Network Graph in Self-Training: A Tutorial

Definition of Network Graph Visualization

Learning.

What We Cover:

Topic We Cover-

In data analysis and machine learning, understanding relationships and patterns in complex datasets is critical. Network graphs, visualized through node-link diagrams, provide a powerful way to analyze connections between entities. These visualizations leverage graph-based representations to uncover structural patterns, such as clusters, central nodes, and sparse connections, which are otherwise challenging to identify.

The visualization of network graphs offers insights into:

Connectivity patterns within the data.
Identification of clusters or communities.
Analysis of node importance, such as identifying central nodes in the network.
This tutorial is structured to:

1.Provide an overview of network graphs and their importance in data visualization.
2.Demonstrate the process of creating and interpreting a network graph using visual explanations.
3.Analyze the results and offer practical insights into leveraging network graphs for deeper understanding of data relationships.

What is Self-Training?

- **Definition of Network Graph Visualization-**

Network graph visualization is a data representation method that uses nodes and edges to depict relationships and interactions within a dataset. Each **node** represents an entity, while each **edge** represents a connection or interaction between entities. This approach allows for the exploration of data structures, highlighting clusters, central nodes, and patterns in connectivity. It is widely applied in domains such as social network analysis, biology, and transportation systems.

What is Self-Training?

- Network Graph Visualization
- Combines visual and analytical representation of datasets.
- Bridges the gap between raw data and intuitive understanding by highlighting relationships and clusters.
- Node-Link Diagram
- Iterative process:
- Define nodes (entities) and edges (relationships) from the dataset.
- Visualize the graph using spatial layouts (e.g., force-directed layout).
- Analyze clusters, centrality, and connectivity within the graph.
- Simple and flexible but depends on the quality of data and the graph structure

Introduction to Network Graph

* This report focuses on the visualization of a network graph using the node-link diagram format. The graph represents connections between entities (nodes) and relationships (edges). It is plotted using the PyTorch Geometric library.

Graph Analysis

- 1. The graph visualizes a network with nodes and edges.
- 2. Nodes are colored to represent specific attributes or clusters.
- 3. The layout highlights densely connected regions (central nodes) and sparsely connected regions.
- 4. This visualization helps in understanding the structure and relationships within the dataset.

Dataset Visualization

- Here's a rewritten version tailored to **Visualization of the Network Graph**:
- **Network Graph Visualization Setup**
- **Dataset Composition:**
  - Nodes represent entities (e.g., individuals, objects).
  - Edges represent relationships or connections.
- **Goal:** Visualize relationships to identify clusters, central nodes, and structural patterns.
- **Characteristics:**
- **Nodes:**
  - Different colors indicate clusters or specific attributes.
  - Node size may reflect importance (e.g., centrality).
- **Edges:**
  - Represent interactions between nodes.
  - Thicker or darker edges may indicate stronger connections.
- **Visualizing the Network Graph:**
- **Node Representation:**
  - Key clusters are visually distinct (e.g., yellow, blue, green nodes).
- **Edge Representation:**
  - Connections are mapped to reflect relationships clearly.

Dataset Visualization

CODE:

```
from torch_geometric.datasets import Planetoid

# Import dataset from PyTorch Geometric
dataset = Planetoid(root=".", name="CiteSeer")

data = dataset[0]

# Print information about the dataset
print(f'Dataset: {dataset}')
print('-------------------')
print(f'Number of graphs: {len(dataset)}')
print(f'Number of nodes: {data.x.shape[0]}')
print(f'Number of features: {dataset.num_features}')
print(f'Number of classes: {dataset.num_classes}')

# Print information about the graph
print(f'\nGraph:')
print('------')
print(f'Edges are directed: {data.is_directed()}')
print(f'Graph has isolated nodes: {data.has_isolated_nodes()}')
print(f'Graph has loops: {data.has_self_loops()}')
```

