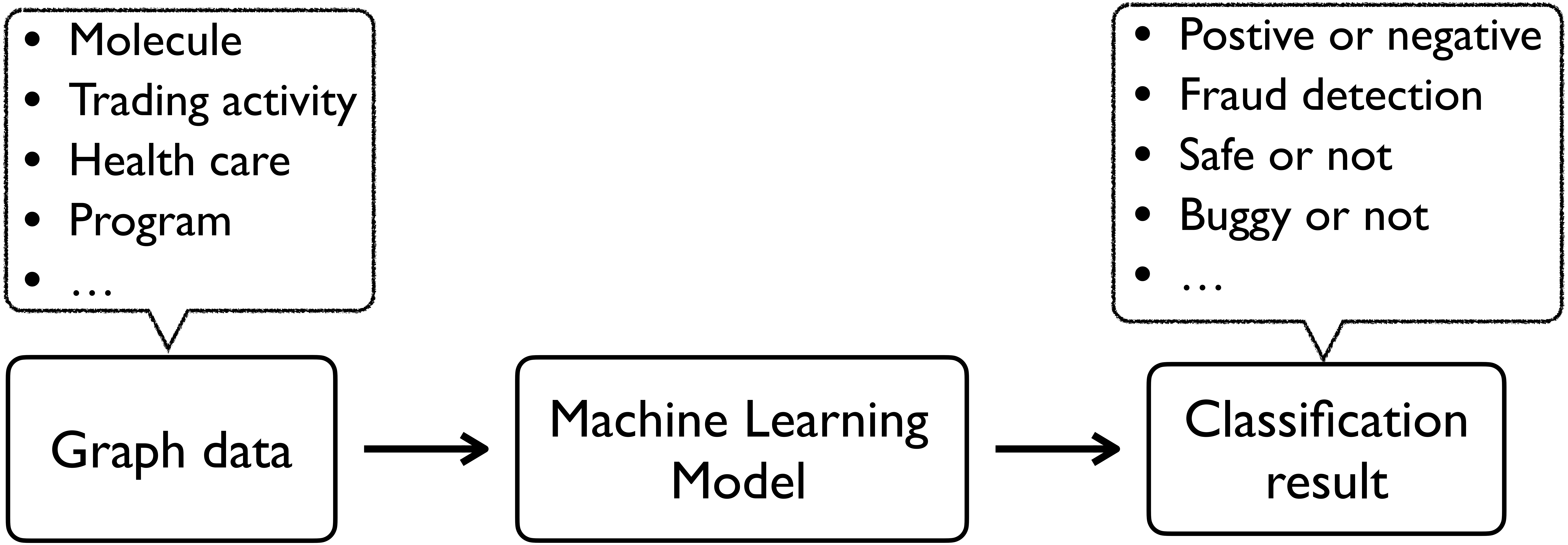
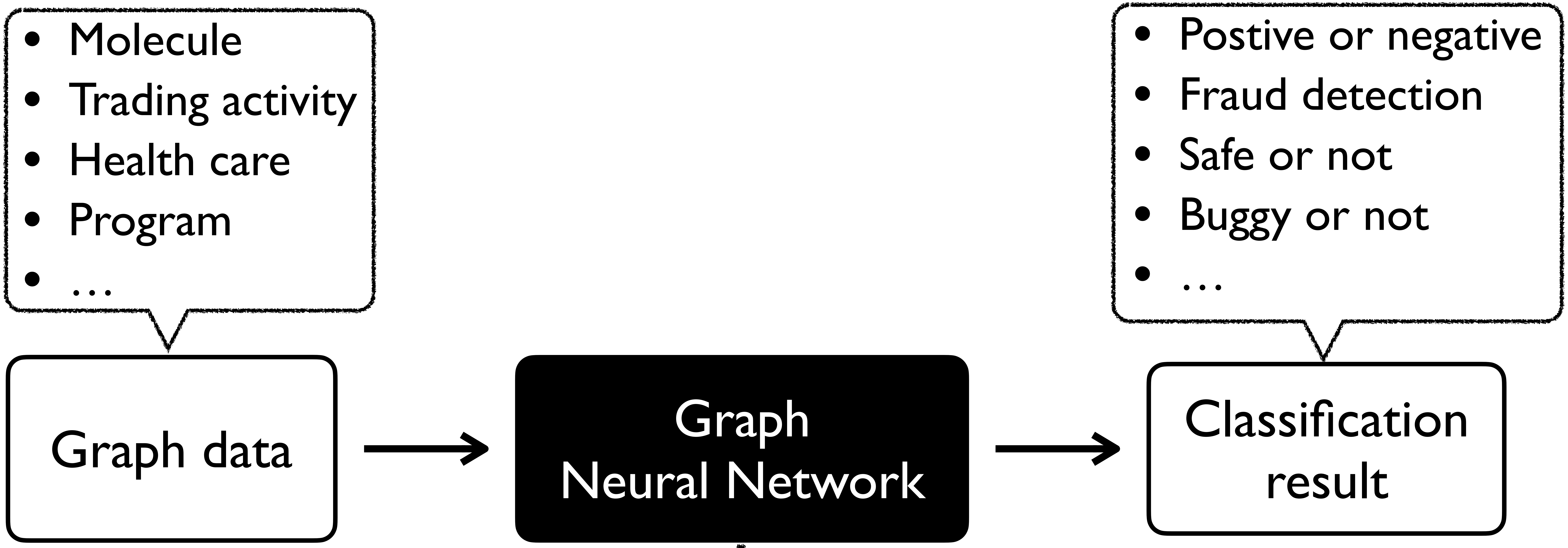
PLDI 2024

PL4XGL: A Programming Language Approach to Explainable Graph Learning

Minseok Jeon

18 September 2025 @ IC637


• Molecule
• Trading activity
• Health care
• Program
• …
Graph data
Machine Learning Model
Classification result
• Postive or negative
• Fraud detection
• Safe or not
• Buggy or not
• …



- GNN is the dominant graph machine learning method

arXiv
https://arxiv.org › cs

Semi-Supervised Classification with Graph Convolutional ...

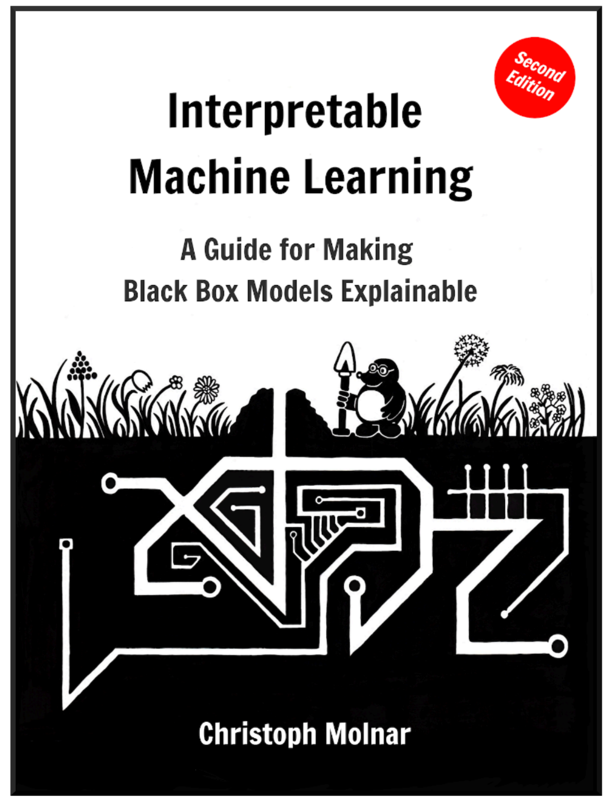
TN Kipf 저술 2016 · 32974회 인용 — Access **Paper**: View a PDF of the **paper** titled Semi-Supervised Classification with Graph Convolutional Networks, by Thomas N. Kipf and 1 other ...

Graph data → **Graph Neural Network** → Classification result

Problem

Does not explain **why**

The value of **why** is growing fast



> A correct prediction only partially solves your problem. The model must also explain **why**.
>
> \- Molnar [2022]

