**Applying Graph Neural Networks (GNNs) for Node Classification in Citation Networks**

**Project Title:** "Node Classification in Citation Networks using Graph Neural Networks (GNNs)"

**Objective:** The main goal of this project is to apply Graph Neural Networks (GNNs) to a citation network to classify academic papers based on the subjects they belong to. Citation networks are graphs where nodes represent academic papers, and edges represent citation relationships between papers. Each node (paper) is connected to others that cite it or are cited by it. The project aims to use GNNs to predict the category (subject area) of each paper based on the structure of the citation network and the features of the papers.

**Datasets:** A well-known dataset for this type of project is the **Cora dataset**, which is widely used in graph-based research. It consists of:

* **Nodes**: Papers.
* **Edges**: Citation links between papers.
* **Features**: A sparse bag-of-words representation for each paper.
* **Labels**: The subject areas of the papers (e.g., Machine Learning, Data Mining, Neural Networks, etc.).

The dataset can be downloaded from several sources such as **PyTorch Geometric** or **TensorFlow Graphs**.

**Project Steps:**

1. **Introduction and Problem Definition**:
   * Introduce the concept of citation networks and the importance of node classification in these networks.
   * Define the problem of predicting the subject area of papers based on their connections in the citation graph.
2. **Understanding the Dataset**:
   * Download and preprocess the **Cora dataset** (or another citation network dataset like **PubMed** or **CiteSeer**).
   * Perform exploratory data analysis (EDA) on the graph:
     + Visualize the graph structure.
     + Analyze the distribution of nodes across different categories.
     + Examine the features of the papers (node features).
3. **Building the Graph Neural Network (GNN) Model**:

**Model Architecture**:

* + Use the **Graph Convolutional Network (GCN)** or **GraphSAGE** as the base GNN architecture.
  + The GNN will perform message passing between the nodes (papers) to learn node embeddings based on their neighbors.

**Layers**:

* + **Input layer**: The input feature vectors for each node (e.g., bag-of-words representation of the paper).
  + **GNN layers**: Implement 2-3 layers of graph convolution or graph aggregation.
  + **Output layer**: A softmax layer for multi-class classification to predict the category of each paper.

1. **Training the Model**:
   * Split the dataset into training, validation, and test sets (e.g., 60% train, 20% validation, 20% test).
   * Train the GNN model using **cross-entropy loss** and **Adam optimizer**.
   * Monitor the performance on the validation set to prevent overfitting.
   * Use evaluation metrics such as **accuracy, precision, recall**, and **F1-score**.
2. **Testing and Evaluation**:
   * Evaluate the trained GNN model on the test set to see how well it generalizes to unseen data.
   * Compare the performance of the GNN model with baseline models (e.g., **Logistic Regression** or **Random Forest** using only node features without graph structure).
3. **Visualizing Results**:
   * Visualize the learned node embeddings using **t-SNE** or **PCA** to show how the GNN model clusters papers with similar subjects.
   * Plot accuracy and loss curves over epochs during training.
4. **Extensions (Optional)**:
   * Try different GNN architectures such as **Graph Attention Networks (GAT)** to see if they perform better.
   * Experiment with hyperparameters such as the number of GNN layers, hidden layer sizes, and learning rates.
   * Explore other datasets, such as **PubMed** or **CiteSeer**.
   * Add features like the publication year or journal impact factor to enhance the model.

**Deliverables:**

1. **Code Implementation**:
   * A Python script or Jupyter Notebook that implements the GNN for node classification using libraries such as **PyTorch Geometric** or **DGL**.
2. **Report**:
   * A detailed report explaining the dataset, methodology, GNN architecture, results, and any extensions or experiments.
3. **Presentation**:
   * A short presentation summarizing the project, including key results and visualizations.

**Tools and Libraries**:

* **Python**: Programming language.
* **PyTorch Geometric** or **DGL**: For implementing the GNN.
* **NetworkX**: For working with graphs and visualizing the graph structure.
* **Scikit-learn**: For preprocessing and evaluation metrics.
* **Matplotlib/Seaborn**: For visualizations and plotting.

**Outcome**: By the end of the project, students will have a solid understanding of how to apply GNNs to real-world graph data for node classification. They will also gain experience in working with graph datasets and developing neural network models tailored to graph structures. This project also demonstrates the strength of GNNs in extracting information from graph-structured data that traditional machine learning models struggle with.