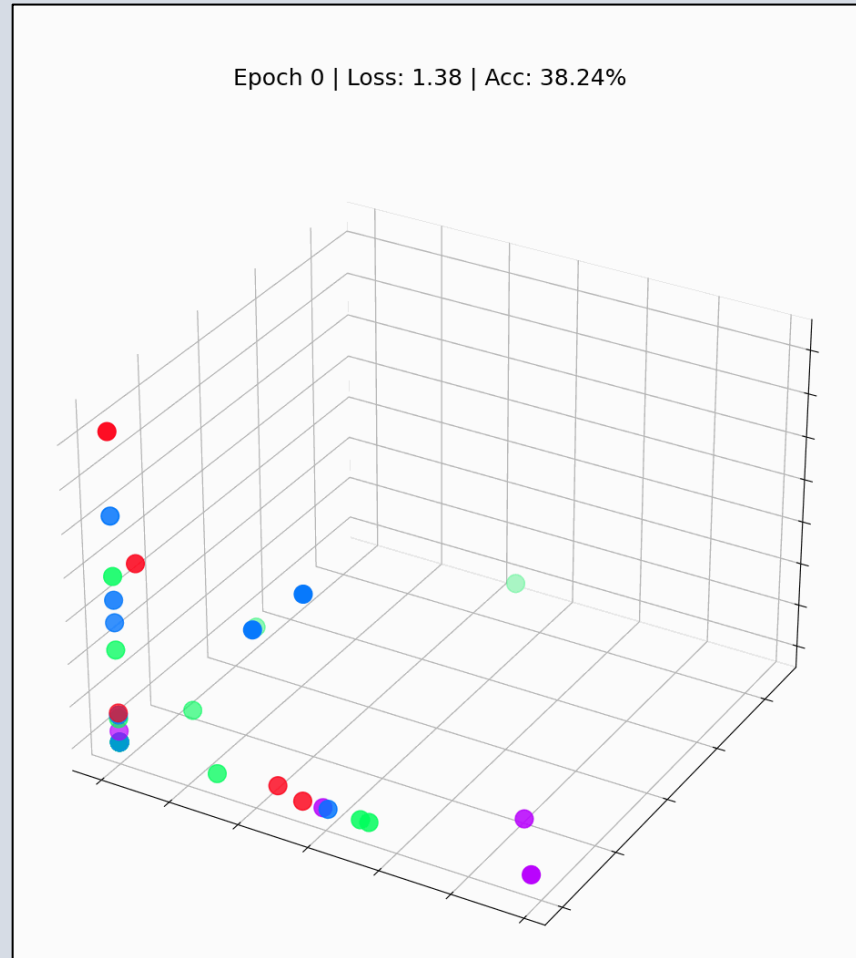
## ¿RESULTADOS?