Graph data → Graph Neural Network → Classification result

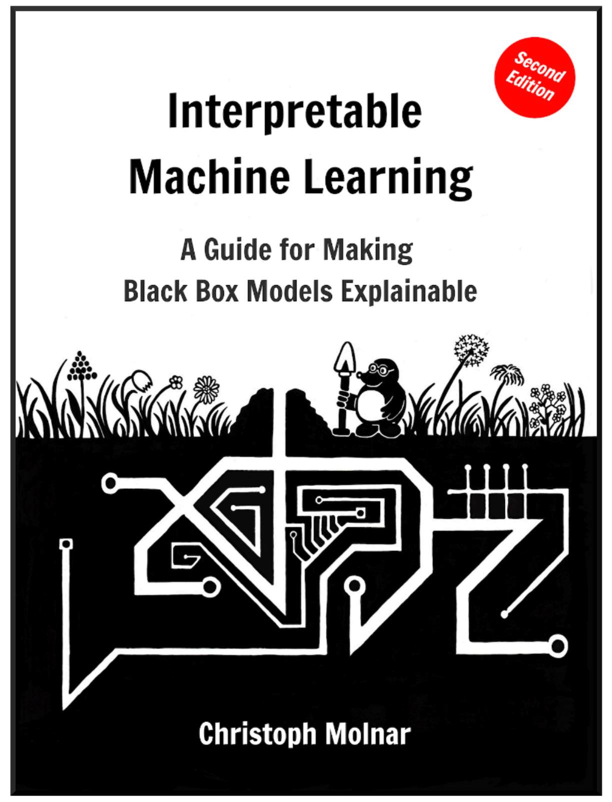
- Health care
- Trading activity
- …

Problem

Does not explain **why**

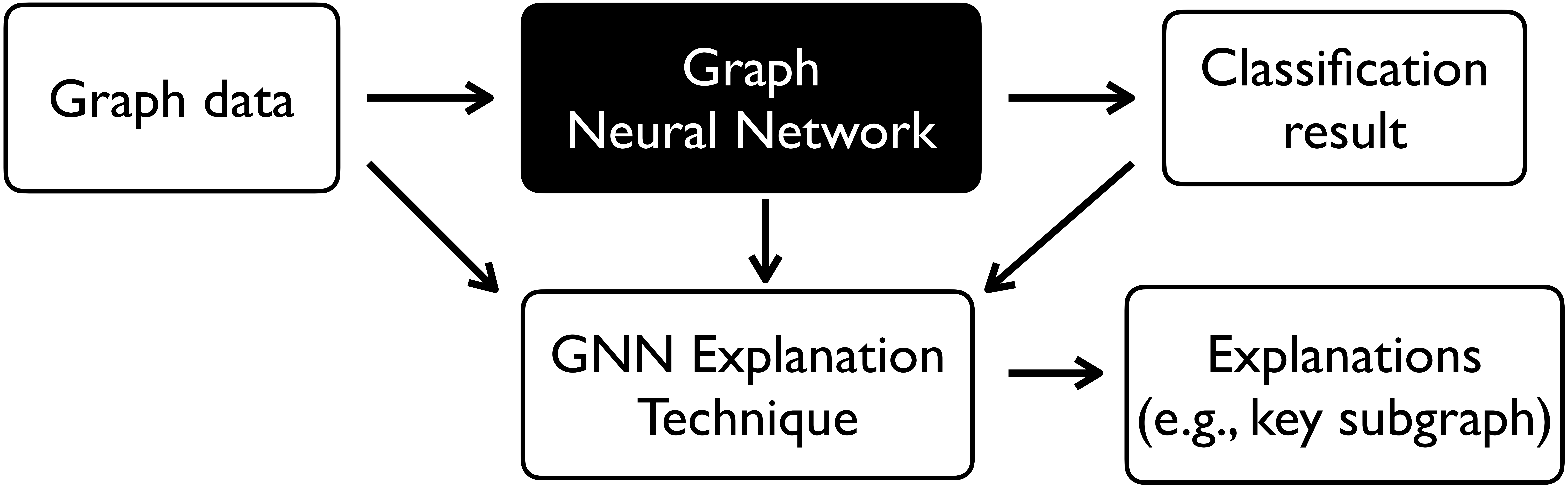
- Safe or not
- Fraud or not
- …

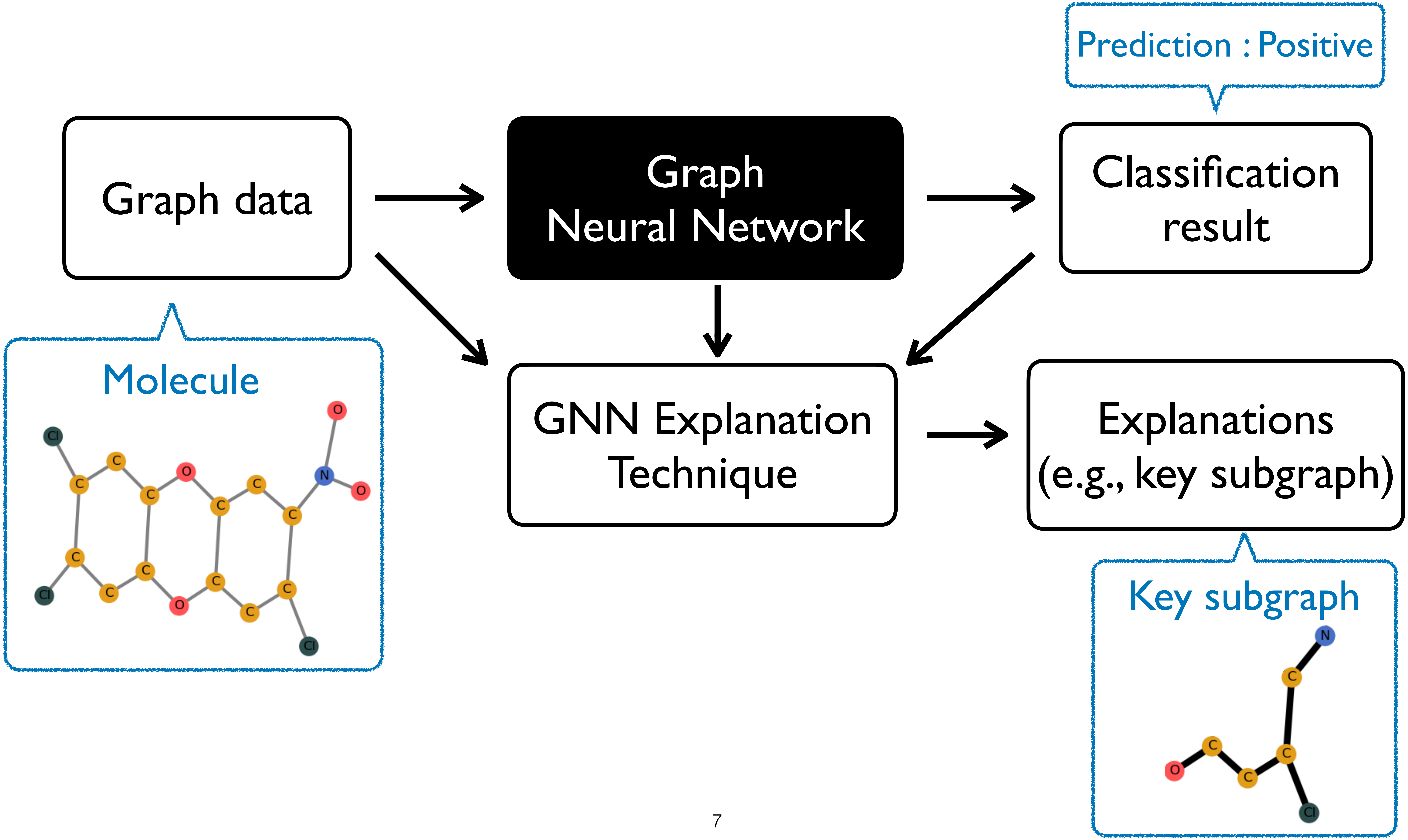
The value of **why** is growing fast

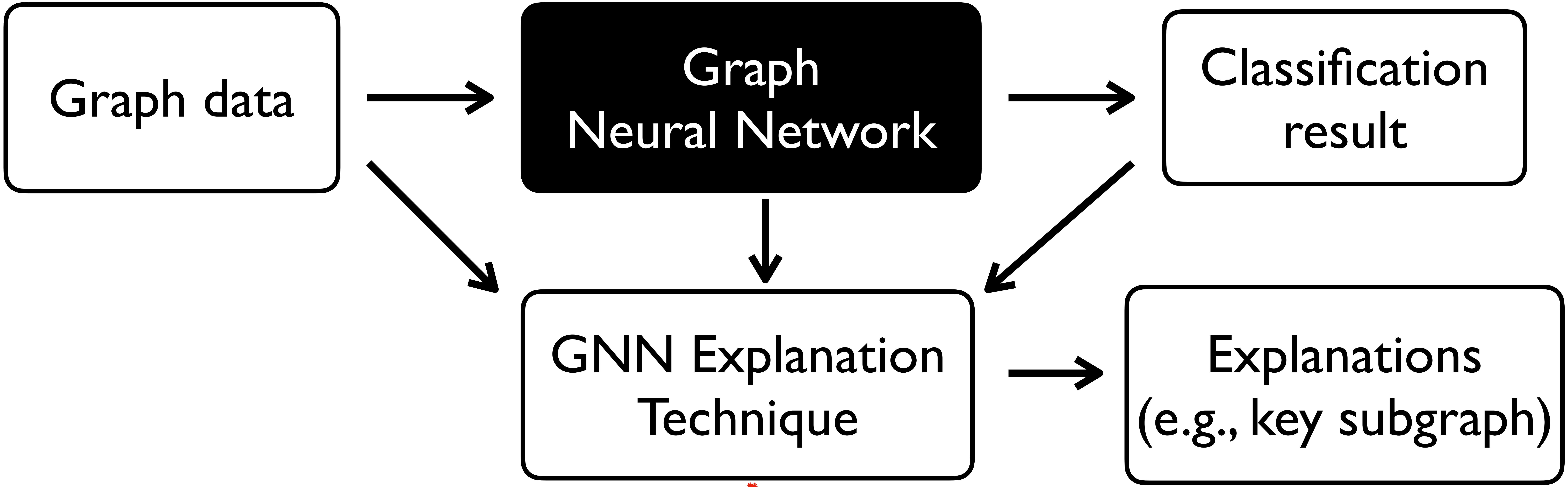


A correct prediction only partially solves your problem. The model must also explain **why**.

- Molnar [2022]


Graph data
Graph
Neural Network
Classification
result
GNN Explanation
Technique
Explanations
(e.g., key subgraph)


Prediction : Positive
Graph data
Graph
Neural Network
Classification
result
Molecule
GNN Explanation
Technique
Explanations
(e.g., key subgraph)
Key subgraph



Two key limitations

- Additional (expensive) explanation cost is required
- The explanations are not guaranteed to be correct

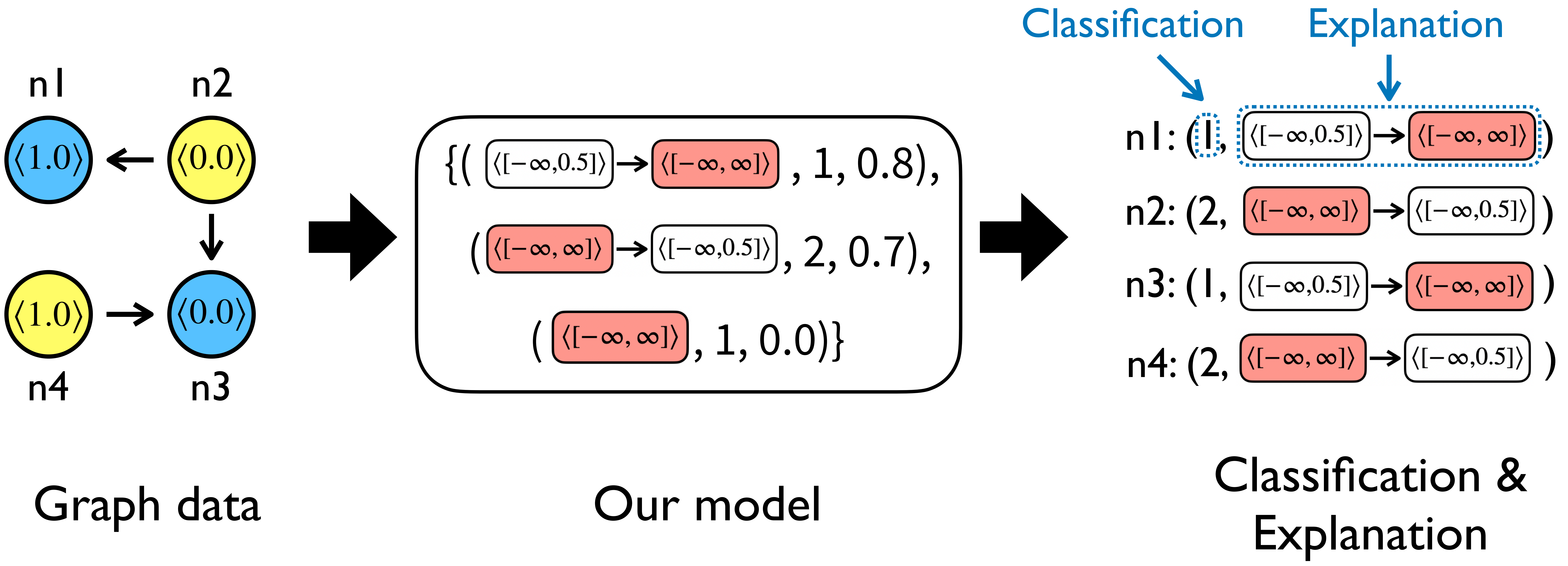
PLDI 2024

- PL4XGL: inherently explainable graph machine learning method

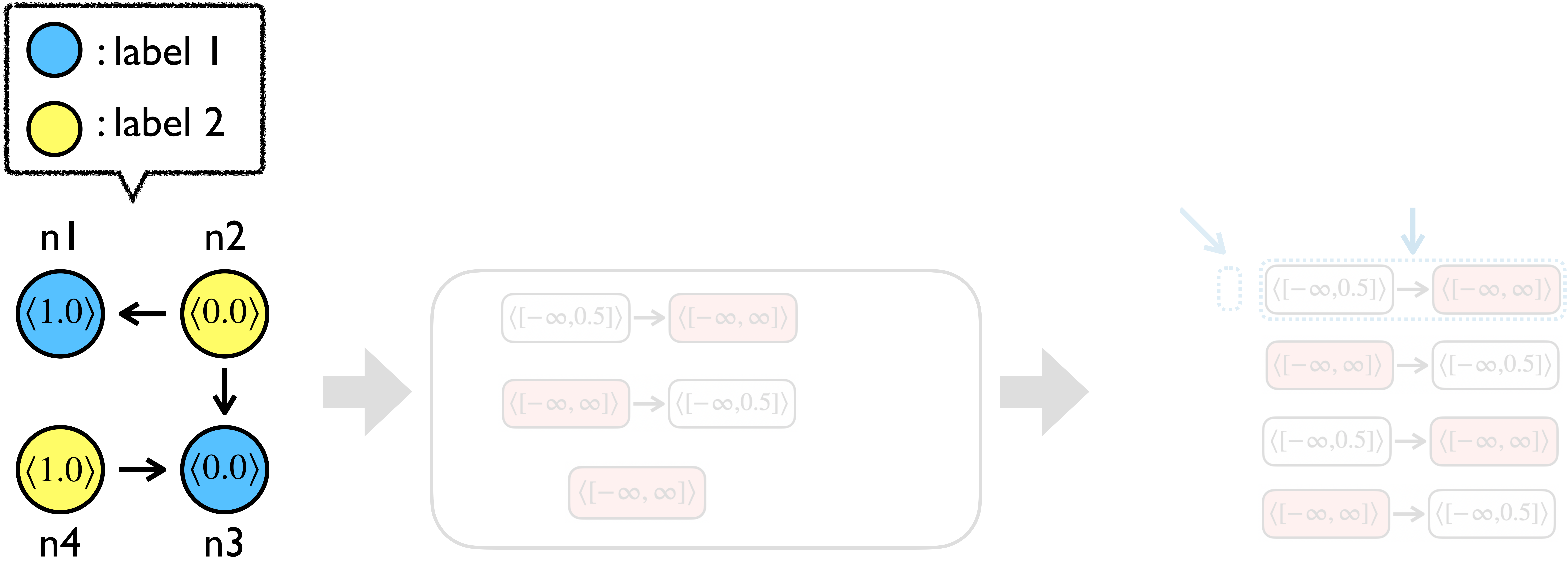
Graph data → PL4XGL → Classification result & correct explanation

Graph Description Language (GDL)

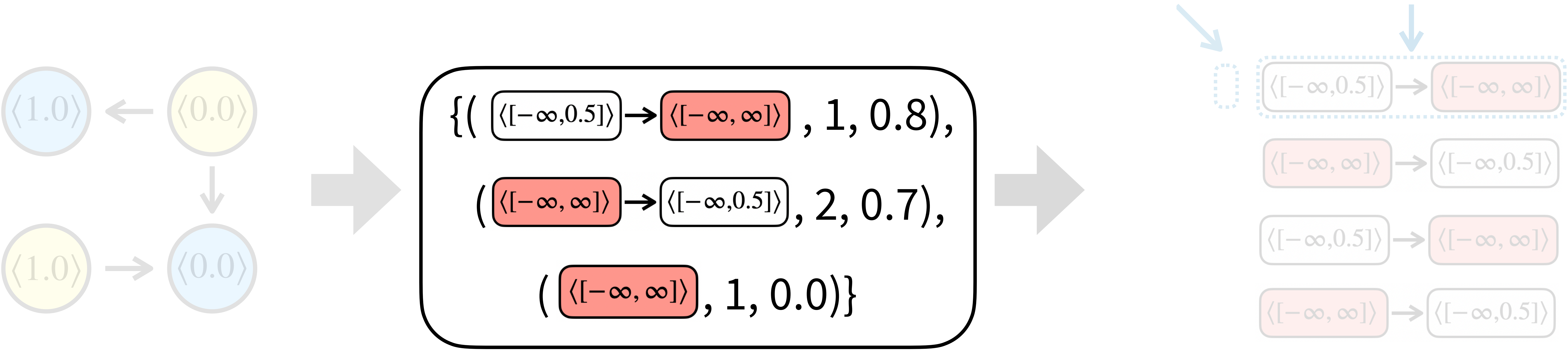
Programs	$P ::= \overline{\delta}$ **target** t	$\in$	$\mathbb{P}$	$= \mathbb{D}^* \times \mathbb{T}$
Descriptions	$\delta ::= \delta_V \mid \delta_E$	$\in$	$\mathbb{D}$	$= \mathbb{D}_V \uplus \mathbb{D}_E$
Node Descriptions	$\delta_V ::=$ **node** x $<\overline{\phi}>^?$	$\in$	$\mathbb{D}_V$	$= \mathbb{X} \times \Phi^d$
Edge Descriptions	$\delta_E ::=$ **edge** (x, x) $<\overline{\phi}>^?$	$\in$	$\mathbb{D}_E$	$= \mathbb{X} \times \mathbb{X} \times \Phi^c$
Target Symbols	$t ::=$ **node** x $\mid$ **edge** (x, x) $\mid$ **graph**	$\in$	$\mathbb{T}$	$= \mathbb{X} \uplus (\mathbb{X} \times \mathbb{X}) \uplus \{\epsilon\}$
Intervals	$\phi ::= [n^?, n^?]$	$\in$	Φ	$= (\mathbb{R} \uplus \{-\infty\}) \times (\mathbb{R} \uplus \{\infty\})$
Real Numbers	$n ::= 0.2 \mid 0.7 \mid 6 \mid -8 \ \ldots$	$\in$	$\mathbb{R}$	
Variables	$x ::= \mathtt{x} \mid \mathtt{y} \mid \mathtt{z} \mid \ldots$	$\in$	$\mathbb{X}$	



n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
Graph data
{(⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩ , 1, 0.8),
(⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩ , 2, 0.7),
(⟨[−∞, ∞]⟩ , 1, 0.0)}
Our model
Classification
Explanation
n1: (1, ⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
n2: (2, ⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩)
n3: (1, ⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
n4: (2, ⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩)
Classification &
Explanation