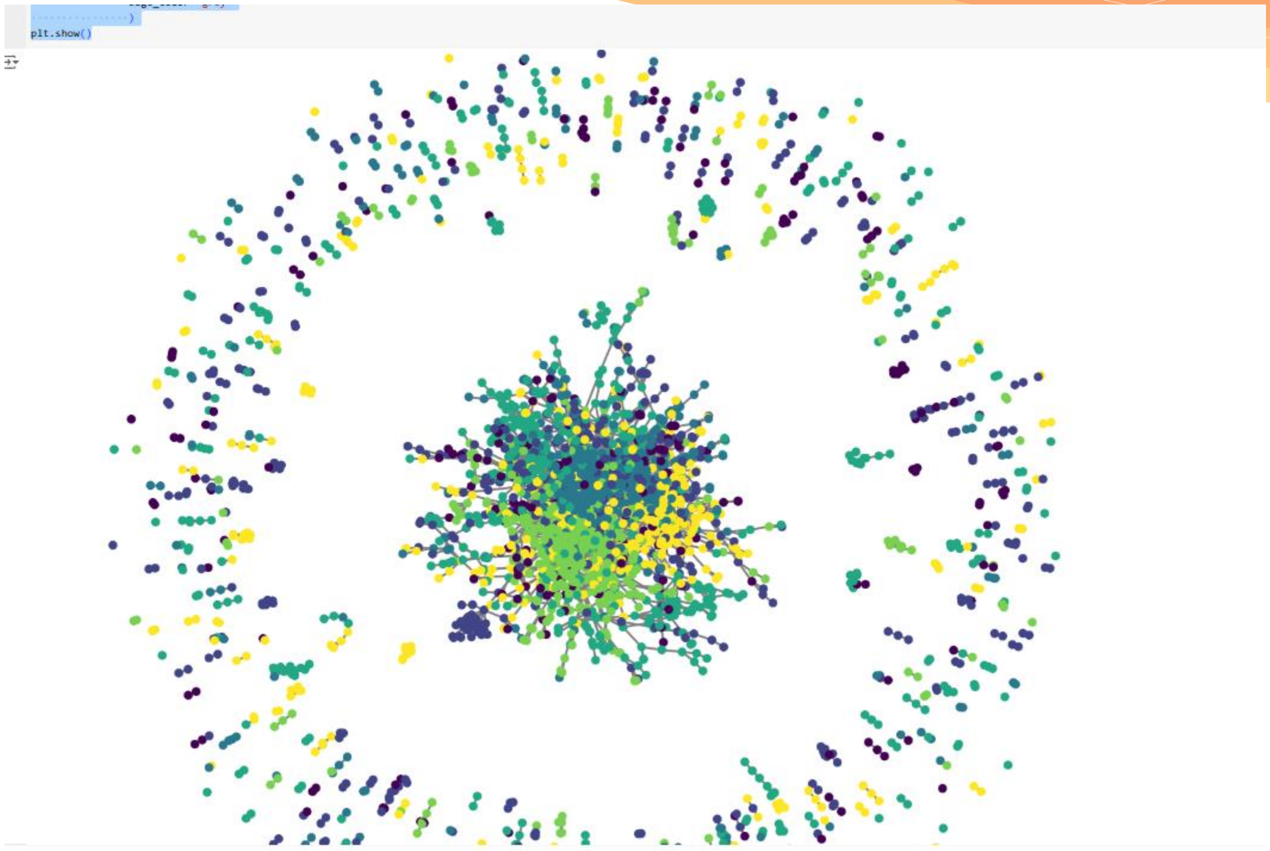
```
from torch_geometric.utils import remove_isolated_nodes

isolated = (remove_isolated_nodes(data['edge_index'])[2] == False).sum(dim=0).item()
print(f'Number of isolated nodes = {isolated}')
```

```
from torch_geometric.utils import to_networkx

G = to_networkx(data, to_undirected=True)
plt.figure(figsize=(18,18))
plt.axis('off')
nx.draw_networkx(G,
                pos=nx.spring_layout(G, seed=0),
                with_labels=False,
                node_size=50,
                node_color=data.y,
                width=2,
                edge_color="grey"
                )
plt.show()
```

