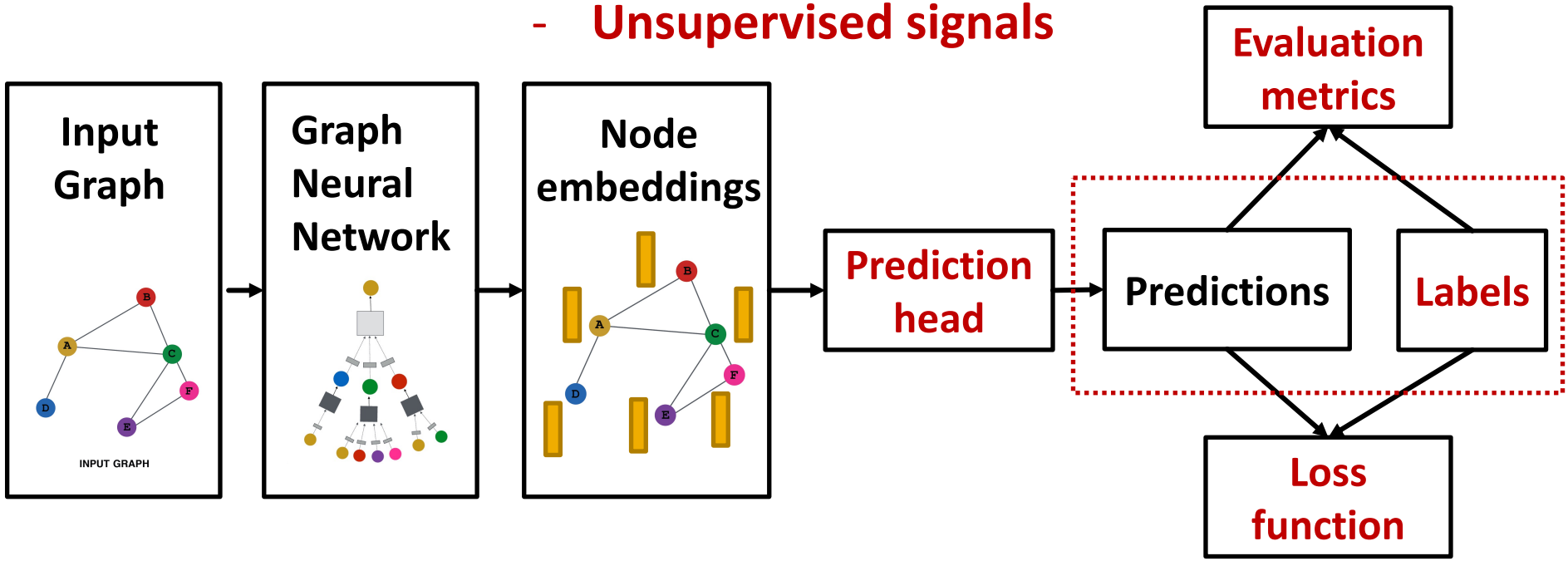
Training Graph Neural Networks

GNN Training Pipeline (2)

(2) Where does ground-truth come from?

- **Supervised labels**
- **Unsupervised signals**



Supervised vs Unsupervised

- **Supervised learning on graphs**
  - **Labels come from external sources**
    - E.g., predict drug likeness of a molecular graph
- **Unsupervised learning on graphs**
  - **Signals come from graphs themselves**
    - E.g., link prediction: predict if two nodes are connected
- **Sometimes the differences are blurry**
  - We still have "supervision" in unsupervised learning
    - E.g., train a GNN to predict node clustering coefficient
  - An alternative name for "**unsupervised**" is "**self-supervised**"