Graph data

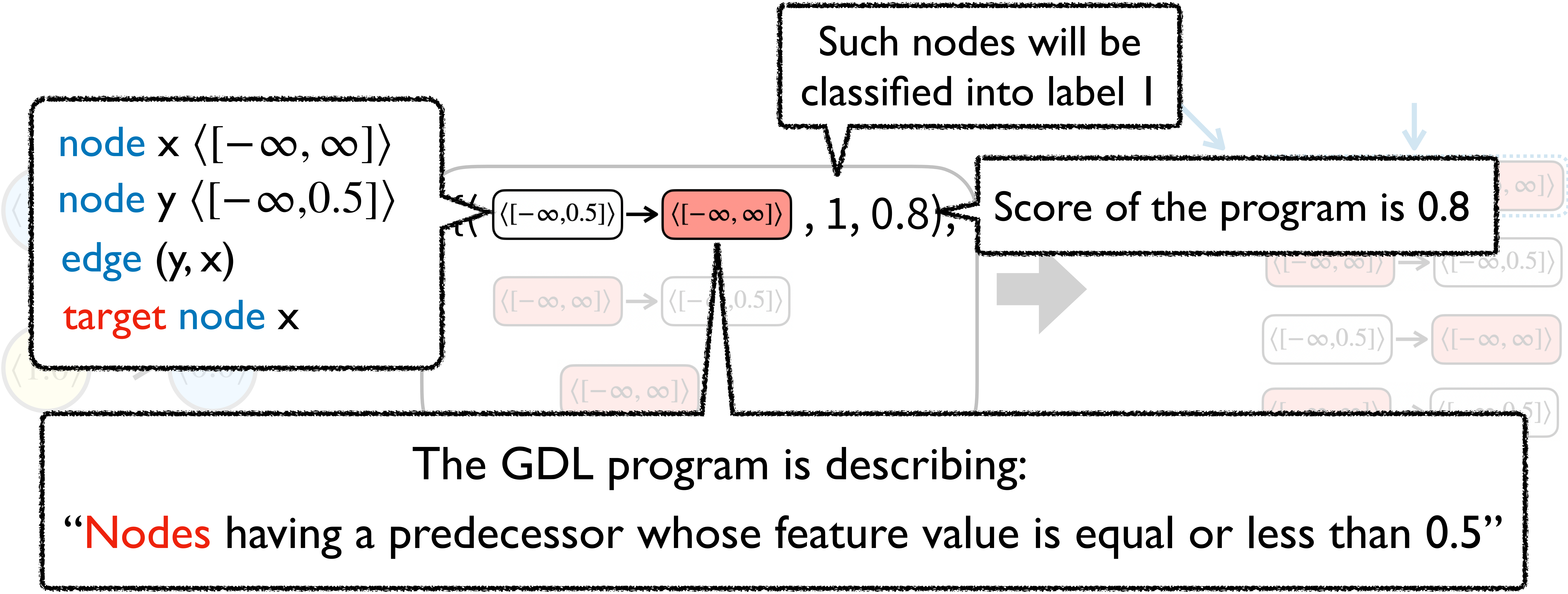


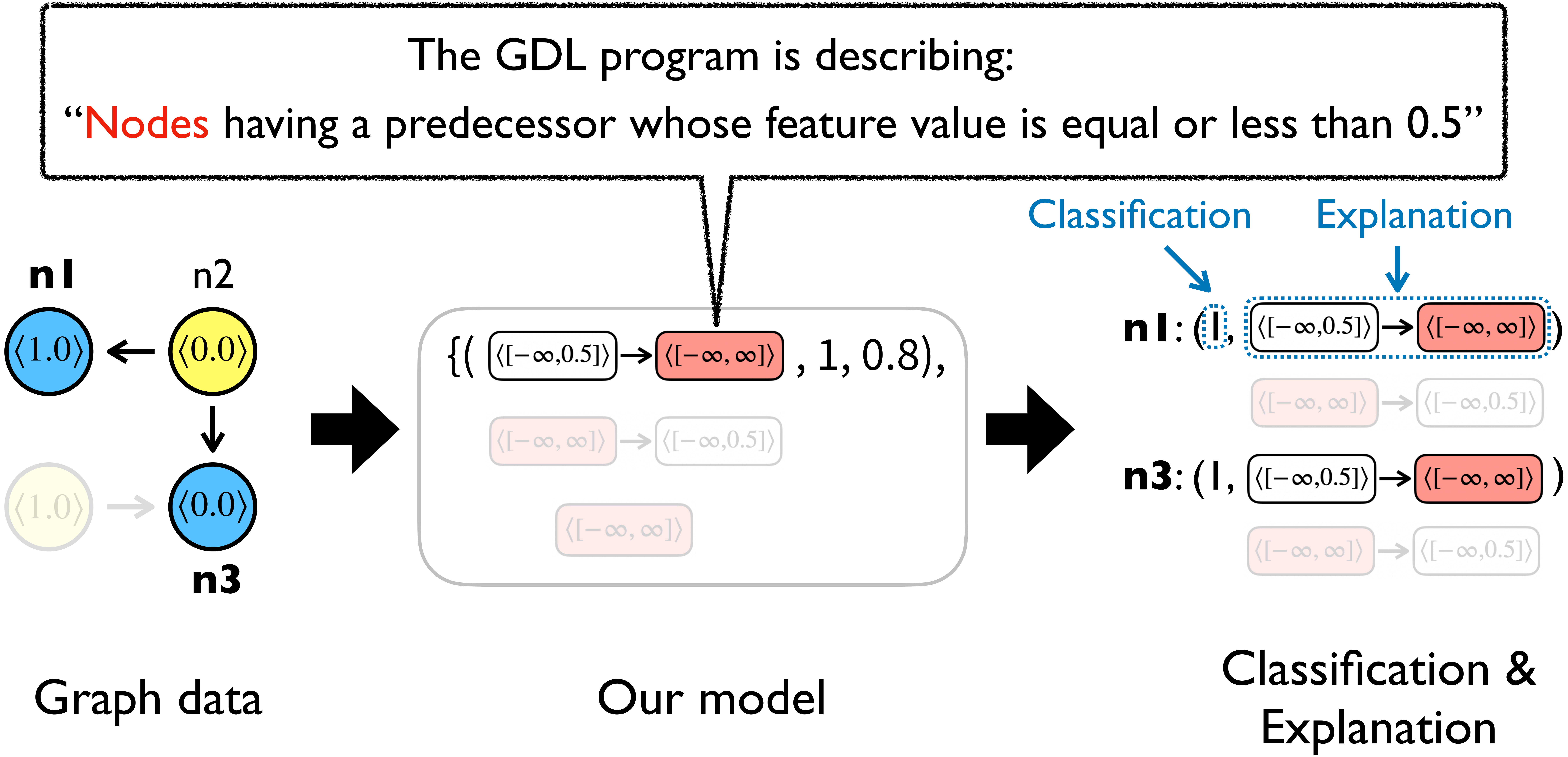
Our model

node x $\langle[-\infty, \infty]\rangle$
node y $\langle[-\infty, 0.5]\rangle$
edge (y, x)
target node x

The GDL program is describing:

"Nodes having a predecessor whose feature value is equal or less than 0.5"

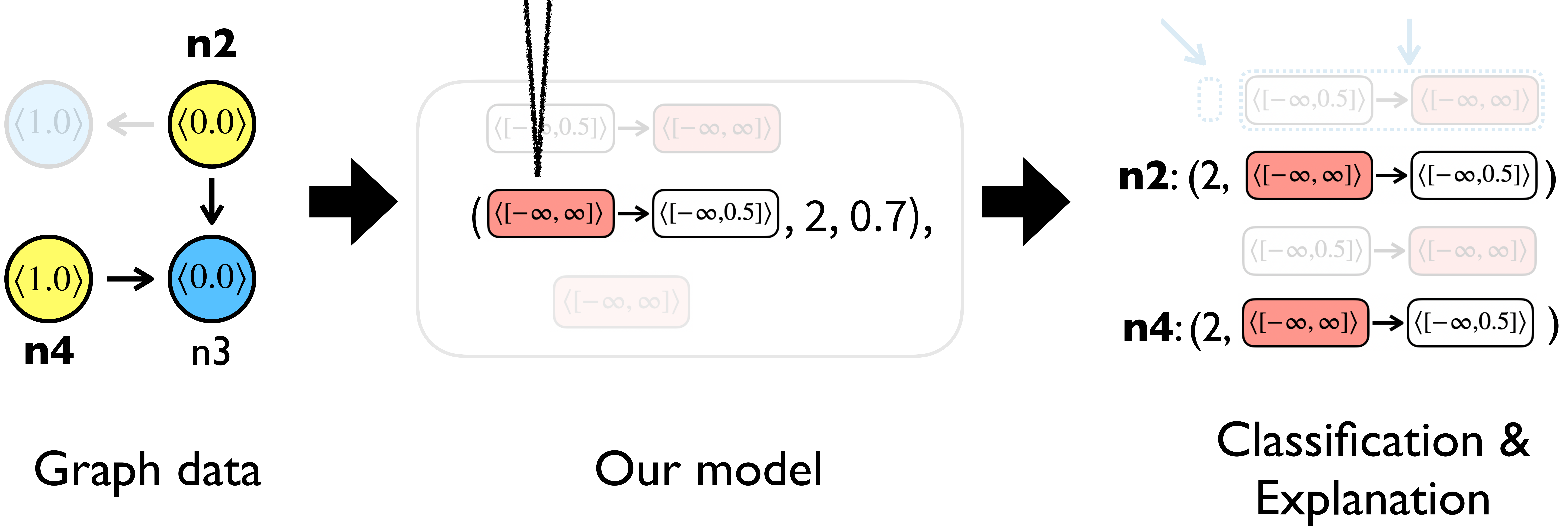

Such nodes will be classified into label 1
node x ⟨[−∞, ∞]⟩
node y ⟨[−∞,0.5]⟩
edge (y, x)
target node x
⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩ , 1, 0.8);
Score of the program is 0.8
The GDL program is describing:
“Nodes having a predecessor whose feature value is equal or less than 0.5”

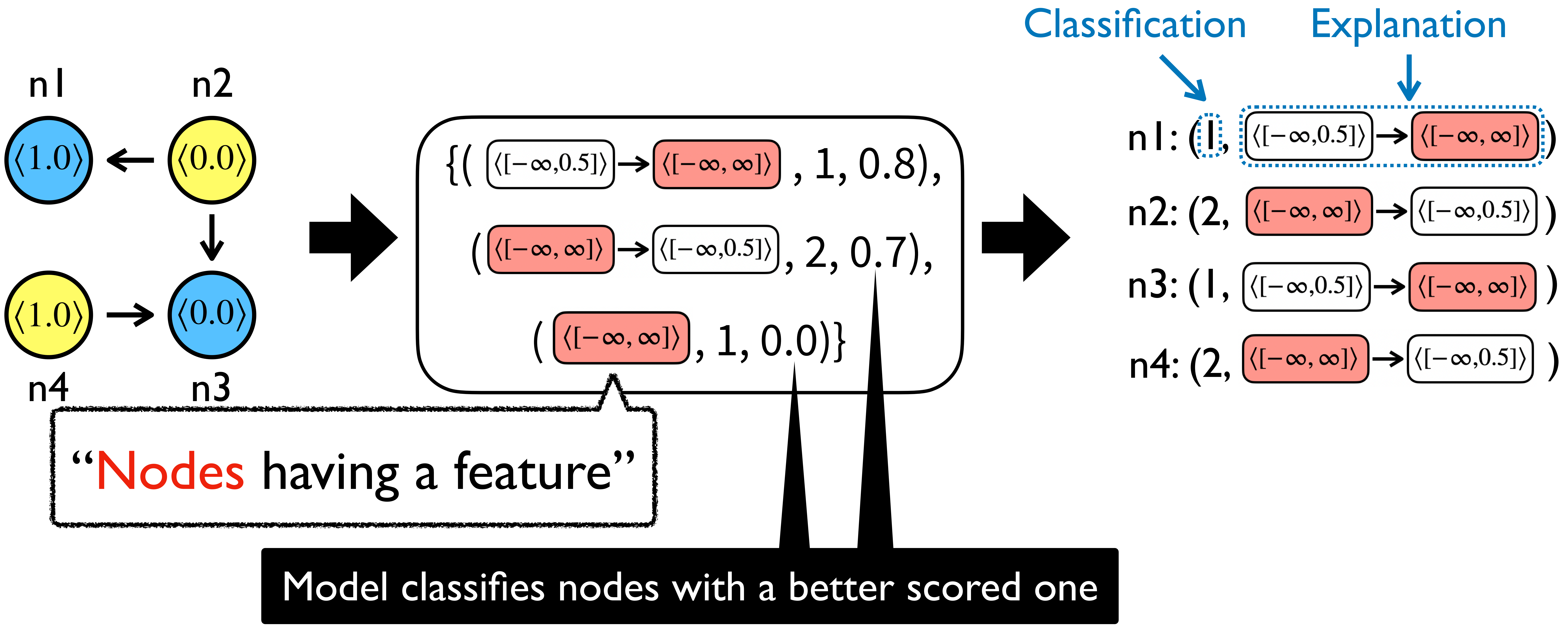


The GDL program is describing:
"Nodes having a predecessor whose feature value is equal or less than 0.5"
n1
n2
n3
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
{(⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩ , 1, 0.8),
⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩
⟨[−∞, ∞]⟩
Classification
Explanation
n1: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩
n3: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩
Graph data
Our model
Classification &
Explanation