**Figura 16.** Evolución en las predicciones del grafo



## ¿RESULTADOS?



**Figura 17.** Evolución de los *embeddings* que llegan a la capa lineal





# DETECCIÓN DE ANOMALÍAS

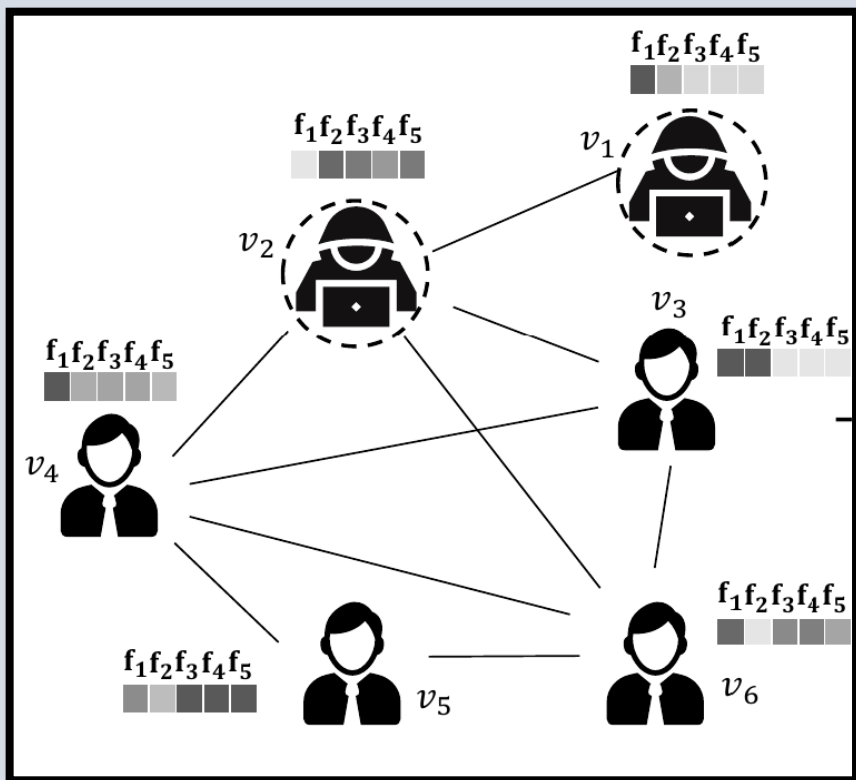


Figura 18. Esquema conceptual de un grafo con nodos con atributos

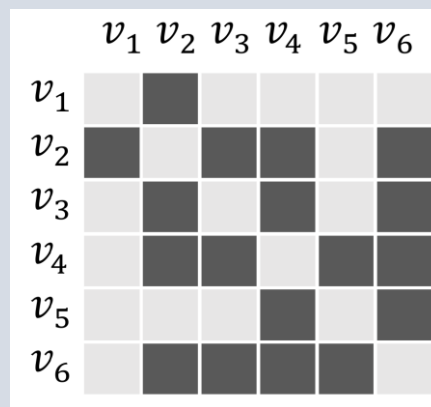


Figura 19. Matriz de enlaces

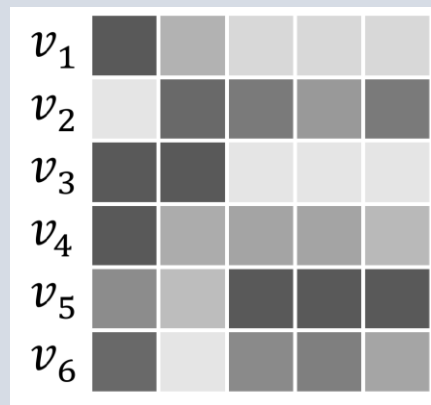
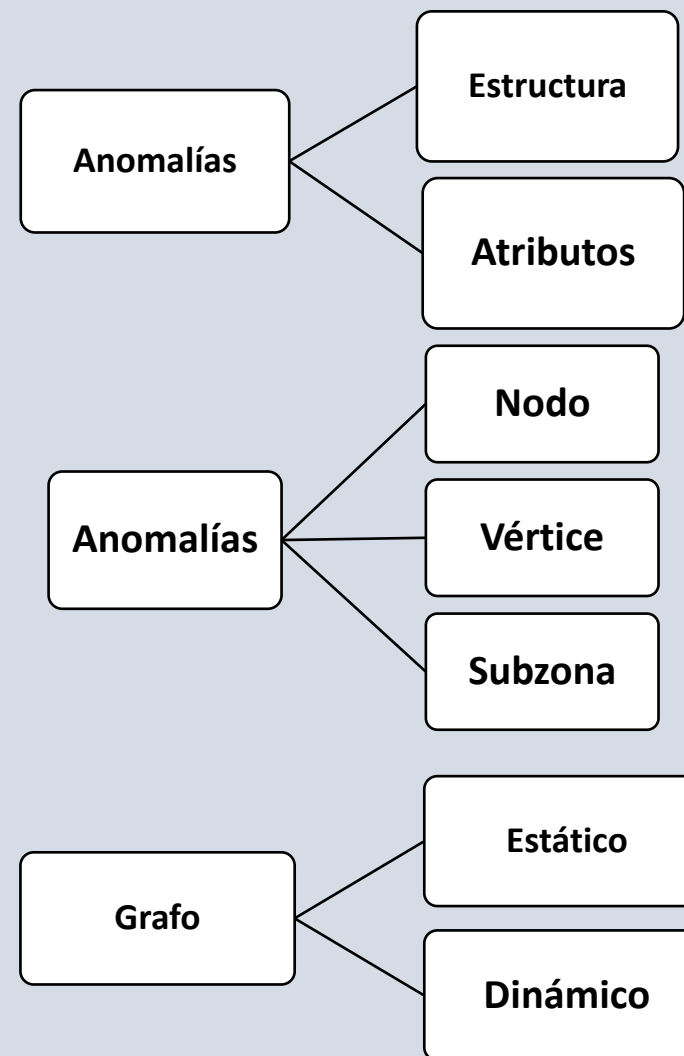


Figura 20. Matriz de atributos

## CLASIFICACIÓN BINARIA:

- Dato común
- Anomalía





# DETECCIÓN DE ANOMALÍAS

Graph type	Anomaly type	Network architecture	Method	Summary (key issue addressed → solution)
Static graph	Node anomaly	GCN-based GAE	Dominant [7] (2019)	Complex interactions, sparsity, non-linearity → GCN-based encoder
			Dual-SVDAE [21] (2021)	Overfitting for normal & abnormal → hypersphere embedding space
			GUIDE [22] (2021)	Complex interactions → higher-order structure decoder
			SpecAE [8] (2019)	Over-smoothing issue → tailored embedding space
			ComGA [23] (2022)	Over-smoothing issue → community-specific representation
			ALARM [24] (2020)	Heterogeneous attributes → multiple GCN-based encoders
			AnomMAN [25] (2022)	Heterogeneous attributes → multiple GCN-based encoders
			SL-GAD [26] (2021)	Contextual information → subgraph sampling & contrastive learning
		GCN alone	Semi-GCN [27] (2021)	Label information → semi-supervised learning by GCN
			HCM [28] (2021)	Label & contextual information → hop-count prediction model
			ResGCN [29] (2021)	Over-smoothing issue → GCN with residual-based attention
			CoLA [30] (2021)	Targeting issue of GAE → contrastive self-supervised learning
			ANEMONE [31] (2021)	Contextual information → multi-scale contrastive learning
			PAMFUL [32] (2021)	Contextual information → pattern mining algorithm with GCN
		GAT-based GAE	AnomalyDAE [33] (2020)	Complex interactions → GAT-based encoder
			GATAE [34] (2020)	Over-smoothing issue → GAT-based encoder
			AEGIS [35] (2020)	Handling unseen nodes → generative adversarial learning with GAE
		Other GNN-based model	OCGNN [36] (2021)	Targeting issue of GAE → GNN with hypersphere embedding space
			AAGNN [37] (2021)	Targeting issue of GAE → GNN with hypersphere embedding space
			Meta-GDN [38] (2021)	Hard work to label anomalies → meta-learning with auxiliary graphs
			AANE [39] (2020)	Noise or adversarial links → GAE with a loss for anomalous links
	Edge anomaly	GCN-based GAE	eFraudCom [40] (2022)	Fraud detection → heterogeneous graph and representative data sampling
		GCN alone	SubGNN [41] (2021)	Fraud detection → GIN and extracting and relabeling subgraphs
	Subgraph anomaly	GAT-based GAE	HO-GAT [42] (2021)	Abnormal subgraphs → hybrid-order attention with motif instances
	Graph-level anomaly	GCN alone	OCGIN [43] (2021)	Graph-level anomaly detection → graph classification with GIN
			OCGTL [44] (2022)	Hypersphere collapse → set of GNNs for embedding
			GLocalKD [45] (2022)	Graph-level anomalies → joint learning global & local normality
Dynamic graph	Edge anomaly	GCN and GRU	AddGraph [10] (2019)	Long-term patterns → temporal GCN with attention-based GRU
			DynAD [46] (2020)	Long-term patterns → temporal GCN with attention-based GRU
			Hierarchical-GCN [47] (2020)	Dynamic data evaluation → temporal & hierarchical GCN
			StrGNN [48] (2021)	Structural change → mining unusual temporal subgraph structures
			H-VGRAE [49] (2020)	Anomalous nodes → modeling stochasticity and multi-scale ST dependency
	Node anomaly	GCN & DRNN-based GAE	DEGCN [50] (2022)	To capture node- and global-level patterns → DGCN and GGRU
		GCN alone	TDG with GCN [51] (2022)	Malicious connections on traffic → extracting TDGs