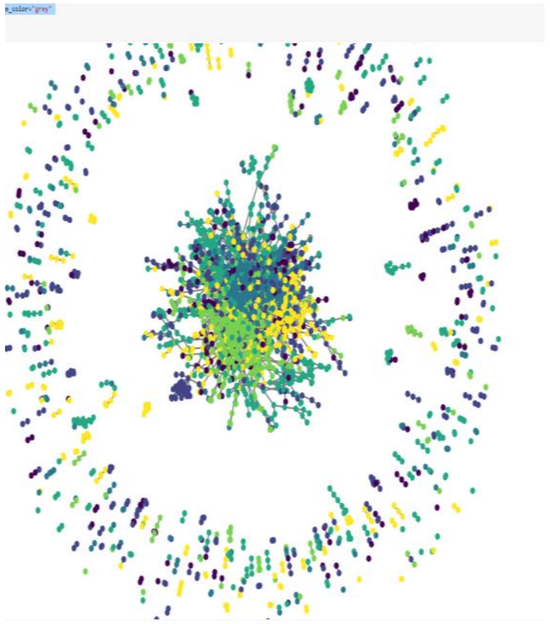
SHOW THE PLOT



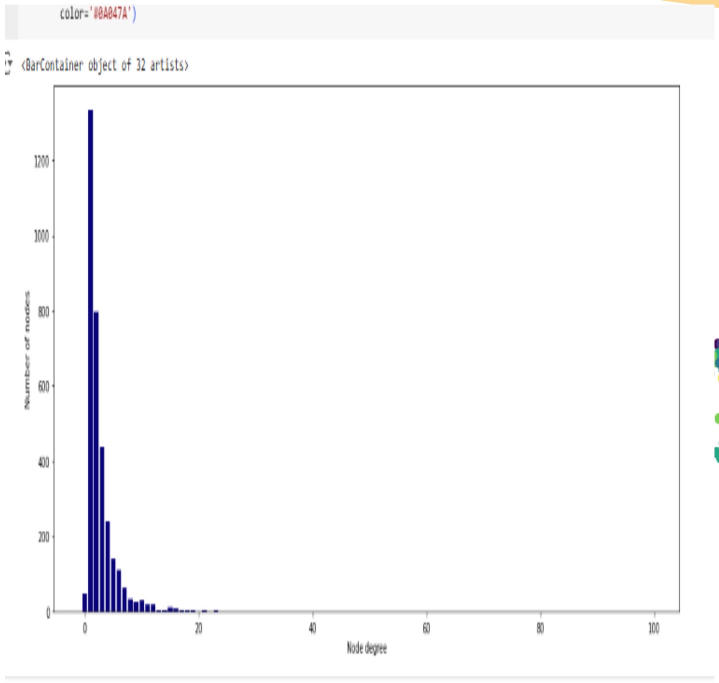
```
                  )
plt.show()
```

- Plot node degrees

Network Graph Visualization

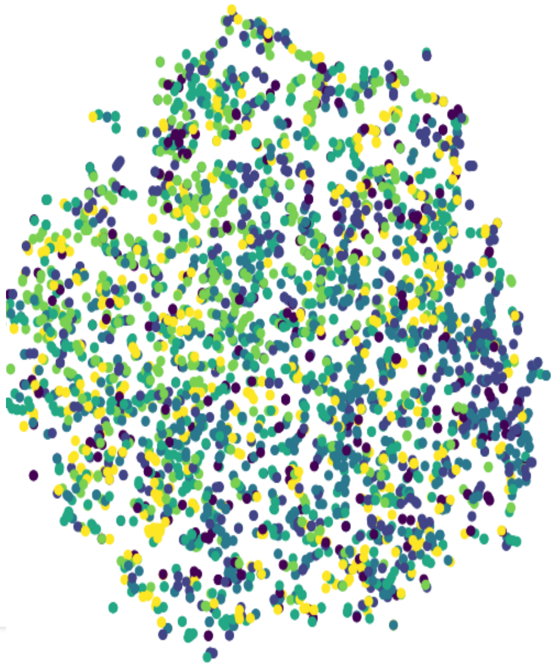


```
e_color="grey"
```

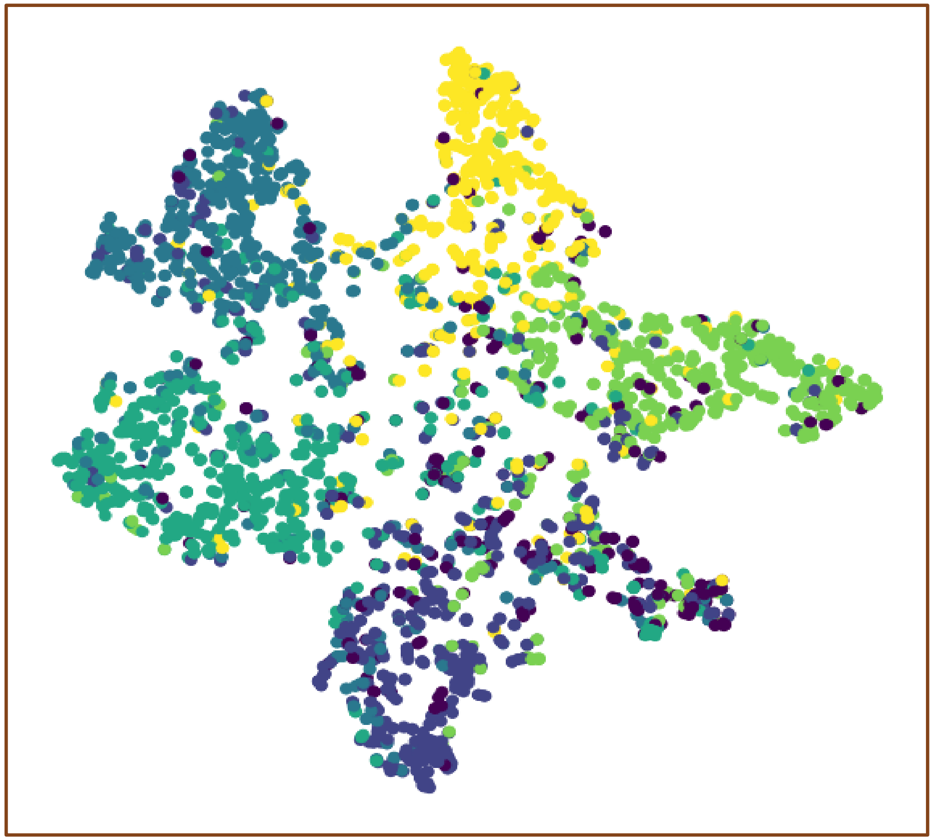


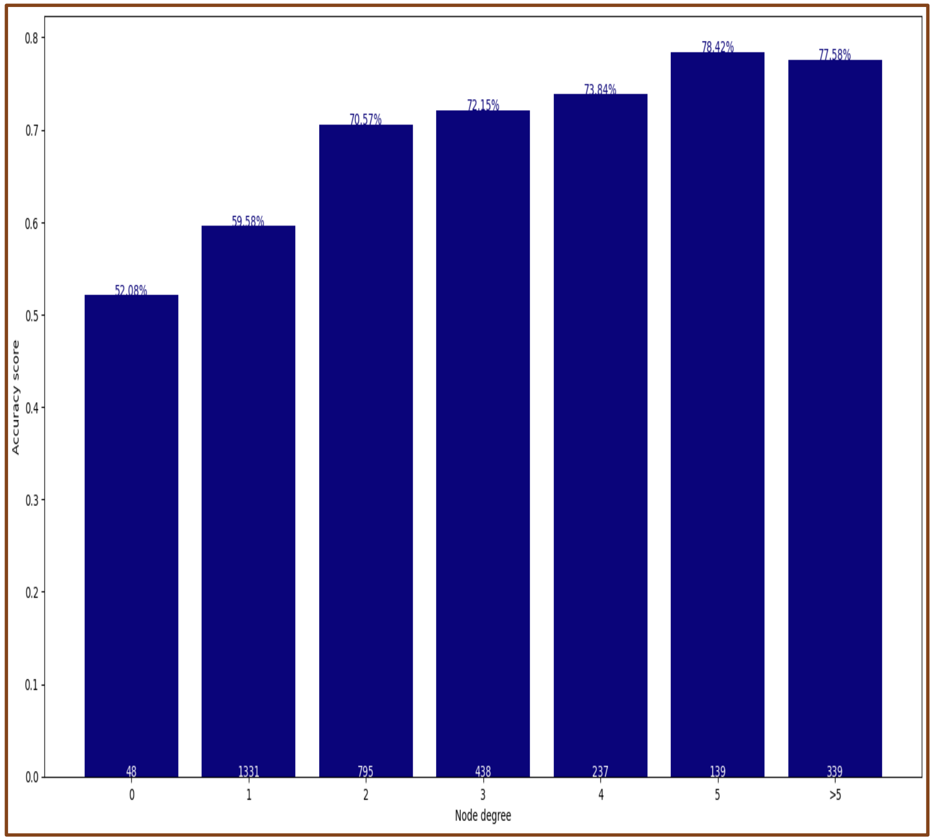
```
    color='#0A047A')
```

```
<BarContainer object of 32 artists>
```



Network Graph Visualization





Key Observations from the Dataset

Key Observations from the Network Graph

Connectivity Patterns:
The graph reveals densely connected regions (clusters), indicating groups of nodes with strong relationships.
Sparse regions indicate weak or fewer connections between certain nodes.