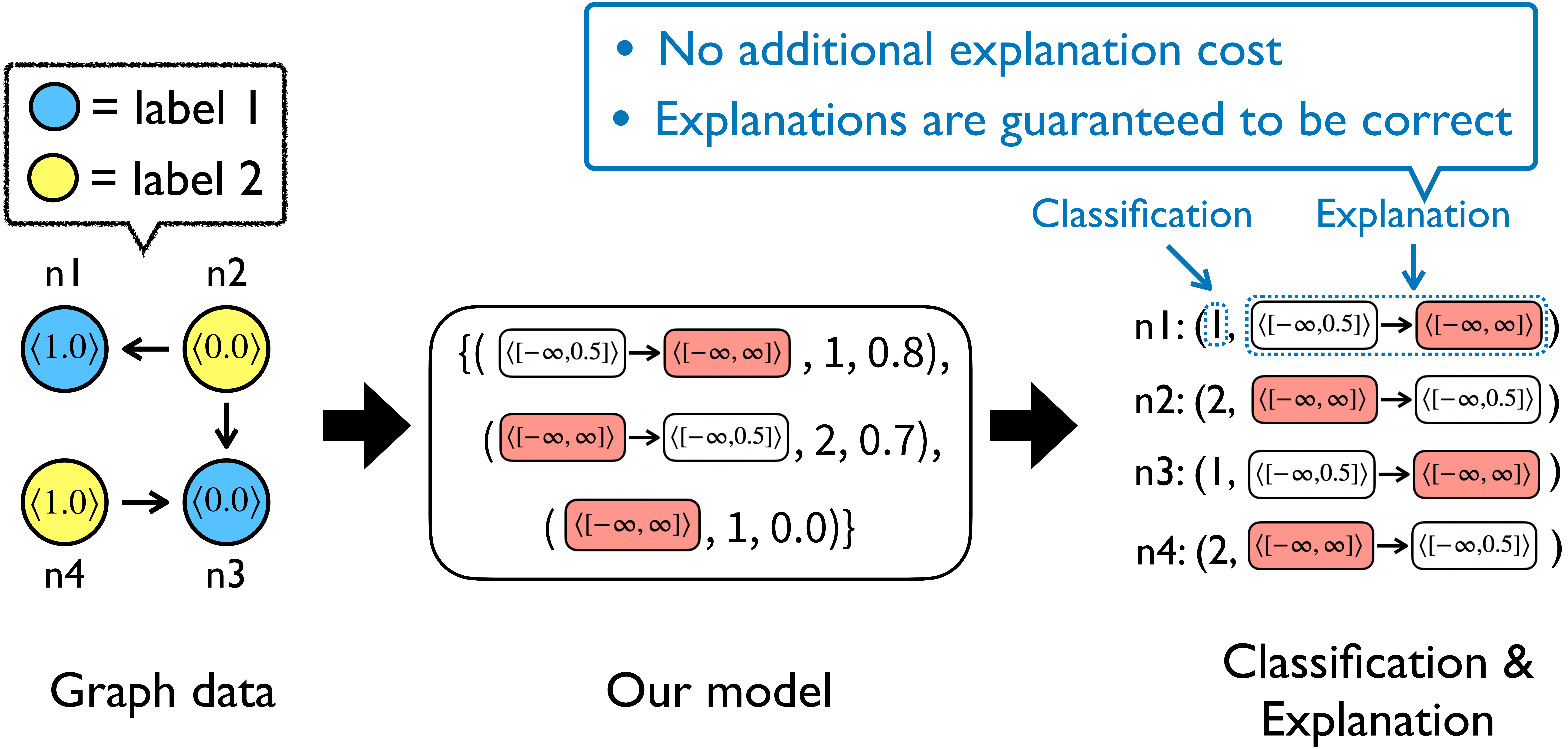
The GDL program is describing:

"Nodes having a successor whose feature value is equal or less than 0.5"

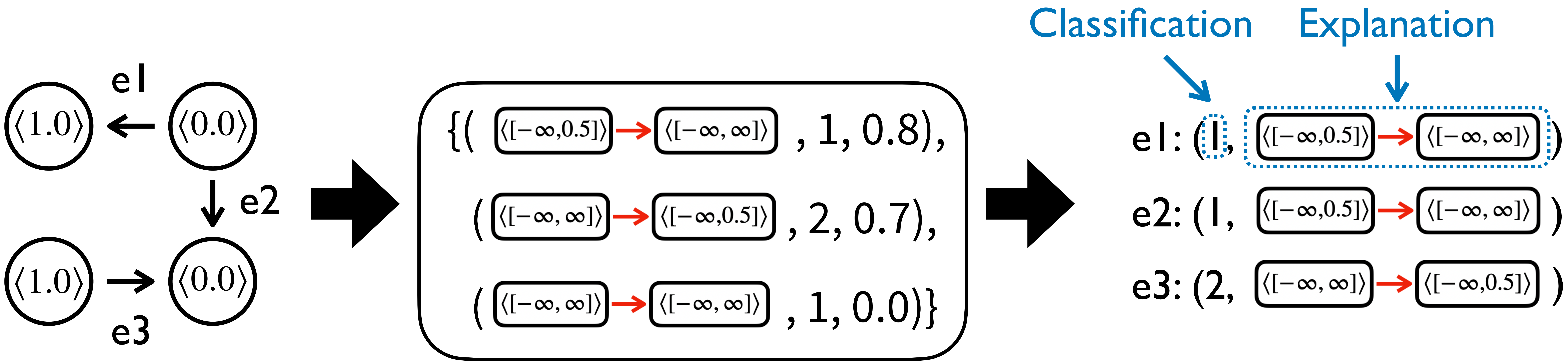




Classification
Explanation
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
{(⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩ , 1, 0.8),
(⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩, 2, 0.7),
(⟨[−∞, ∞]⟩, 1, 0.0)}
n1: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n2: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
n3: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n4: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
“Nodes having a feature”
Model classifies nodes with a better scored one



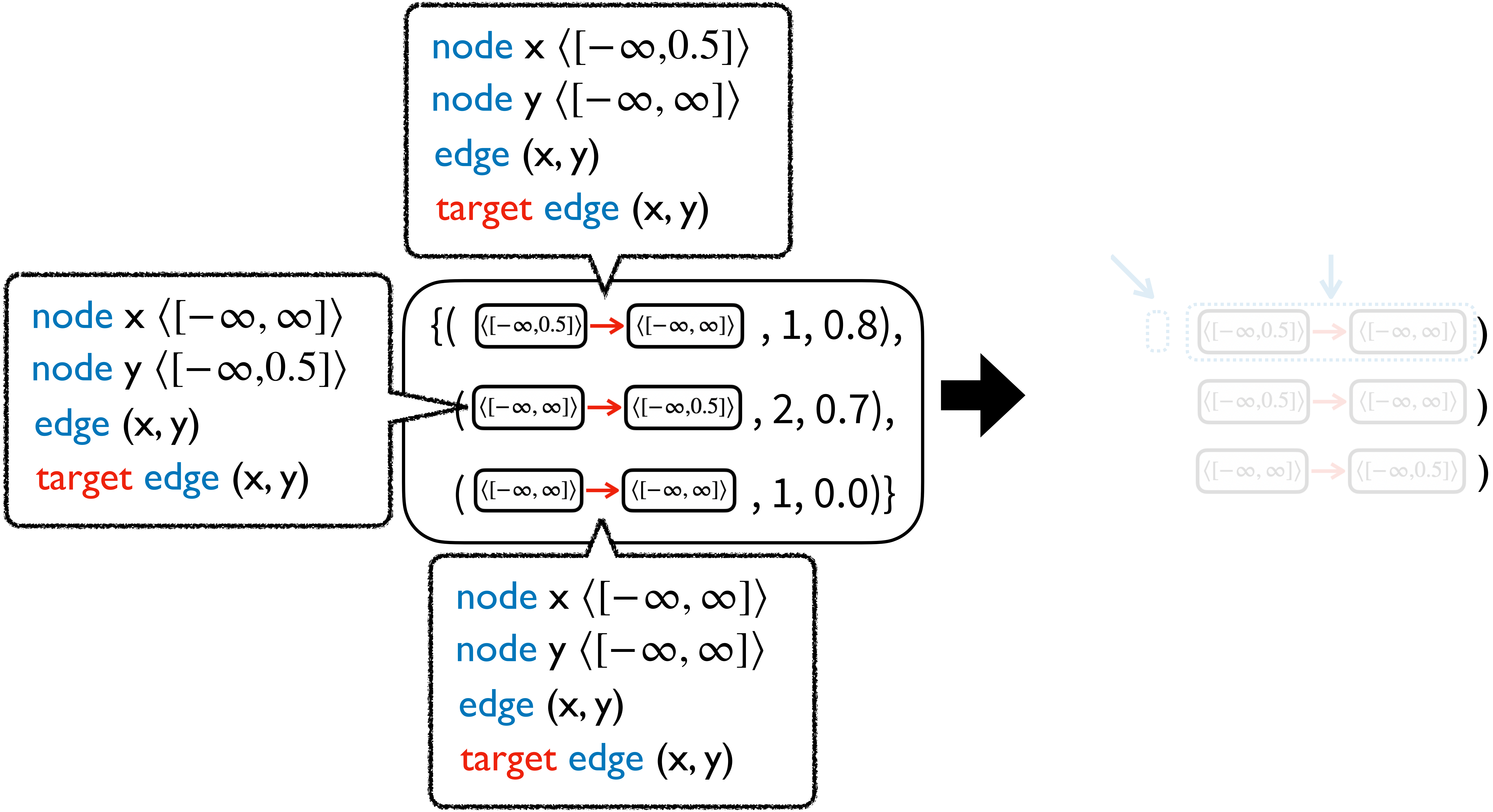
No additional explanation cost
Explanations are guaranteed to be correct
= label 1
= label 2
n1
n2
n3
n4
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
{(⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩, 1, 0.8),
(⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩, 2, 0.7),
(⟨[−∞, ∞]⟩, 1, 0.0)}
Classification
Explanation
n1: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n2: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
n3: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n4: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
Graph data
Our model
Classification & Explanation



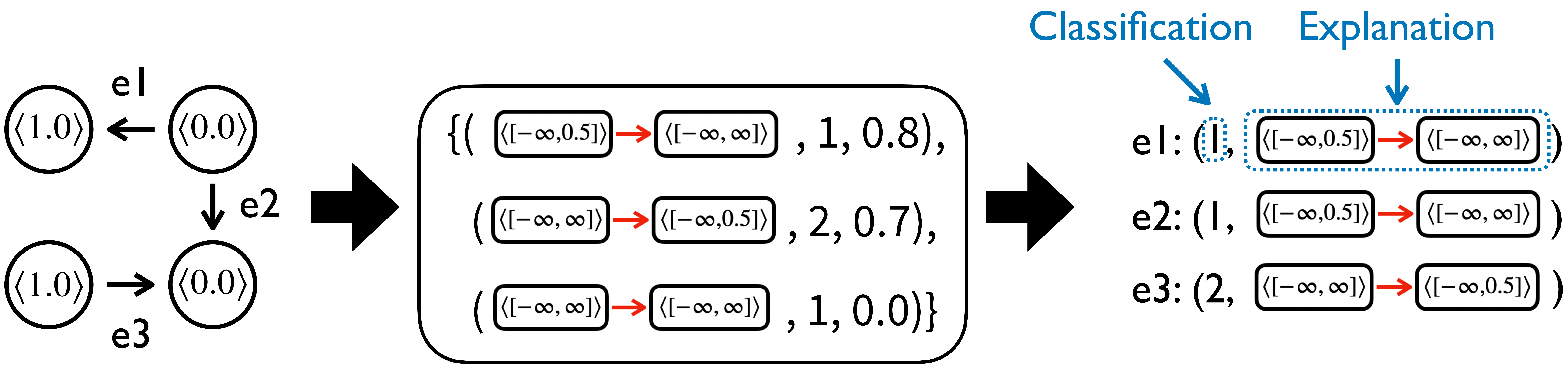
Graph data

Our model
(**edge** classification)