**Figura 21.** Tabla de algoritmos según tipo de grafo y de anomalía y arquitectura

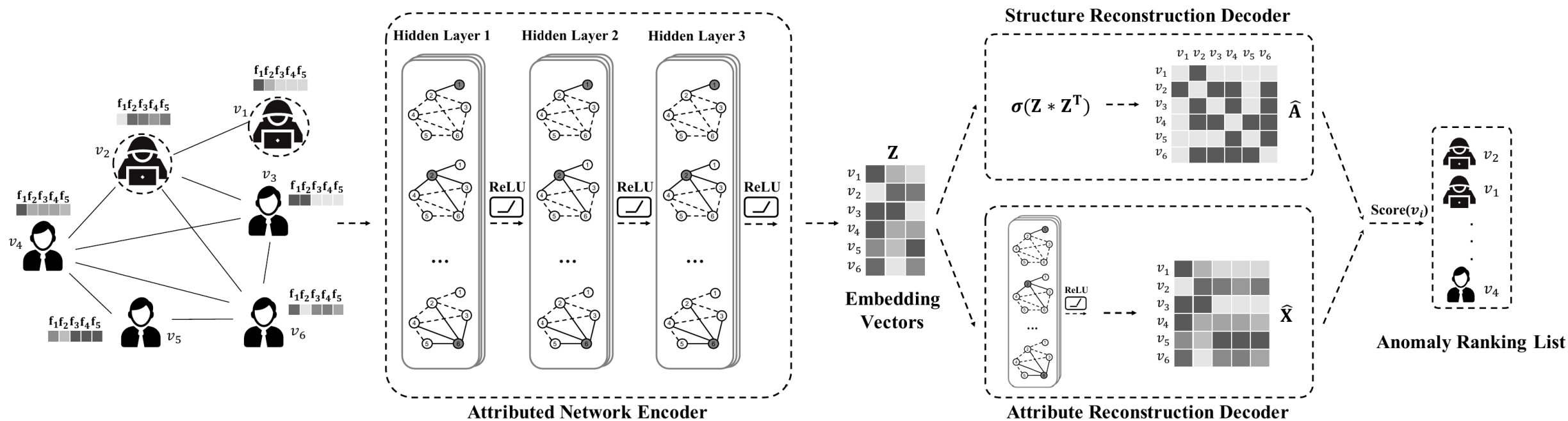
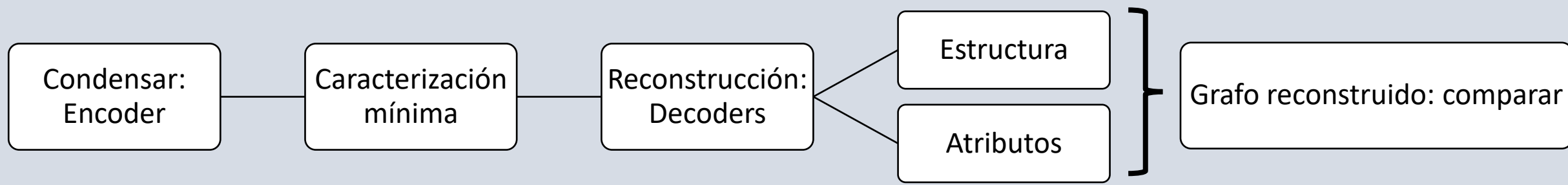
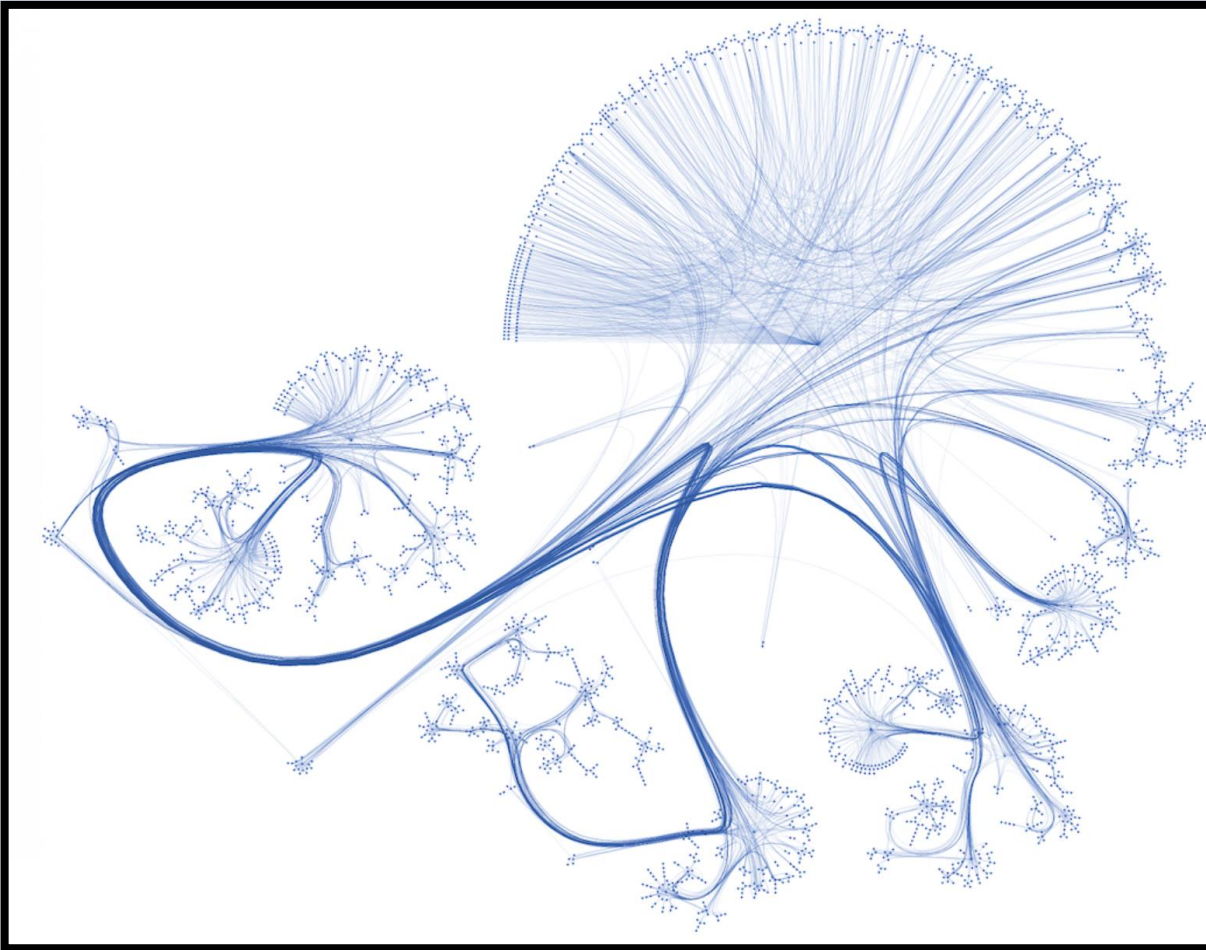