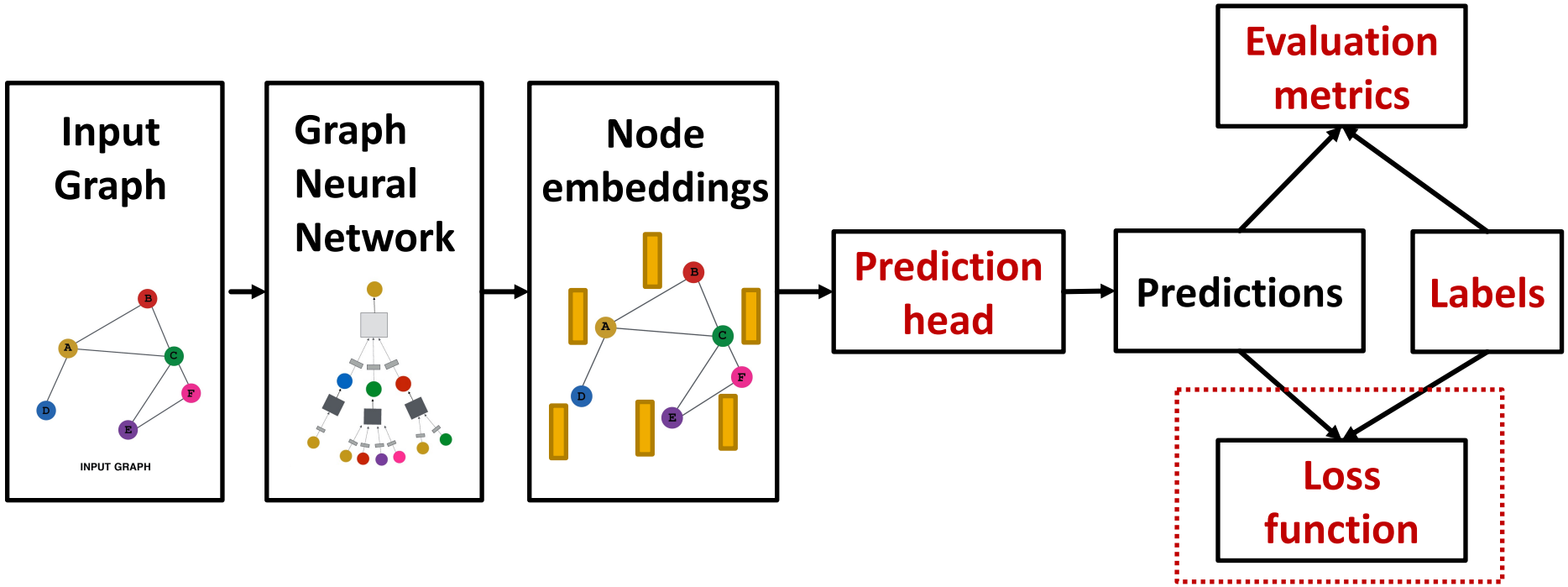
Supervised Labels on Graphs

- **Supervised labels come from the specific use cases**. For example:
  - **Node labels y_v:** in a citation network, which subject area does a node belong to
  - **Edge labels y_{uv}:** in a transaction network, whether an edge is fraudulent
  - **Graph labels y_G:** among molecular graphs, the drug likeness of graphs
- **Advice:** Reduce your task to node / edge / graph labels, since they are easy to work with
  - **E.g.,** we knew some nodes form a cluster. We can treat the cluster that a node belongs to as a **node label**

Unsupervised Signals on Graphs

- **The problem:** sometimes **we only have a graph, without any external labels**
- **The solution:** "self-supervised learning", we can find supervision signals within the graph.
  - For example, we can **let GNN predict the following:**
  - **Node-level y_v.** Node statistics: such as clustering coefficient, PageRank, …
  - **Edge-level y_{uv}.** Link prediction: hide the edge between two nodes, predict if there should be a link
  - **Graph-level y_G.** Graph statistics: for example, predict if two graphs are isomorphic
  - **These tasks do not require any external labels!**

GNN Training Pipeline (3)



(3) How do we compute the final loss?

- **Classification loss**
- **Regression loss**

Settings for GNN Training

- **The setting:** We have N data points
  - Each data point can be a node/edge/graph
  - **Node-level**: prediction $\hat{\boldsymbol{y}}_v^{(i)}$, label $\boldsymbol{y}_v^{(i)}$
  - **Edge-level**: prediction $\hat{\boldsymbol{y}}_{uv}^{(i)}$, label $\boldsymbol{y}_{uv}^{(i)}$
  - **Graph-level**: prediction $\hat{\boldsymbol{y}}_G^{(i)}$, label $\boldsymbol{y}_G^{(i)}$
  - We will use prediction $\hat{\boldsymbol{y}}^{(i)}$, label $\boldsymbol{y}^{(i)}$ to refer **predictions at all levels**

Classification or Regression

- **Classification**: labels $\boldsymbol{y}^{(i)}$ with discrete value
  - E.g., Node classification: which category does a node belong to
- **Regression**: labels $\boldsymbol{y}^{(i)}$ with continuous value
  - E.g., predict the drug likeness of a molecular graph
- GNNs can be applied to both settings
- **Differences: loss function & evaluation metrics**

Classification Loss

- As discussed in lecture 6, **cross entropy (CE)** is a very common loss function in classification
- K-way prediction for i-th data point:

$$\mathrm{CE}\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = -\sum_{j=1}^{K} \boldsymbol{y}_j^{(i)} \log\left(\hat{\boldsymbol{y}}_j^{(i)}\right)$$

Label: $\boldsymbol{y}^{(i)}$ Prediction: $\hat{\boldsymbol{y}}^{(i)}$

i-th data point: (i) j-th class: j

where:

E.g. | 0 | 0 | 1 | 0 | 0 |

$\boldsymbol{y}^{(i)} \in \mathbb{R}^K$ = one-hot label encoding

$\hat{\boldsymbol{y}}^{(i)} \in \mathbb{R}^K$ = prediction after $\mathrm{Softmax}(\cdot)$