Classification &
Explanation



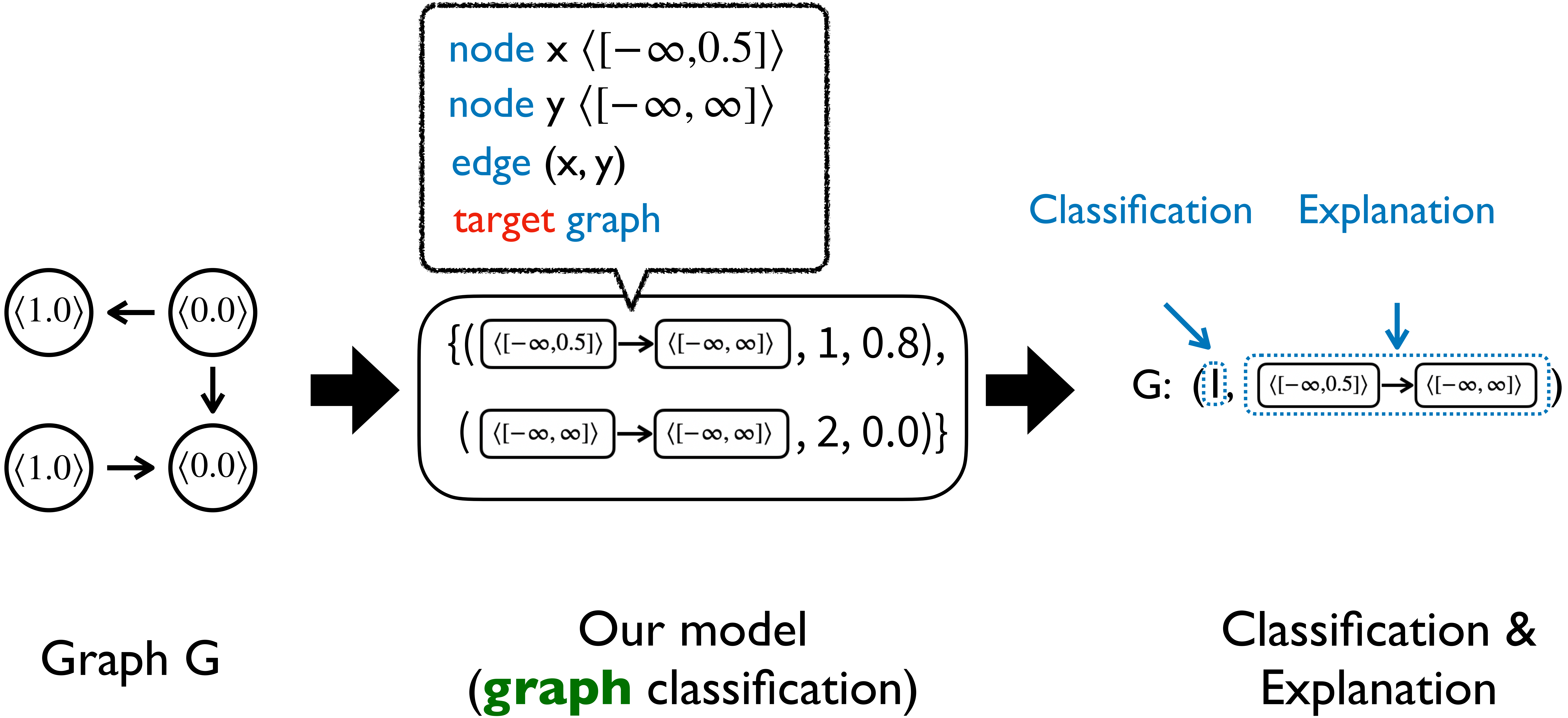
node x ⟨[−∞,0.5]⟩
node y ⟨[−∞, ∞]⟩
edge (x, y)
target edge (x, y)
node x ⟨[−∞, ∞]⟩
node y ⟨[−∞,0.5]⟩
edge (x, y)
target edge (x, y)
{(⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩ , 1, 0.8),
(⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩ , 2, 0.7),
(⟨[−∞, ∞]⟩ → ⟨[−∞, ∞]⟩ , 1, 0.0)}
node x ⟨[−∞, ∞]⟩
node y ⟨[−∞, ∞]⟩
edge (x, y)
target edge (x, y)
⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩)



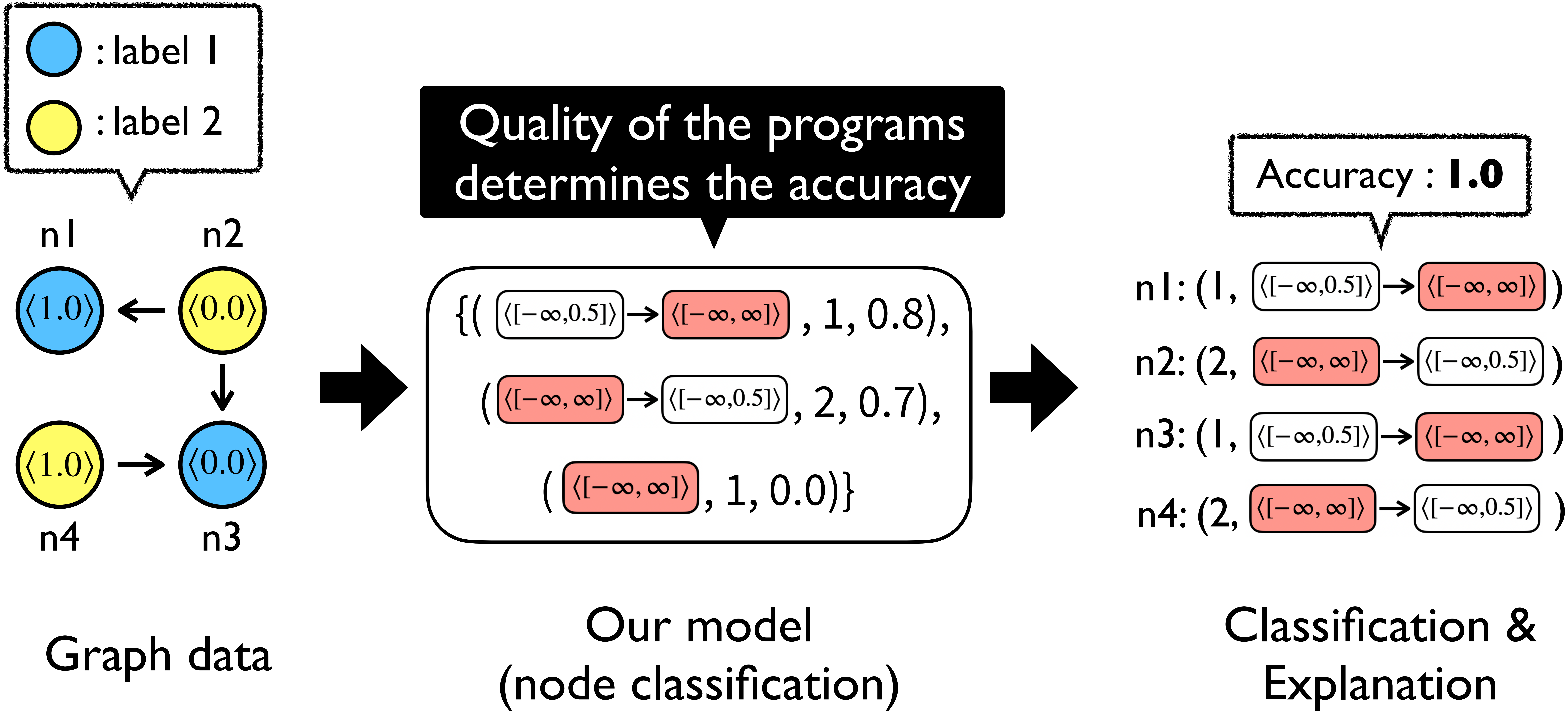
Graph data

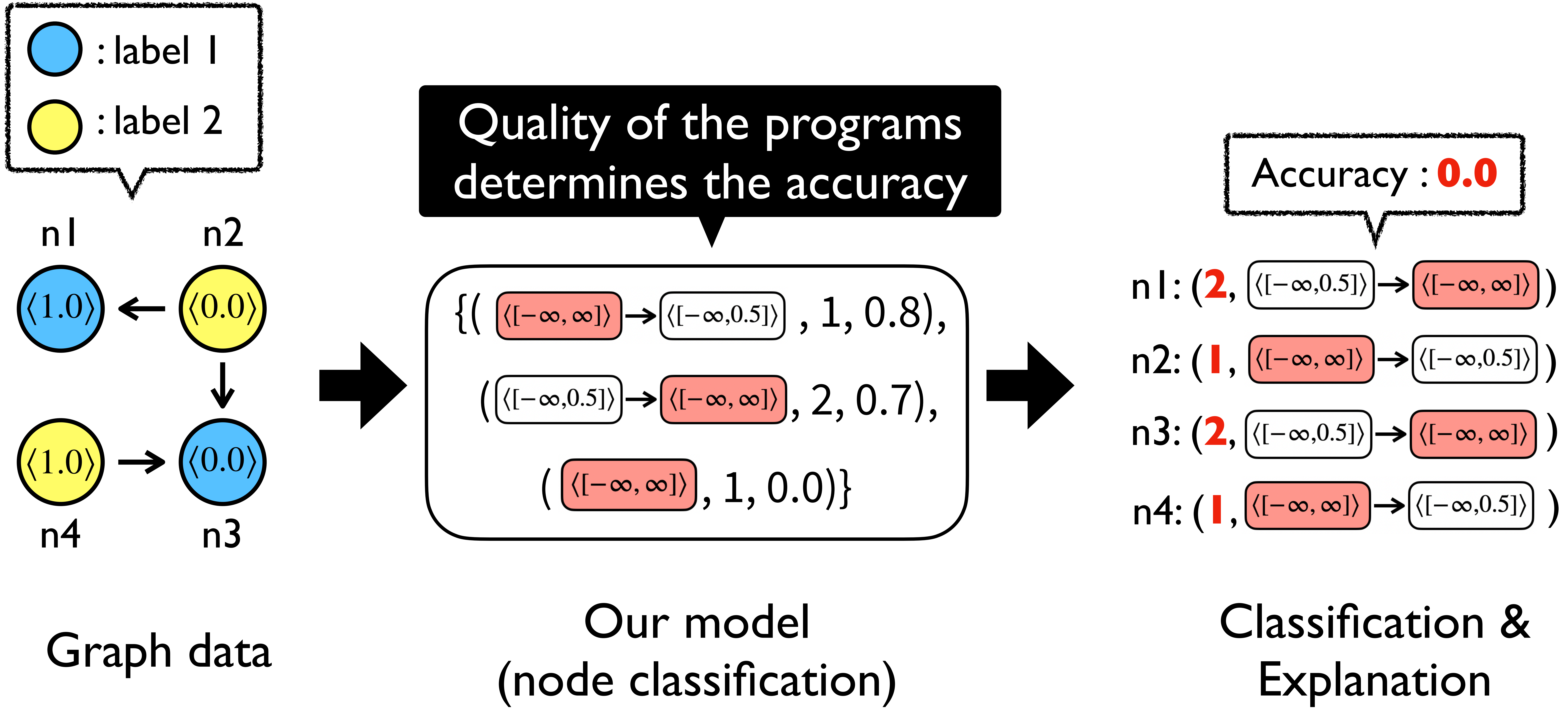
Our model
(**edge** classification)

Classification &
Explanation



node x ⟨[−∞,0.5]⟩
node y ⟨[−∞, ∞]⟩
edge (x, y)
target graph
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
{(⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩, 1, 0.8),
(⟨[−∞, ∞]⟩ → ⟨[−∞, ∞]⟩, 2, 0.0)}
Classification
Explanation
G: (1, ⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
Graph G
Our model
(graph classification)
Classification &
Explanation


: label 1
: label 2
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
Graph data
Quality of the programs determines the accuracy
{(⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩, 1, 0.8),
(⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩, 2, 0.7),
(⟨[−∞, ∞]⟩, 1, 0.0)}
Our model
(node classification)
Accuracy : **1.0**
n1: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n2: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
n3: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n4: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
Classification &
Explanation



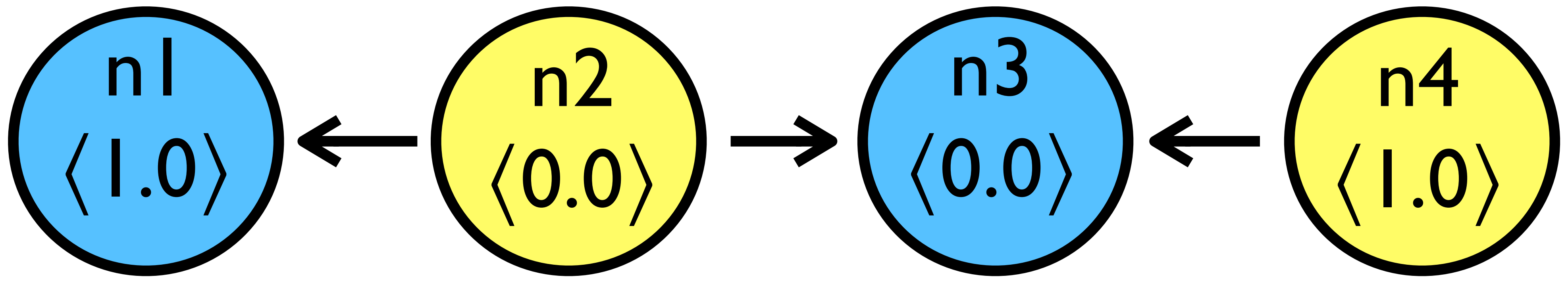
: label 1
: label 2
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
Graph data
Quality of the programs determines the accuracy
{(⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩, 1, 0.8),
(⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩, 2, 0.7),
(⟨[−∞, ∞]⟩, 1, 0.0)}
Our model
(node classification)
Accuracy : 0.0
n1: (2, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n2: (1, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
n3: (2, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n4: (1, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
Classification &
Explanation