Figura 22. Esquema algoritmo DOMINANT





## IMPLEMENTACIÓN



**Figura 23.** Dataset CORA:  
colección de artículos científicos divididos en categorías

**Nodos:** Representan los documentos (artículos científicos).

**Aristas:** Representan las relaciones de citación entre documentos (citaciones entre ellos).

**Etiquetas:** Cada documento está etiquetado con una categoría específica (temática del artículo).

**Atributos:** Cada documento está representado por un vector de características binarias, cada característica representa la presencia o ausencia de una palabra en el documento (de una lista de 1443 palabras de interés)



# IMPLEMENTACIÓN

## 1) Cargamos el Dataset



# IMPLEMENTACIÓN

1) Cargamos el Dataset

2) Generamos Anomalías de forma artificial

```
data, ya = gen_contextual_outlier(data, n=100, k=10)
data, ys = gen_structural_outlier(data, m=10, n=10)
```

```
data.y = torch.logical_or(ys, ya).long()
data_f = torch.logical_or(ys, ya).long()
```

✓ 0.0s

```
#contamos cuantas etiquetas tiene data.y
print(data.y.unique(return_counts=True))
```

✓ 0.0s

```
(tensor([0, 1]), tensor([2510, 198]))
```





# IMPLEMENTACIÓN

- 1) Cargamos el Dataset
- 2) Generamos Anomalías de forma artificial
- 3) Inicializamos el detector y lo entrenamos**

```
from pygod.detector import DOMINANT  
  
detector = DOMINANT(hid_dim=256, num_layers=16, epoch=300, contamination=0.05)
```

✓ 0.0s

Python

```
detector.fit(data)
```

✓ 1m 38.7s

Python



# IMPLEMENTACIÓN

- 1) Cargamos el Dataset
- 2) Generamos Anomalías de forma artificial
- 3) Inicializamos el detector y lo entrenamos
- 4) Predecimos**

```
pred, score, prob, conf = detector.predict(data,
                                           return_pred=True,
                                           return_score=True,
                                           return_prob=True,
                                           return_conf=True)

import pandas as pd

df = pd.DataFrame({'pred': pred, 'score': score, 'prob': prob, 'conf': conf})

df
```

✓ 0.2s Open 'df' in Data Wrangler

Python



# IMPLEMENTACIÓN

- 1) Cargamos el Dataset
- 2) Generamos Anomalías de forma artificial
- 3) Inicializamos el detector y lo entrenamos
- 4) Predecimos

```
pred, score, prob, conf = detector.predict(data,
return_pred=True,
return_score=True,
return_prob=True,
return_conf=True)

import pandas as pd

df = pd.DataFrame({'pred': pred, 'score': score, 'prob': prob, 'conf': conf})

df
```

✓ 0.2s Open 'df' in Data Wrangler

Python

	pred	score	prob	conf
0	0	0.988793	0.073405	1.0
1	0	0.966952	0.069531	1.0
2	0	1.284904	0.125929	1.0
3	0	0.607964	0.005853	1.0
4	0	1.229844	0.116163	1.0
...	...	...	...	...
2703	0	0.603081	0.004987	1.0
2704	0	0.667596	0.016431	1.0
2705	0	0.613789	0.006887	1.0
2706	0	1.132433	0.098884	1.0
2707	0	1.196528	0.110253	1.0

Figura 24. Predicciones



# IMPLEMENTACIÓN

- 1) Cargamos el Dataset
- 2) Generamos Anomalías de forma artificial
- 3) Inicializamos el detector y lo entrenamos
- 4) Predecimos

```
pred, score, prob, conf = detector.predict(data,
return_pred=True,
return_score=True,
return_prob=True,
return_conf=True)

import pandas as pd

df = pd.DataFrame({'pred': pred, 'score': score, 'prob': prob, 'conf': conf})

df
```

✓ 0.2s Open 'df' in Data Wrangler

Python

	pred	score	prob	conf
0	0	0.988793	0.073405	1.0
1	0	0.966952	0.069531	1.0
2	0	1.284904	0.125929	1.0
3	0	0.607964	0.005853	1.0
4	0	1.229844	0.116163	1.0
...	...	...	...	...
2703	0	0.603081	0.004987	1.0
2704	0	0.667596	0.016431	1.0
2705	0	0.613789	0.006887	1.0
2706	0	1.132433	0.098884	1.0
2707	0	1.196528	0.110253	1.0

Figura 24. Predicciones



## IMPLEMENTACIÓN

- 1) Cargamos el Dataset
- 2) Generamos Anomalías de forma artificial
- 3) Inicializamos el detector y lo entrenamos
- 4) Predecimos

```
pred, score, prob, conf = detector.predict(data,
return_pred=True,
return_score=True,
return_prob=True,
return_conf=True)

import pandas as pd


df = pd.DataFrame({'pred': pred, 'score': score, 'prob': prob, 'conf': conf})

df
```

✓ 0.2s Open 'df' in Data Wrangler

Python

Cuanto mayor sea el valor “score”  
de un nodo, más probabilidades  
tendrá de ser una anomalía



	pred	score	prob	conf
0	0	0.988793	0.073405	1.0
1	0	0.966952	0.069531	1.0
2	0	1.284904	0.125929	1.0
3	0	0.607964	0.005853	1.0
4	0	1.229844	0.116163	1.0
...	...	...	...	...
2703	0	0.603081	0.004987	1.0
2704	0	0.667596	0.016431	1.0
2705	0	0.613789	0.006887	1.0
2706	0	1.132433	0.098884	1.0
2707	0	1.196528	0.110253	1.0