E.g. | 0.1 | 0.3 | 0.4 | 0.1 | 0.1 |

- Total loss over all N training examples

$$\text{Loss} = \sum_{i=1}^{N} \mathrm{CE}\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right)$$

Regression Loss

- For regression tasks we often use **Mean Squared Error (MSE)** a.k.a. **L2 loss**
- K-way regression for data point (i):

$$\text{MSE}\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = \sum_{j=1}^{K} \left(\boldsymbol{y}_j^{(i)} - \hat{\boldsymbol{y}}_j^{(i)}\right)^2$$

i-th data point

j-th target

where:

E.g. | 1.4 | 2.3 | 1.0 | 0.5 | 0.6 |

$\boldsymbol{y}^{(i)} \in \mathbb{R}^k$ = Real valued vector of targets

$\hat{\boldsymbol{y}}^{(i)} \in \mathbb{R}^k$ = Real valued vector of predictions

E.g. | 0.9 | 2.8 | 2.0 | 0.3 | 0.8 |

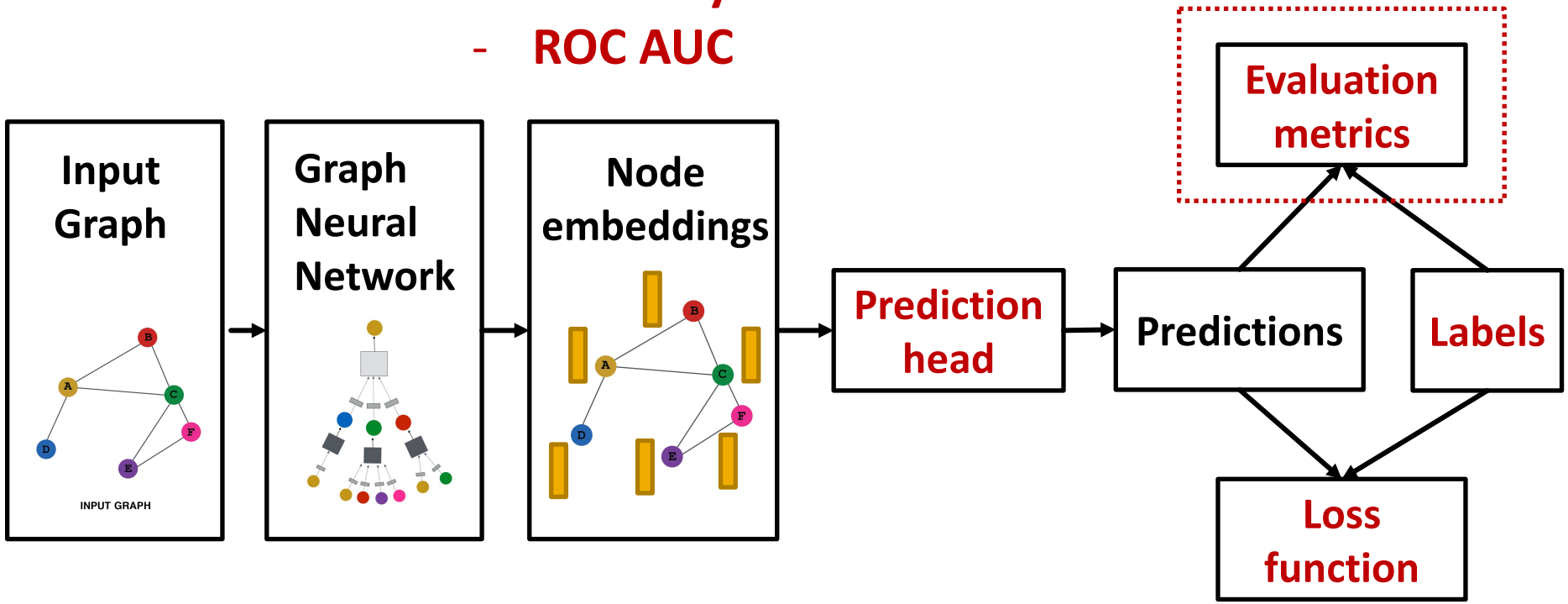
- Total loss over all N training examples

$$\text{Loss} = \sum_{i=1}^{N} \text{MSE}\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right)$$

GNN Training Pipeline (4)

(4) How do we measure the success of a GNN?