Training data
Learning algorithm
Learning objective:
Learn high-quality GDL programs
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
{(⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩ , 1, 0.8),
(⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩ , 2, 0.7),
(⟨[−∞, ∞]⟩ , 1, 0.0)}
n1: (1, ⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
n2: (2, ⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩)
n3: (1, ⟨[−∞,0.5]⟩ → ⟨[−∞, ∞]⟩)
n4: (2, ⟨[−∞, ∞]⟩ → ⟨[−∞,0.5]⟩)
Graph data
Our model
Classification & Explanation

Training data
Top-down learning algorithm
Bottom-up learning algorithm
Learning algorithm
Learning objective:
Learn high-quality GDL programs
n1
n2
⟨1.0⟩
⟨0.0⟩
⟨1.0⟩
⟨0.0⟩
n4
n3
{(⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩ , 1, 0.8),
(⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩, 2, 0.7),
(⟨[−∞, ∞]⟩, 1, 0.0)}
n1: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n2: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
n3: (1, ⟨[−∞,0.5]⟩→⟨[−∞, ∞]⟩)
n4: (2, ⟨[−∞, ∞]⟩→⟨[−∞,0.5]⟩)
Graph data
Our model
Classification & Explanation

GDL Program Learning Algorithm

Training graph :



Target node = n1

: label 1

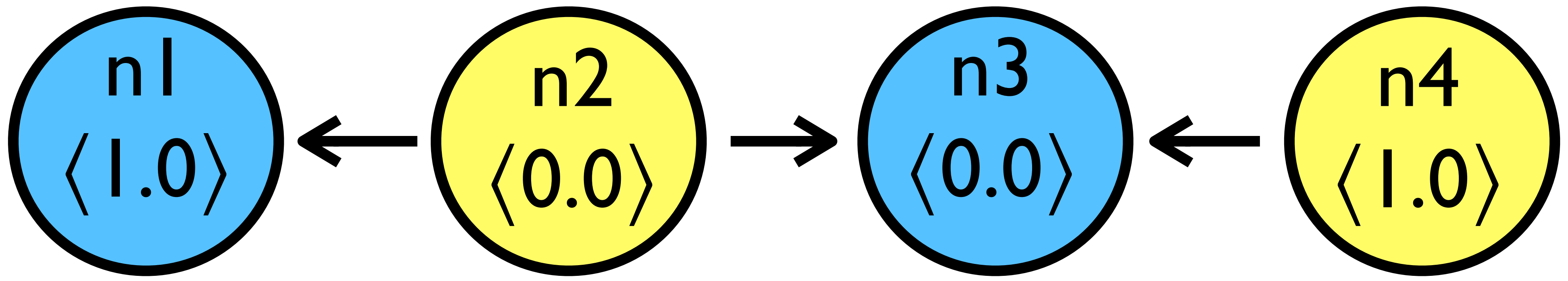
: label 2

Learning Objective

Generate a GDL program that includes the target node n1 and precisely describes the nodes belong to the label 1 ()

Top-down Learning Algorithm

Training graph :



Target node = n1

: label 1

: label 2

(1) Starts from the most general program

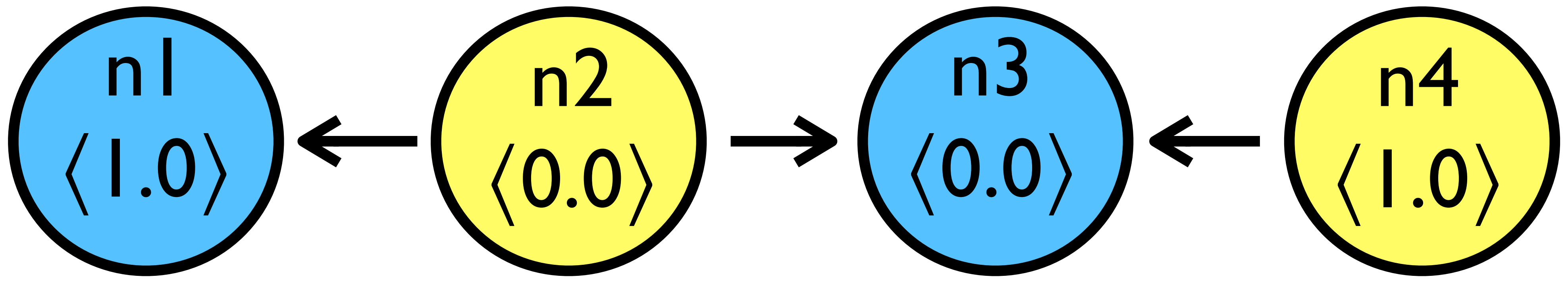
$\langle[-\infty, \infty]\rangle$

Score : 0.4

$$\frac{|\{\mathbf{n1}, n3\}|}{|\{\mathbf{n1}, n2, n3, n4\}| + 1}$$

Top-down Learning Algorithm

Training graph :



Target node = n1

: label 1

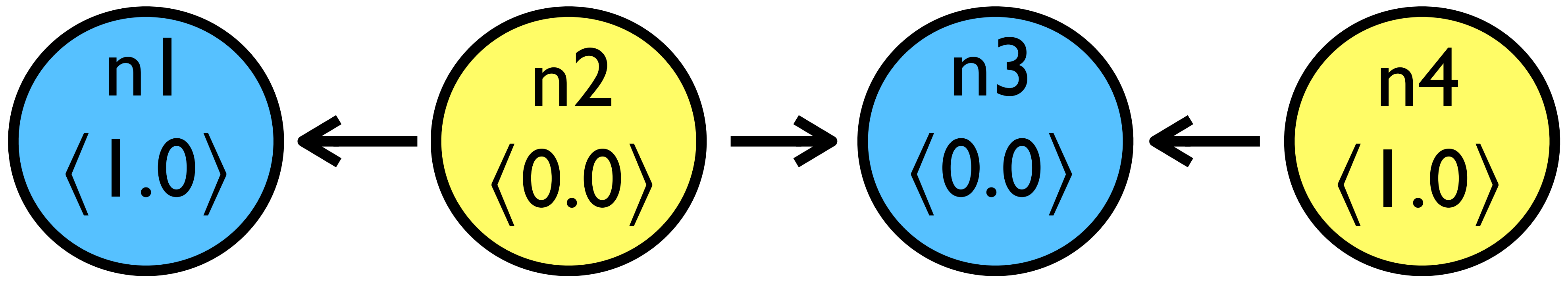
: label 2

(1) Starts from the most general program $\langle[-\infty,\infty]\rangle$ Score : 0.4

(2) Enumerate possible specified programs and choose a better scored one.

Top-down Learning Algorithm

Training graph :



Target node = n1

: label 1

: label 2

(1) Starts from the most general program $\langle[-\infty, \infty]\rangle$ Score : 0.4

(2) Enumerate possible specified programs and choose a better scored one.

$\langle[-\infty, \infty]\rangle \rightarrow \langle[-\infty, \infty]\rangle$ Score : 0.66

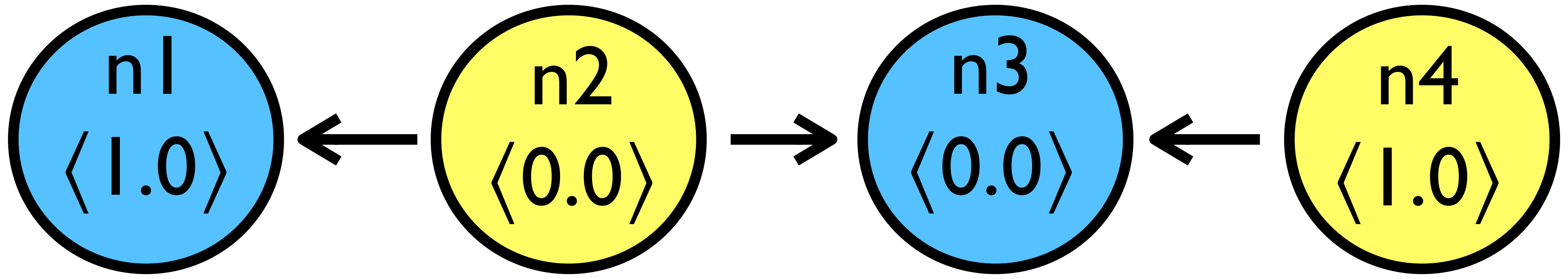
(3) Repeat (2) until no better program is enumerated

(4) Return the current program

$(\langle[-\infty, \infty]\rangle \rightarrow \langle[-\infty, \infty]\rangle$, label 1, 0.66)

Bottom-up Learning Algorithm

Training graph :

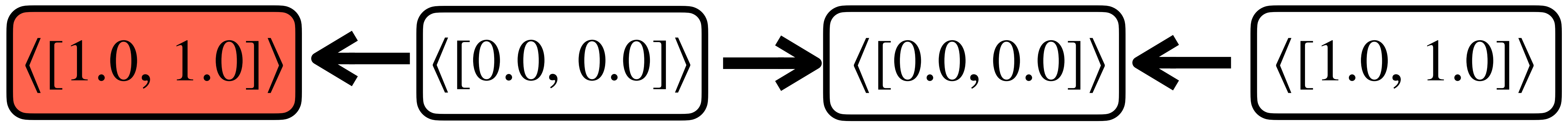


Target node = n1

: label 1

: label 2

(1) Starts from the most specific program



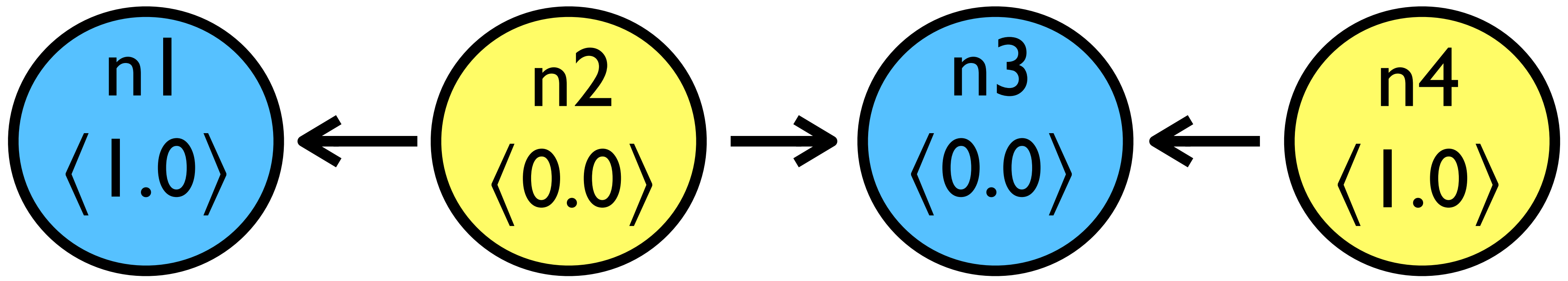
Score : 0.5

$$\frac{|\{\mathbf{n1}\}|}{|\{\mathbf{n1}\}| + 1}$$

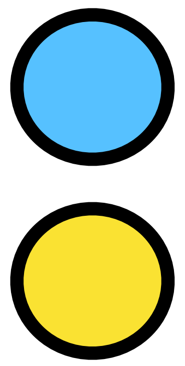
```
node v1 ⟨[1.0,1.0]⟩
node v2 ⟨[0.0,0.0]⟩
node v3 ⟨[0.0,0.0]⟩
node v4 ⟨[0.0,0.0]⟩
edge (v2, v1)
edge (v2, v3)
edge (v4, v3)
target node v1
```

Bottom-up Learning Algorithm

Training graph :



Target node = n1



(1) Starts from the most specific program

⟨[1.0, 1.0]⟩ ← ⟨[0.0, 0.0]⟩ → ⟨[0.0, 0.0]⟩ ← ⟨[1.0, 1.0]⟩

Score : 0.5

$$\frac{|\{n1\}|}{|\{n1\}| + 1}$$

(2) Enumerate possible generalized programs and choose an equal or better scored one

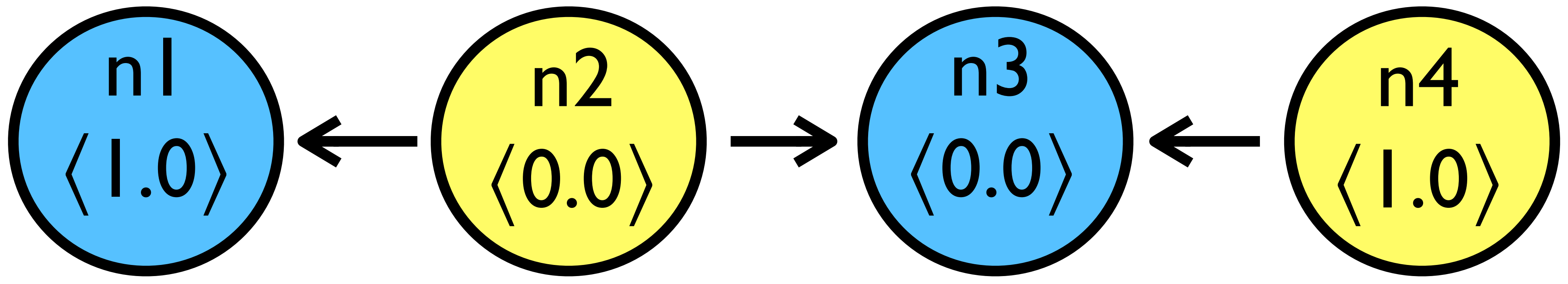
⟨[1.0, 1.0]⟩ ← ⟨[0.0, 0.0]⟩ → ⟨[0.0, 0.0]⟩