Figura 24. Predicciones



## RESULTADOS

### 5) Evaluamos el Algoritmo

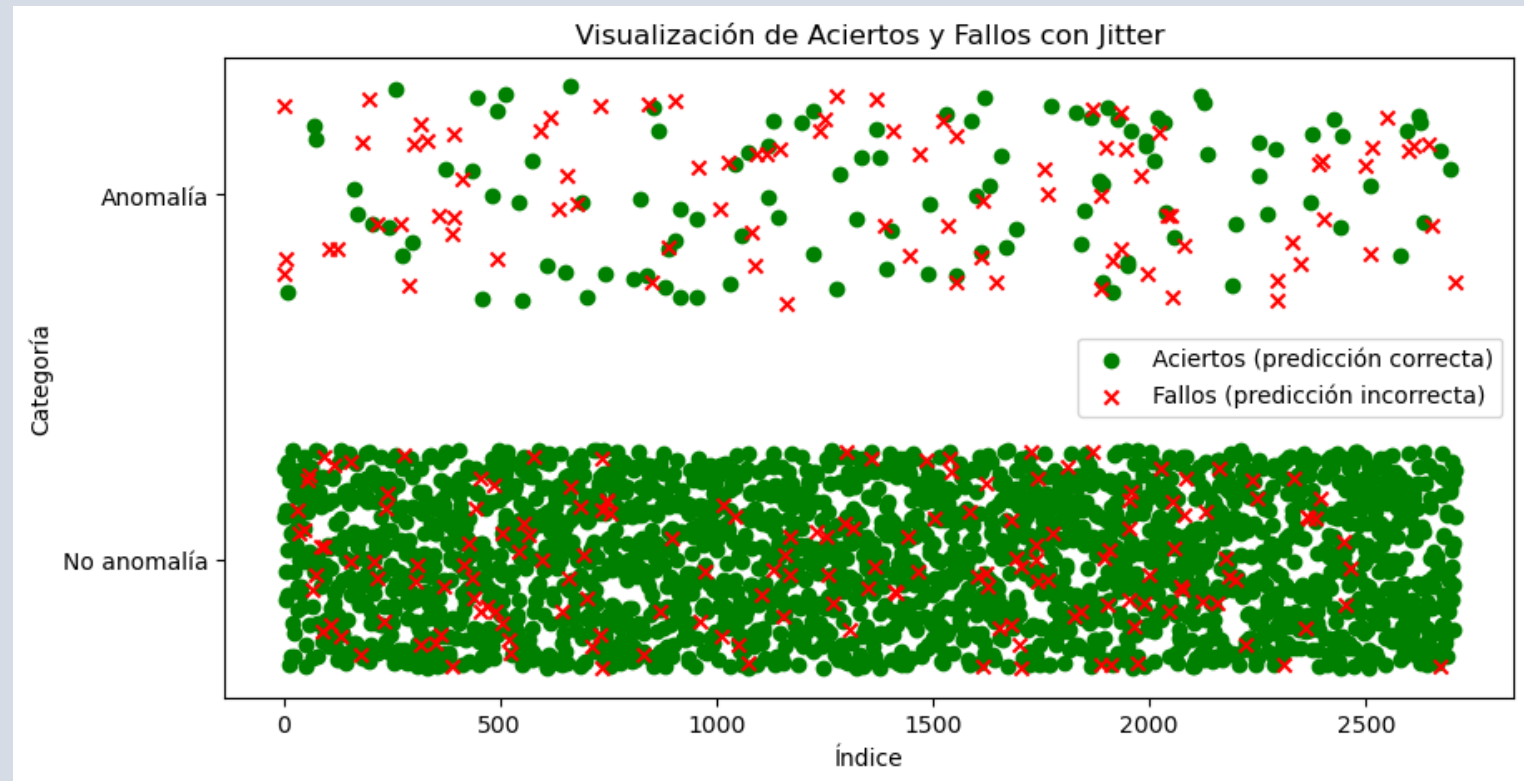


Figura 25. Predicciones (visual)