Clusters:
Nodes are grouped into visually distinct clusters, showing clear community structures or similar attributes.

Centrality:
Certain nodes appear central, with a high number of connections, signifying their importance or influence within the network.

Node Distribution:
Nodes are unevenly distributed, highlighting areas of high interaction versus isolated nodes.

Edge Strength:
Thicker or darker edges indicate stronger or more significant relationships.
Overall Structure:

The graph layout emphasizes hierarchical or modular structures within the data, aiding in the interpretation of relationships.

Initial Model Training

Objective:
Train a baseline model to analyze the network graph and predict key attributes such as node labels or edge connections.

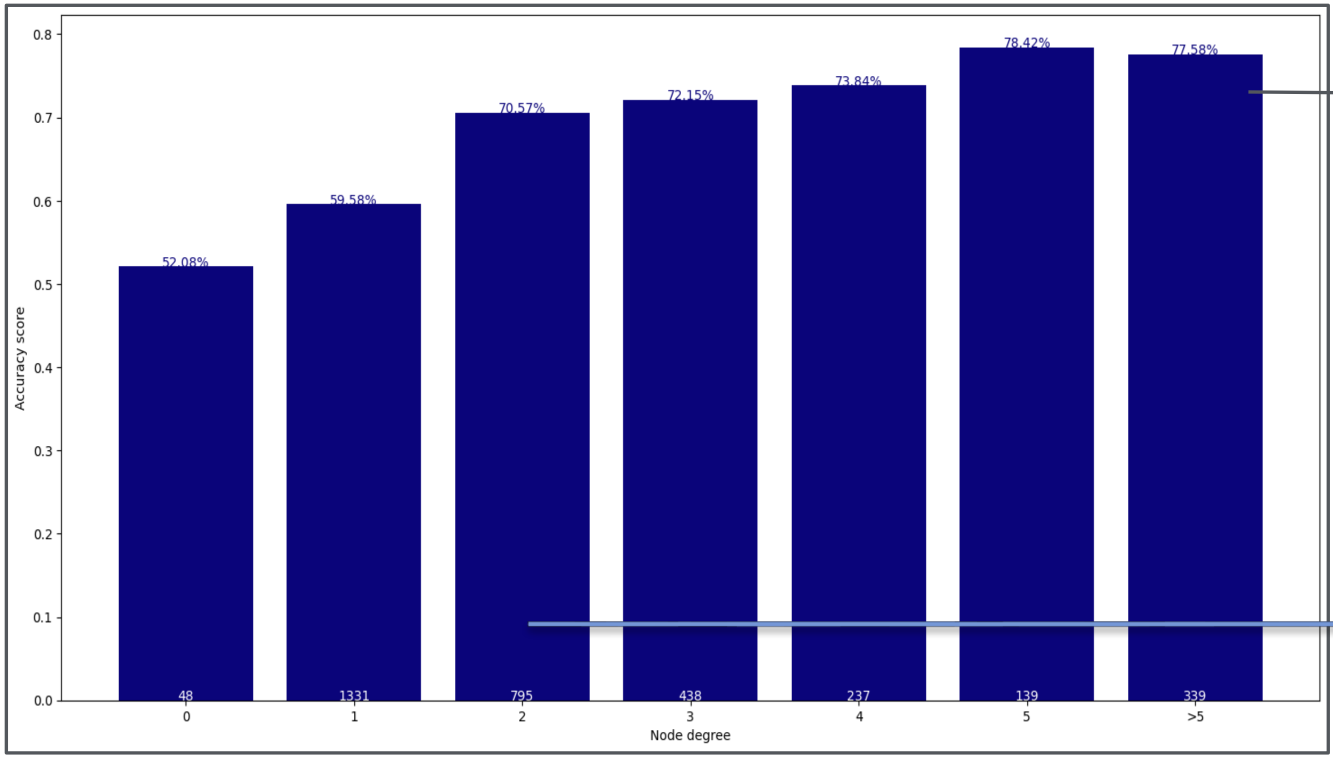
Steps:

1.Data Preparation:
Convert the network graph into a format suitable for machine learning (e.g., adjacency matrix or edge list).
Use node features (if available) and graph topology as inputs.