Score : 0.5

$$\frac{|\{n1\}|}{|\{n1\}| + 1}$$

Bottom-up Learning Algorithm

Training graph :



Target node = n1

: label 1

: label 2

(1) Starts from the most specific program

$\boxed{\langle[1.0, 1.0]\rangle} \leftarrow \boxed{\langle[0.0, 0.0]\rangle} \rightarrow \boxed{\langle[0.0, 0.0]\rangle} \leftarrow \boxed{\langle[1.0, 1.0]\rangle}$ Score : 0.5

(2) Enumerate possible generalized programs and choose an equal or better scored one

$\boxed{\langle[1.0, 1.0]\rangle} \leftarrow \boxed{\langle[0.0, 0.0]\rangle} \rightarrow \boxed{\langle[0.0, 0.0]\rangle}$ Score : 0.5

(3) Repeat (2) until all the enumerated programs have lower score

(4) Return the current program ($\boxed{\langle[-\infty, \infty]\rangle} \leftarrow \boxed{\langle[-\infty, \infty]\rangle}$, label 1, 0.66)

Target node = n1
n1
⟨1.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 1, 0.66)
Target node = n2
n2
⟨0.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 2, 0.66)
Target node = n3
n3
⟨0.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 1, 0.66)
Target node = n4
n4
⟨1.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 2, 0.66)
Training graph
{(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 1, 0.66),
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 2, 0.66)}
Learned model

Target node = n1
n1
⟨1.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 1, 0.66)
Target node = n2
n2
⟨0.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 2, 0.66)
Target node = n3
n3
⟨0.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 1, 0.66)
Target node = n4
n4
⟨1.0⟩
cpu
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 2, 0.66)
Training graph
{(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 1, 0.66),
(⟨[−∞, ∞]⟩ ← ⟨[−∞, ∞]⟩, 2, 0.66),
(⟨[−∞, ∞]⟩, 1, 0.0) }
Learned model
Default triple

Accuracy Comparison

	GCN	GAT	ChebyNet	JKNet	GraphSage	GIN	DGCN	PL4XGL
MUTAG	80.0±0.0	89.0±2.2	86.0±4.1	68.0±7.5	78.0±4.4	91.0±5.4	N/A	**100.0±0.0**
BBBP	83.6±1.4	82.3±1.6	84.6±1.0	85.6±1.9	86.6±0.9	86.2±1.4	N/A	**86.8±0.0**
BACE	78.4±2.8	52.4±3.3	78.9±1.4	79.9±1.9	79.8±0.8	**80.9±0.4**	N/A	**80.9±0.0**
HIV	96.4±0.0	96.4±0.0	96.8±0.2	96.8±0.1	**96.9±0.2**	96.8±0.1	N/A	N/A
BA-Shapes	95.1±0.6	76.8±2.3	**97.1±0.0**	94.3±0.0	**97.1±0.0**	92.0±1.1	95.1±0.7	95.7±0.0
Tree-Cycles	97.7±0.0	90.9±0.0	**100.0±0.0**	98.9±0.0	**100.0±0.0**	93.2±0.0	99.2±0.5	**100.0±0.0**
Wisconsin	64.0±0.0	49.6±3.1	86.4±3.9	64.8±1.5	92.8±2.9	56.0±0.0	**96.0±0.0**	88.0±0.0
Texas	67.7±5.3	50.0±0.0	87.7±2.1	68.8±4.3	86.6±2.6	50.0±0.0	**86.6±2.6**	83.3±0.0
Cornell	58.9±2.6	61.1±0.0	81.0±6.5	61.1±0.0	87.7±2.1	61.1±0.0	86.6±2.6	**88.8±0.0**
Cora	85.6±0.3	86.4±1.8	86.5±5.2	84.9±3.5	86.3±3.2	**86.7±0.0**	83.2±5.9	80.0± 0.0
Citeseer	75.2±0.0	74.3±0.7	**79.1±0.9**	73.7±4.2	75.9±2.3	75.2±0.0	71.3±6.0	63.8± 0.0
Pubmed	82.8±1.1	84.7±1.2	**88.7±1.0**	83.2±0.4	88.0±0.4	86.1±0.6	85.1±0.6	81.4±0.0

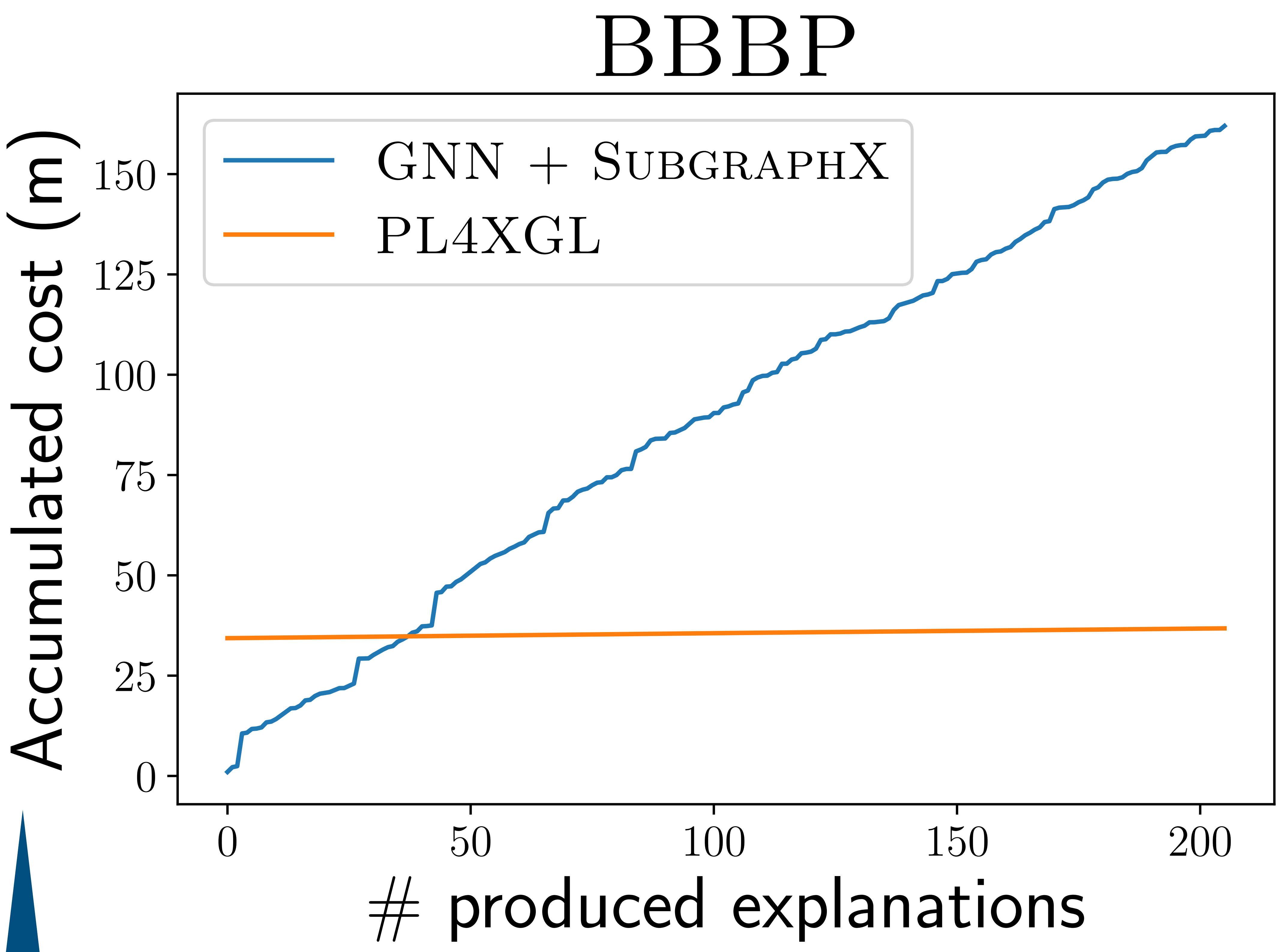
Molecule datasets (graph classification)

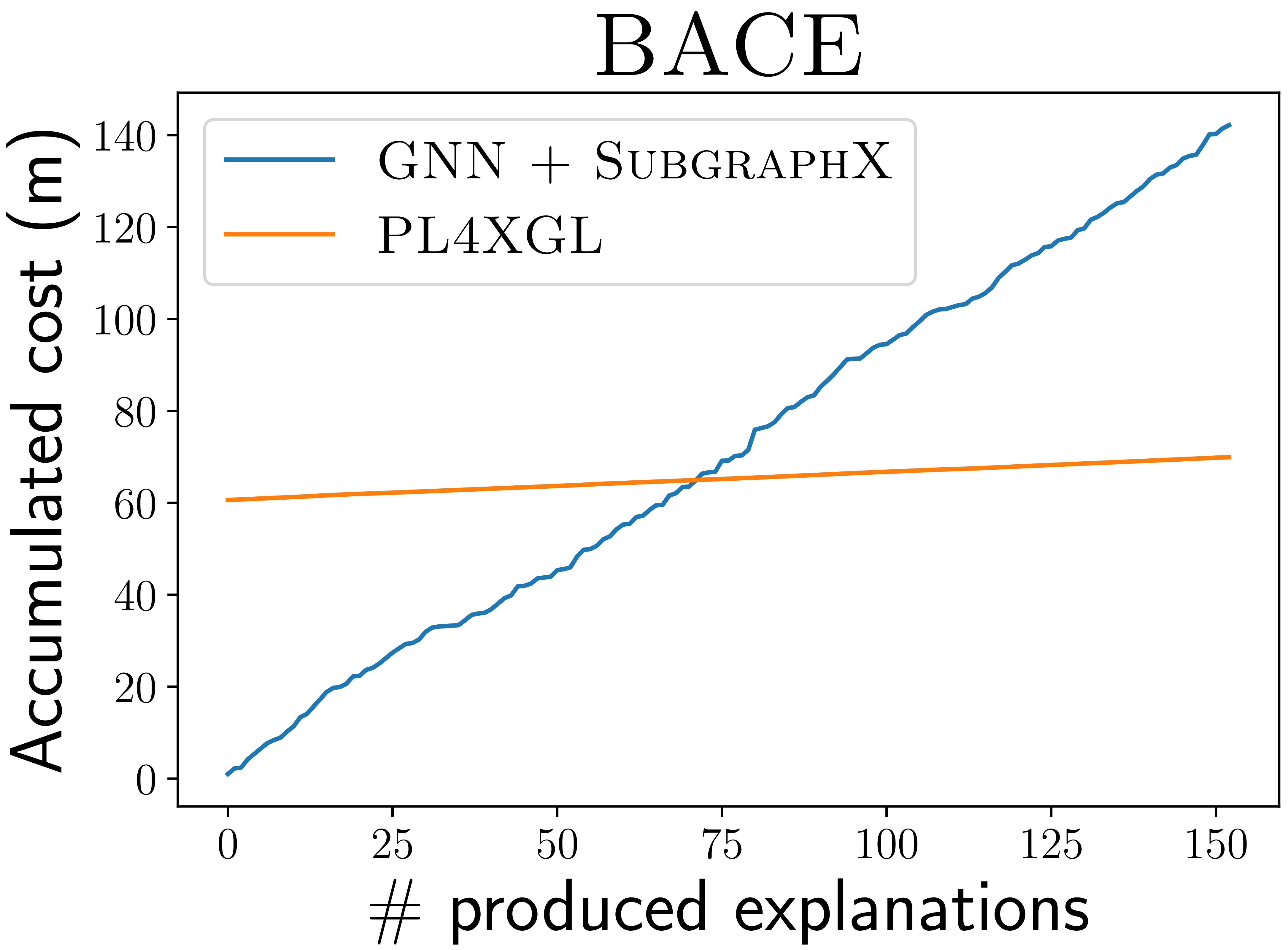
	GCN	GAT	ChebyNet	JKNet	GraphSage	GIN	DGCN	PL4XGL
MUTAG	80.0±0.0	89.0±2.2	86.0±4.1	68.0±7.5	78.0±4.4	91.0±5.4	N/A	**100.0±0.0**
BBBP	83.6±1.4	82.3±1.6	84.6±1.0	85.6±1.9	86.6±0.9	86.2±1.4	N/A	**86.8±0.0**
BACE	78.4±2.8	52.4±3.3	78.9±1.4	79.9±1.9	79.8±0.8	**80.9±0.4**	N/A	**80.9±0.0**
HIV	96.4±0.0	96.4±0.0	96.8±0.2	96.8±0.1	**96.9±0.2**	96.8±0.1	N/A	N/A
BA-Shapes	95.1±0.6	76.8±2.3	**97.1±0.0**	94.3	[illegible]	[illegible]	[illegible]	[illegible]
Tree-Cycles	97.7±0.0	90.9±0.0	**100.0±0.0**	98.9	[illegible]	[illegible]	[illegible]	[illegible]
Wisconsin	64.0±0.0	49.6±3.1	86.4±3.9	64.8	[illegible]	[illegible]	[illegible]	[illegible]
Texas	67.7±5.3	50.0±0.0	87.7±2.1	68.8±4.3	86.6±2.6	50.0±0.0	**86.6±2.6**	83.3±0.0
Cornell	58.9±2.6	61.1±0.0	81.0±6.5	61.1±0.0	87.7±2.1	61.1±0.0	86.6±2.6	**88.8±0.0**
Cora	85.6±0.3	86.4±1.8	86.5±5.2	84.9±3.5	86.3±3.2	**86.7±0.0**	83.2±5.9	80.0± 0.0
Citeseer	75.2±0.0	74.3±0.7	**79.1±0.9**	73.7±4.2	75.9±2.3	75.2±0.0	71.3±6.0	63.8± 0.0
Pubmed	82.8±1.1	84.7±1.2	**88.7±1.0**	83.2±0.4	88.0±0.4	86.1±0.6	85.1±0.6	81.4±0.0