# RESULTADOS

## 5) Evaluamos el Algoritmo

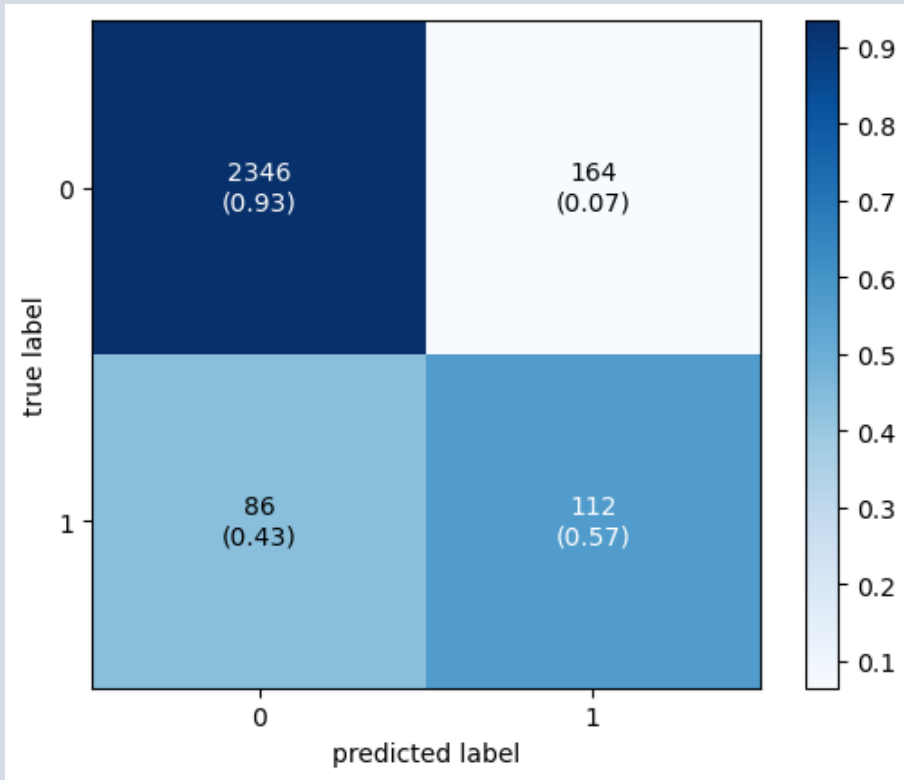


Figura 26. Matriz de Confusión

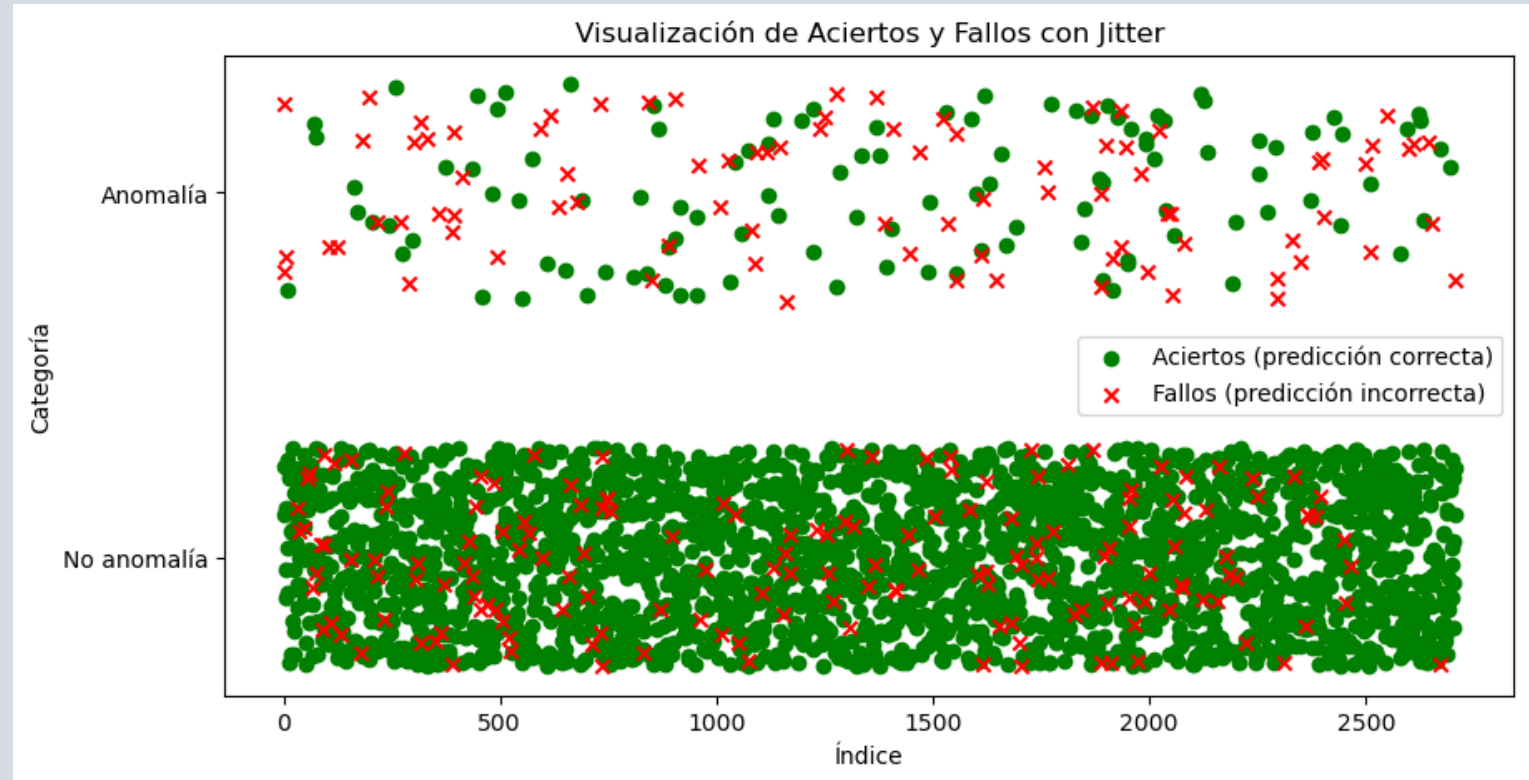


Figura 25. Predicciones (visual)