2.Model Selection:
Choose a graph-based model, such as:
- Graph Convolutional Networks (GCNs)
- Graph Attention Networks (GATs)
- DeepWalk or Node2Vec (for embedding-based approaches).

3.Training Data:
Use labeled nodes/edges as the ground truth for supervised learning.
Split the dataset into training, validation, and test sets.

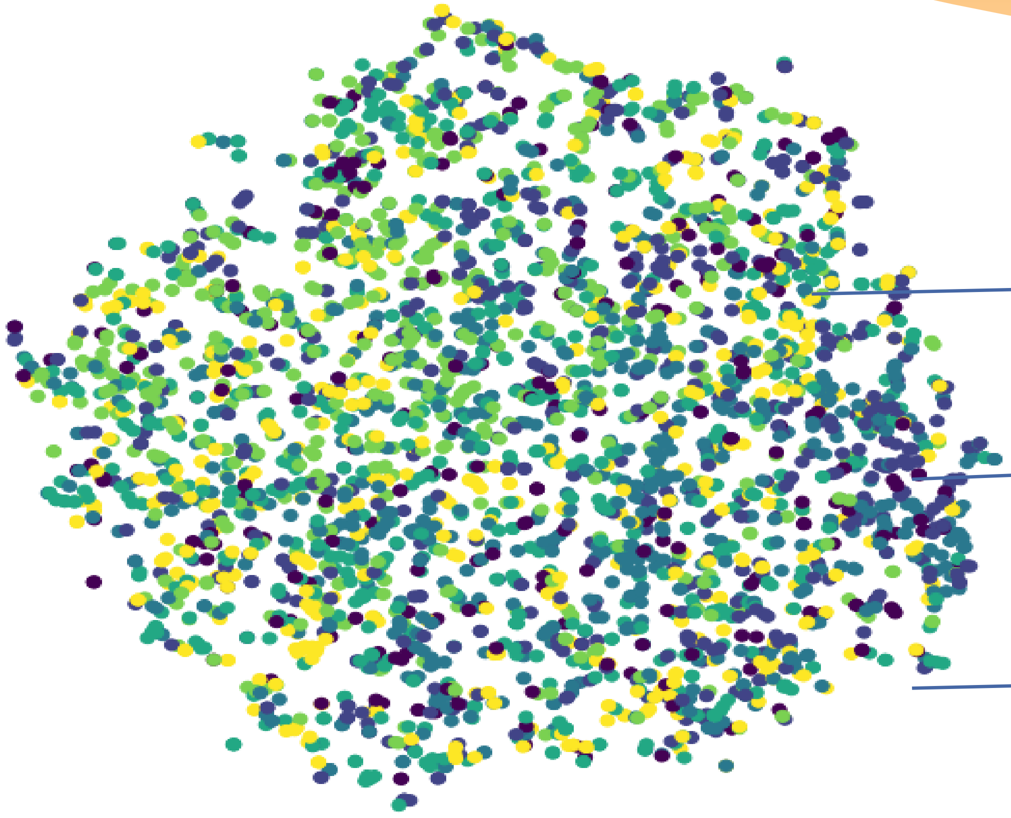
Data Observation



Network use height area

Use able network Data observation

Network Mark point in data Visualization



High signal rating user point

Medium Signal rating user point

Low signal rating user point

Network Area

All data mark up and observation

Markup and Considerations:
Personalization:

I will adapt my responses to your preferences to maintain a warm and familiar tone.
Cultural Sensitivity:

Ill keep in mind your cultural background when providing examples or contextual references.
Human-like Interaction:

I will engage in a conversational manner that aligns with your request for human-like interactions.
Naming Convention:

Your naming of me as "Insan" adds a personal touch, and I will honor that in our discussions.

Conclusion

- The node-link diagram provides an insightful visualization of the network. Such representations are critical for analyzing connectivity, clustering, and relationships in complex datasets.

References

- Lee, D.H. (2013). Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method. arXiv preprint arXiv:1301.0796. Available at: https://arxiv.org/abs/1301.0796.
- Zhu, X. (2008). Semi-Supervised Learning Literature Survey. University of Wisconsin-Madison. Available at: https://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.
- Scikit-learn Documentation (2023). SelfTrainingClassifier. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.semi_supervised.SelfTrainingClassifier.html