PL4XGL shows the best accuracy

Cost Comparison

- Our approach is fast if explanation cost is included



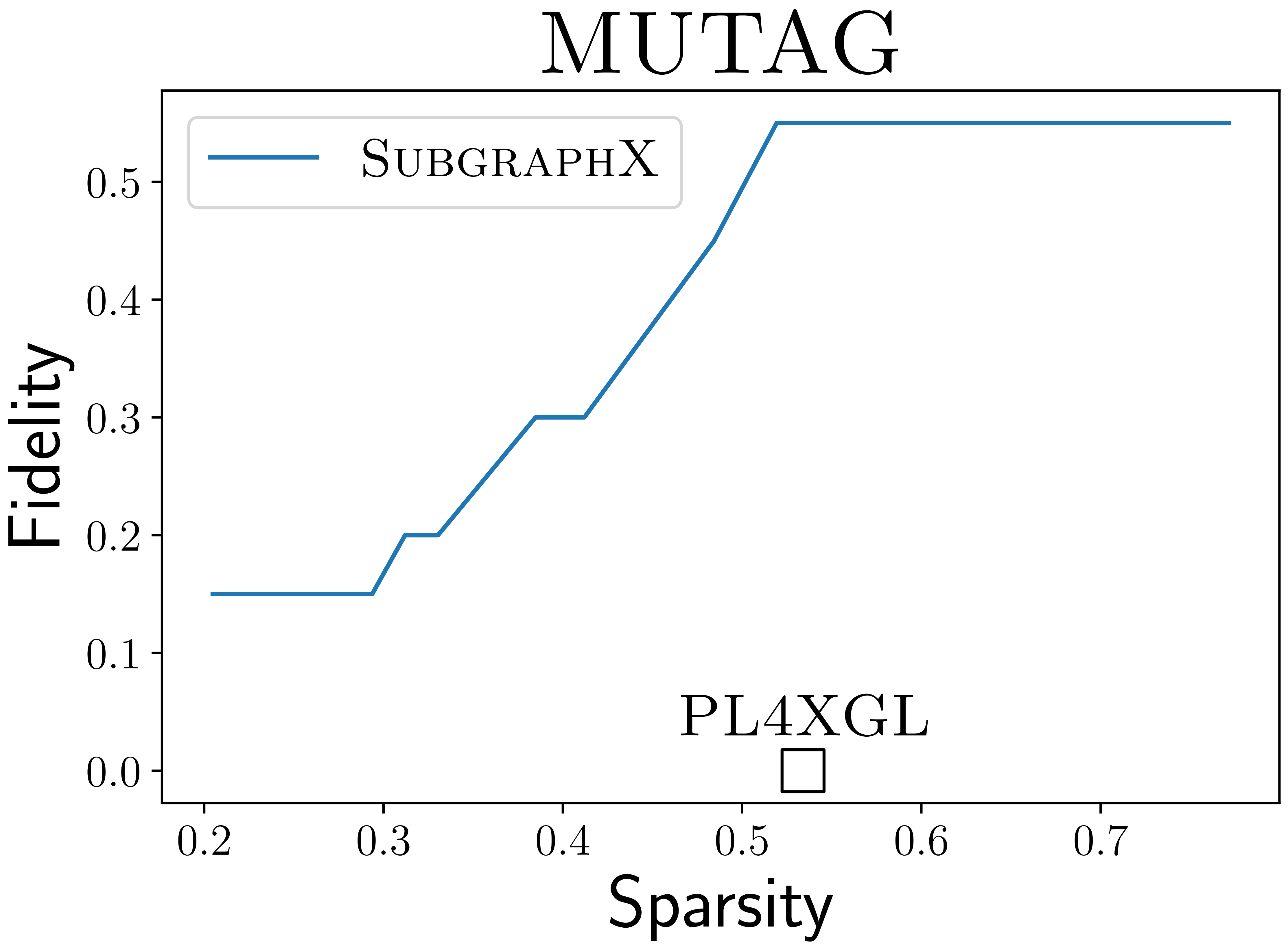


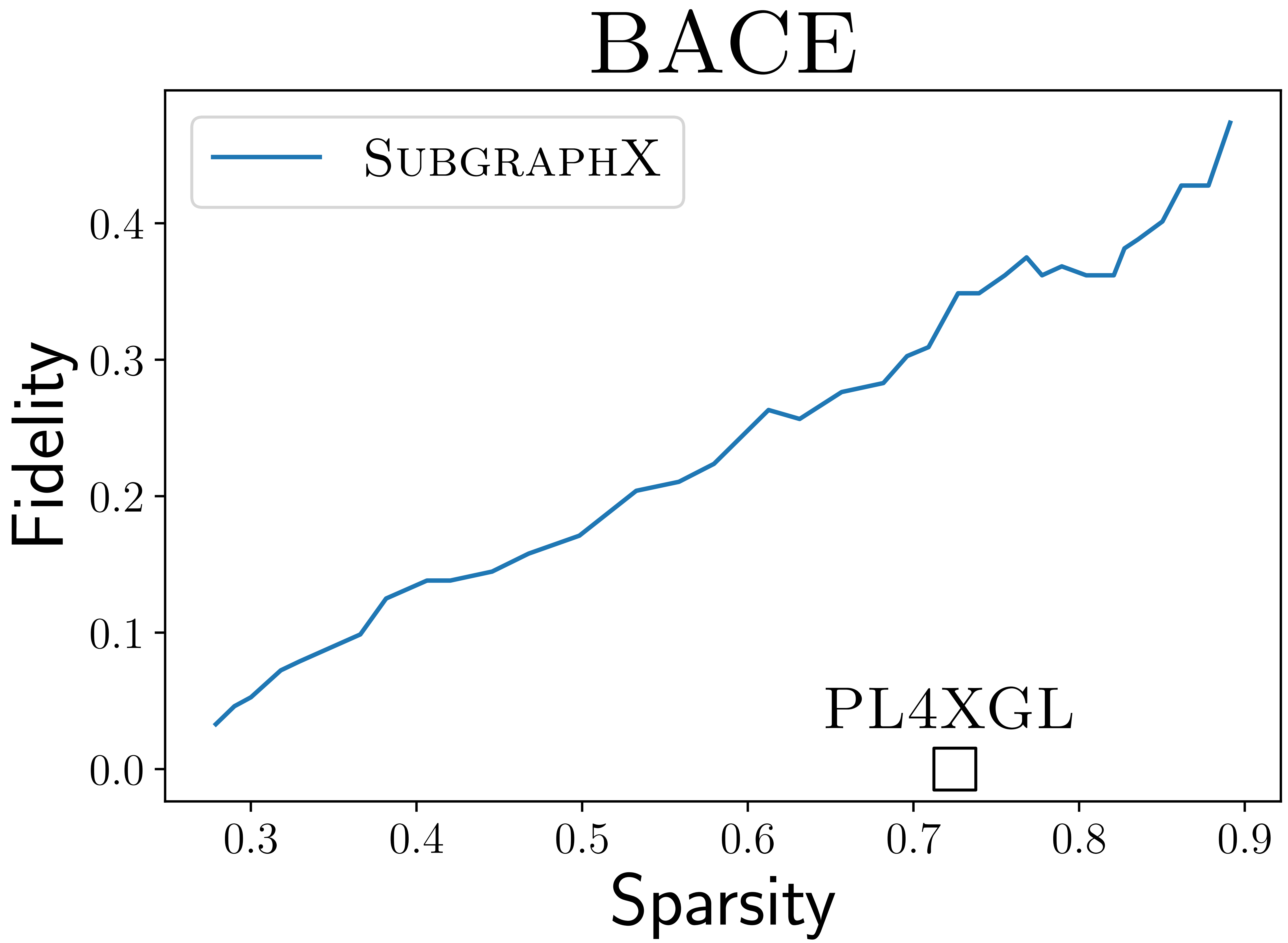
Training + classification + explanation cost

Explainability Comparison

- Our approach provides correct & simple explanations

Explanations are correct





The explanations are simple

GDL Program vs Subgraph

- GDL is a strictly more expressive than subgraph in graph pattern description

$$\langle 1 \rangle \longrightarrow \langle 0 \rangle \quad = \quad \langle [1, 1] \rangle \longrightarrow \langle [0, 0] \rangle$$

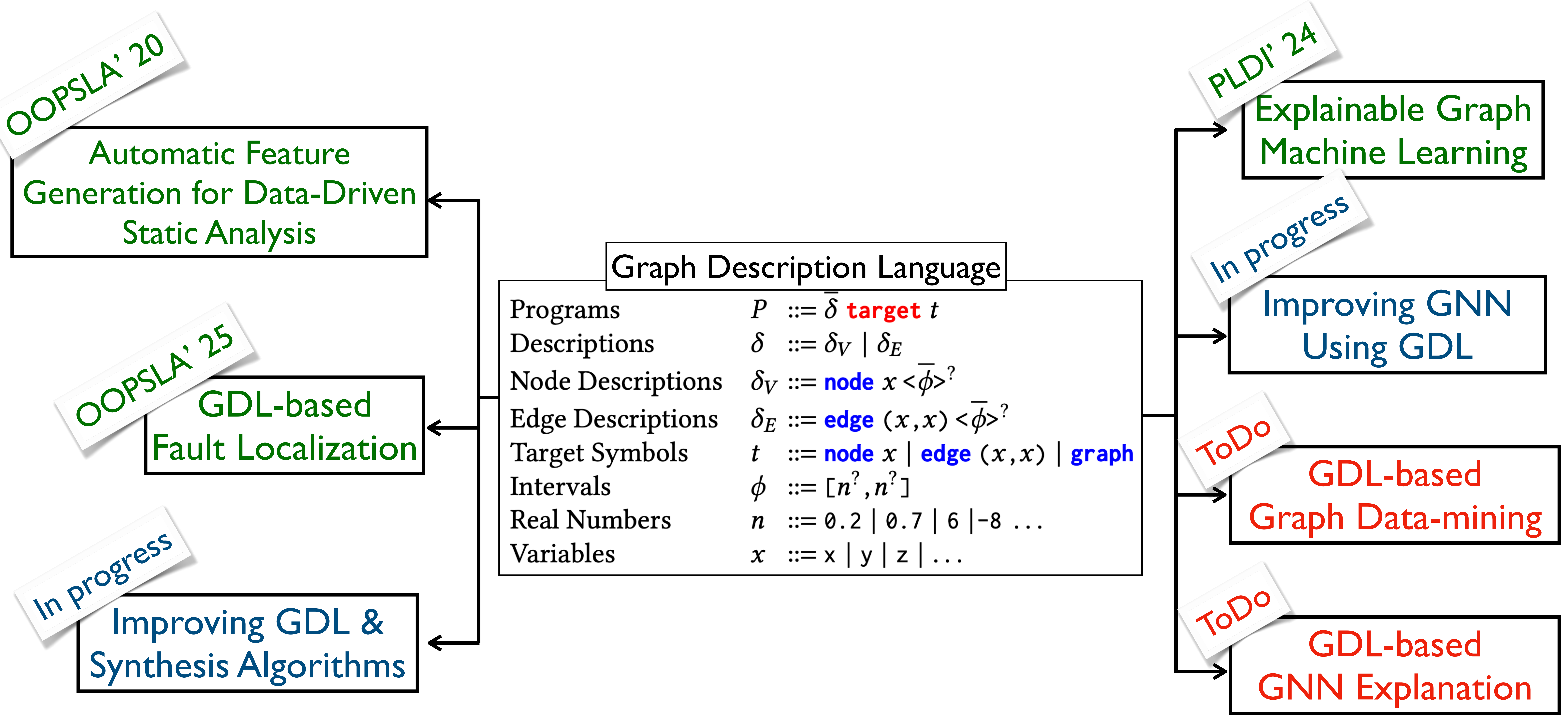
subgraph pattern

GDL program

$$\langle [0, 1] \rangle \longrightarrow \langle [0, 1] \rangle \quad = \quad \left\{ \langle 0.0 \rangle \longrightarrow \langle 0.0 \rangle, \cdots, \langle 1.0 \rangle \longrightarrow \langle 1.0 \rangle \right\}$$

GDL program

subgraph patterns


OOPSLA' 20
Automatic Feature Generation for Data-Driven Static Analysis
OOPSLA' 25
GDL-based Fault Localization
In progress
Improving GDL & Synthesis Algorithms
Graph Description Language
Programs P ::= $\overline{\delta}$ target t
Descriptions δ ::= δ_V | δ_E
Node Descriptions δ_V ::= node x <$\overline{\phi}$>$^?$
Edge Descriptions δ_E ::= edge (x,x) <$\overline{\phi}$>$^?$
Target Symbols t ::= node x | edge (x,x) | graph
Intervals ϕ ::= [$n^?$,$n^?$]
Real Numbers n ::= 0.2 | 0.7 | 6 | -8 ...
Variables x ::= x | y | z | ...
PLDI' 24
Explainable Graph Machine Learning
In progress
Improving GNN Using GDL
ToDo
GDL-based Graph Data-mining
ToDo
GDL-based GNN Explanation