## RESULTADOS

### 5) Evaluamos el Algoritmo

2510 No  
Anomalías



198  
Anomalías

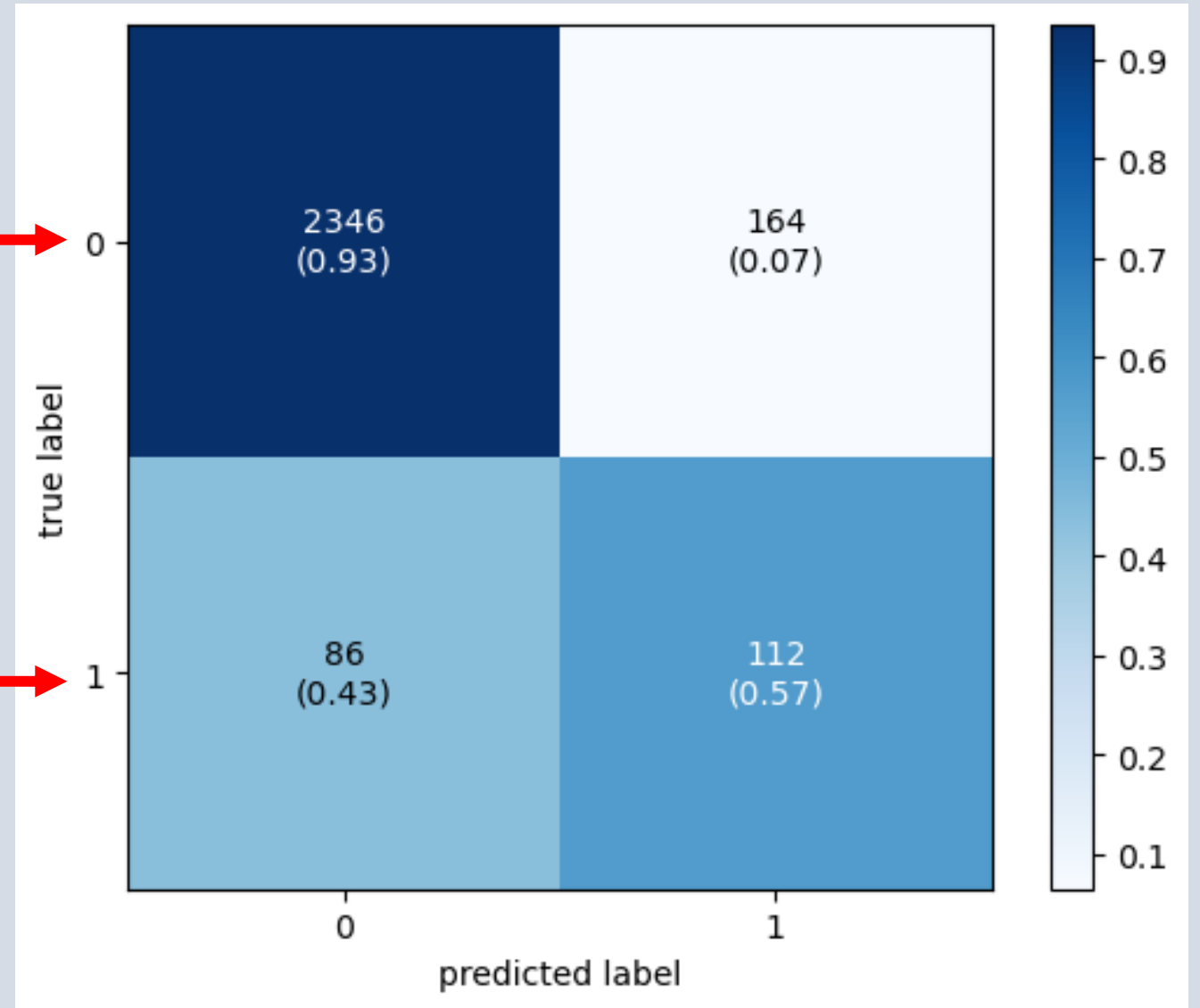


Figura 26. Matriz de Confusión



## RESULTADOS

### 5) Evaluamos el Algoritmo

Podemos ajustar el RECALL del método:  
Se “nos escaparán” menos anomalías,  
pero habrá más Falsos Positivos

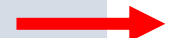
También podemos ajustar mejor el  
detector si predecimos mejor el  
parámetro “Contaminación”.