- **Accuracy**
- **ROC AUC**



Evaluation Metrics: Regression

- **We use standard evaluation metrics for GNN**
  - (Content below can be found in any ML course)
  - In practice we will use sklearn for implementation
  - Suppose we make predictions for N data points
- **Evaluate regression tasks on graphs:**
  - **Root mean square error (RMSE)**

$$\sqrt{\sum_{i=1}^{N} \frac{(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{y}}^{(i)})^2}{N}}$$

  - **Mean absolute error (MAE)**

$$\frac{\sum_{i=1}^{N} \left| \boldsymbol{y}^{(i)} - \hat{\boldsymbol{y}}^{(i)} \right|}{N}$$

Evaluation Metrics: Classification

- **Evaluate classification tasks on graphs:**
- **(1) Multi-class classification**
  - **We simply report the accuracy**

$$\frac{1\left[\text{argmax}\left(\hat{\boldsymbol{y}}^{(i)}\right) = \boldsymbol{y}^{(i)}\right]}{N}$$

- **(2) Binary classification**
  - Metrics sensitive to classification threshold
    - **Accuracy**
    - **Precision / Recall**
    - If the range of prediction is $[0,1]$, we will use 0.5 as threshold
  - Metric Agnostic to classification threshold
    - **ROC AUC**

Metrics for Binary Classification

- **Accuracy**:

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{|Dataset|}$$

- **Precision (P)**:

$$\frac{TP}{TP + FP}$$

- **Recall (R)**:

$$\frac{TP}{TP + FN}$$

- **F1-Score**:

$$\frac{2P * R}{P + R}$$

Confusion matrix

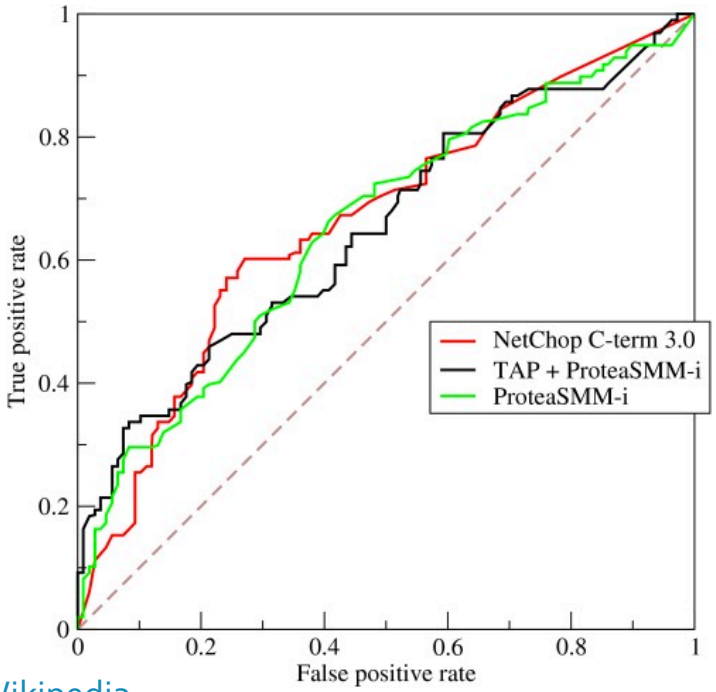
	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Sklearn Classification Report

(4) Evaluation Metrics

- **ROC Curve:** Captures the tradeoff in TPR and FPR **as the classification threshold is varied for a binary classifier.**

TPR



FPR

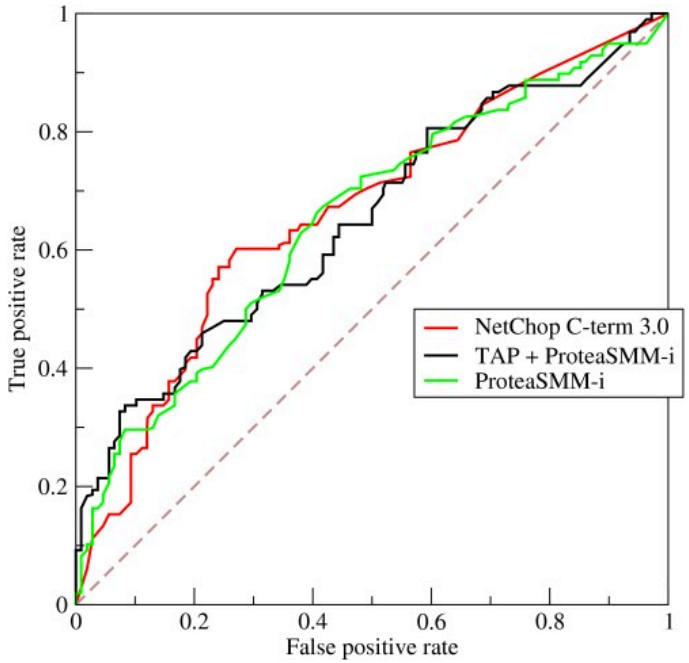
Image Credit: Wikipedia

$$\text{TPR} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Note: the dashed line represents **performance of a random classifier**

(4) Evaluation Metrics



Content Credit: Wikipedia

- **ROC AUC: Area under the ROC Curve.**
- **Intuition:** The probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one