2510 No  
Anomalías



0

198  
Anomalías



1

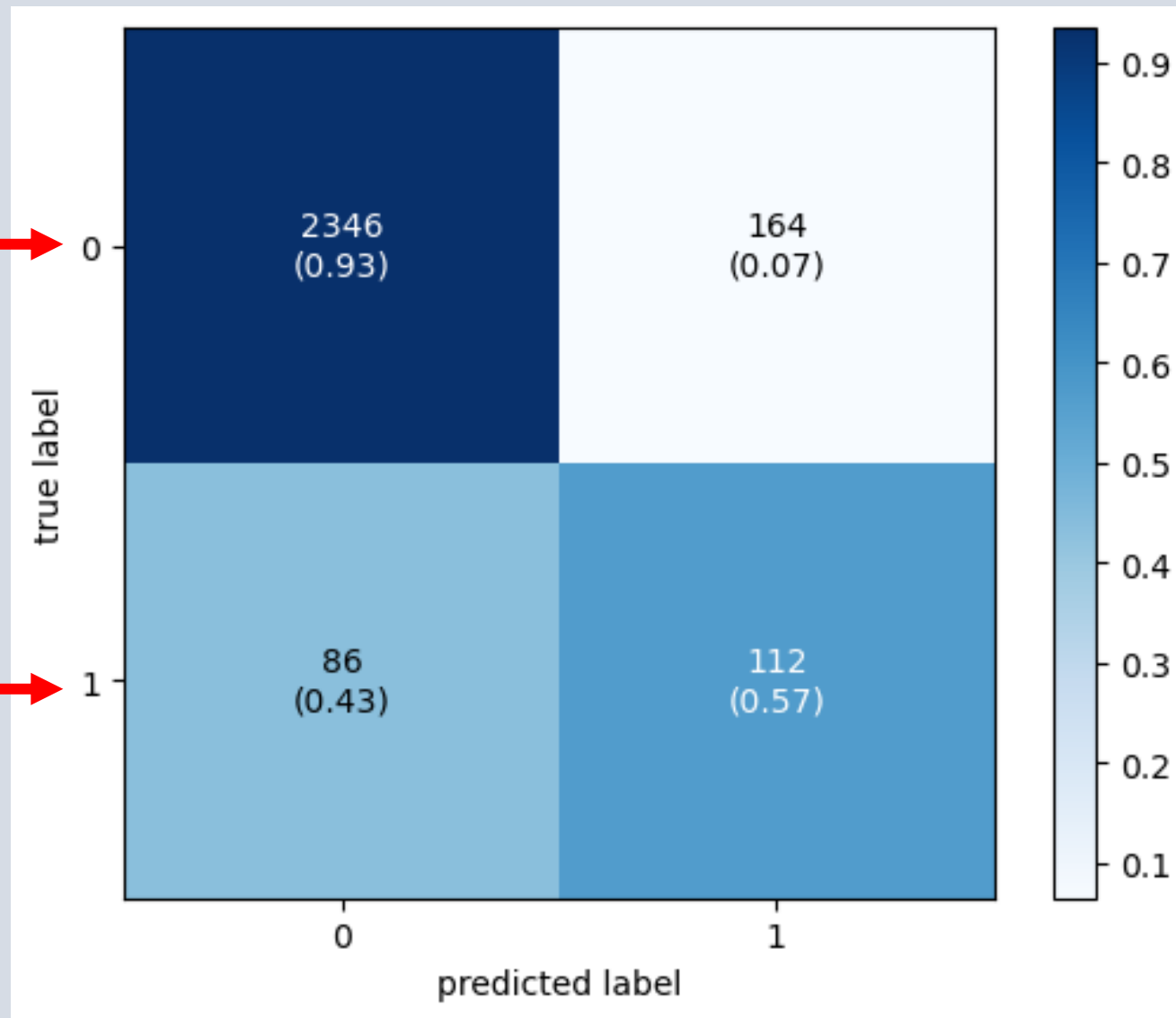


Figura 26. Matriz de Confusión



## RESULTADOS

### 5) Evaluamos el Algoritmo

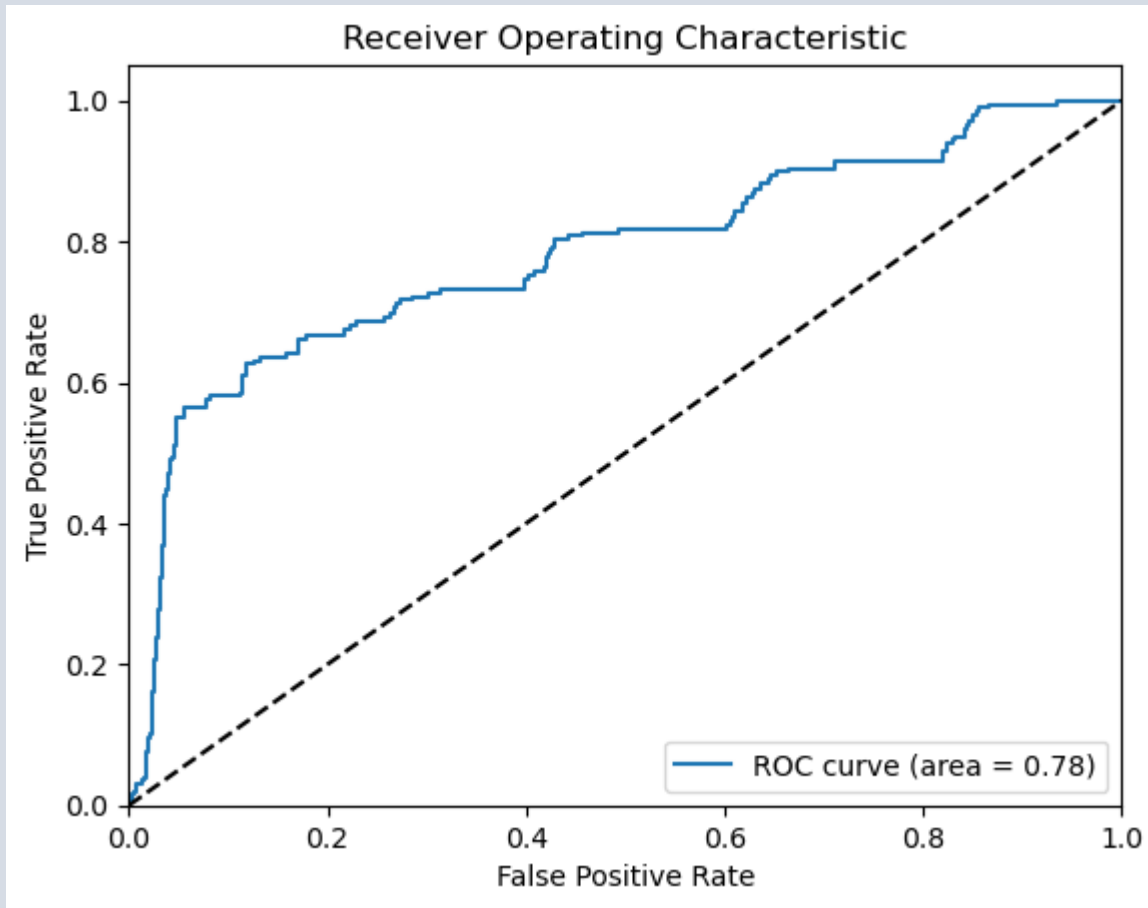


Figura 27. Curva ROC

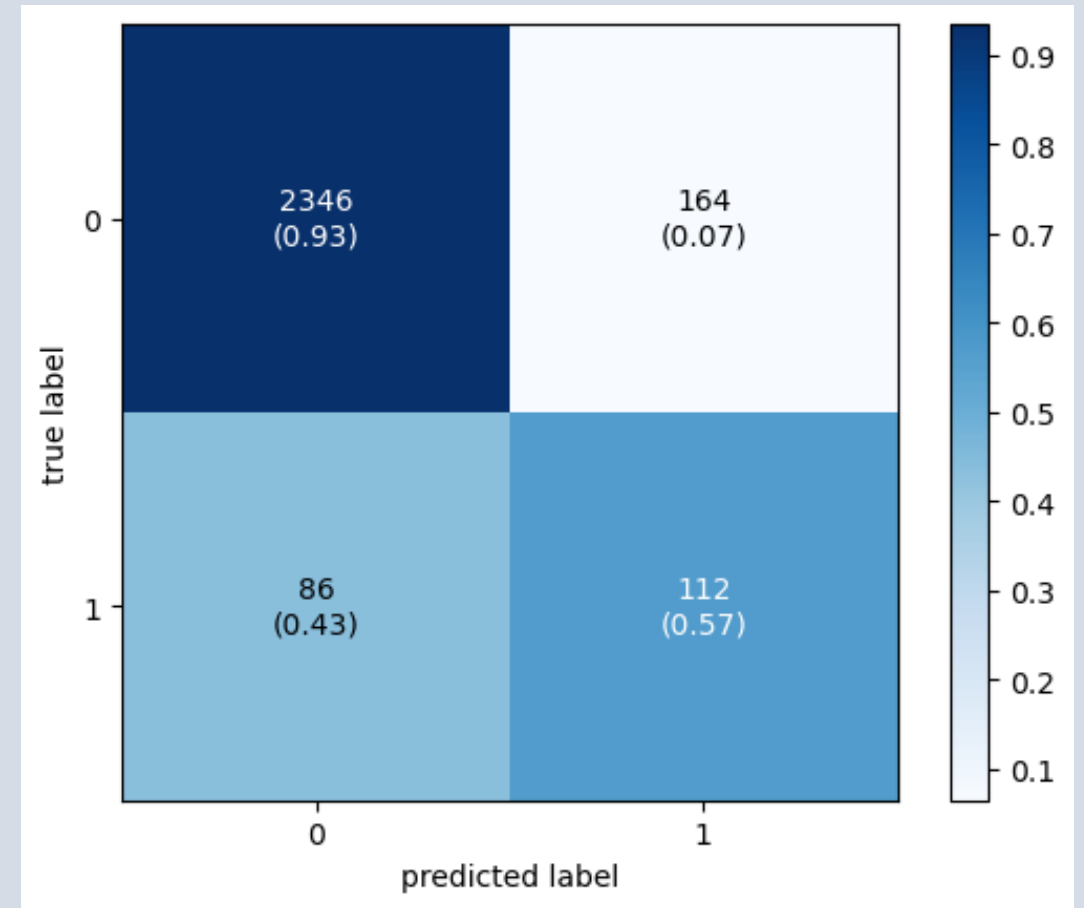


Figura 26. Matriz de Confusión



## Áreas de mejora

**Explicabilidad**

**Heterogeneidad**

**Desbalance de Clases**

**Ruido y Funciones de  
Pérdida**

**Pocos Datos Etiquetados**



## REFERENCIAS

- [Graph Convolutional Networks: Introduction to GNNs](#) (Towardsdatascience).
- [Graph Anomaly Detection with GNN: Current Status and Challenges](#)
- [Basic Understanding of Neural Network Structure](#) (Medium).
- [A Gentle Introduction to Neural Network Series – Part 1](#) (Towardsdatascience).
- [PyTorch Geometric doc](#)
- [Graph Neural Network – Message Passing \(GCN\)](#)
- [Semi-Supervised Classification With GCN](#)



# GRAPH CONVOLUTIONAL NETWORKS

Jesús Martínez, Carlos Sánchez

Aprendizaje Máquina III

Junio 2024