**GCN Homework Assignment**

**Introduction**

This assignment focuses on implementing **Graph Convolutional Networks (GCNs)** for **graph classification** using the **ENZYMES dataset** from PyTorch Geometric. You will explore graph structures, preprocess the data, implement GCN models (both manually and using PyG), and evaluate their performance.

**Dataset: ENZYMES**

The **ENZYMES dataset** consists of **600 graphs**, where each graph represents a **protein**. Nodes correspond to **amino acids**, and edges represent **spatial proximity** between residues. The task is to classify each graph into one of **six enzyme classes**.

**Dataset Features:**

* **Nodes:** Amino acids (**avg ~32 per graph**)
* **Edges:** Spatial connectivity (**~62 per graph**)
* **Node Features:** **21-dimensional feature vectors**
* **Graph Labels:** **One of six enzyme classes**

**Tasks to Complete**

You will complete the following tasks step by step:

**1. Setup Environment**

* Select the computation device (CPU/GPU).
* Install necessary Python packages.

**2. Data Preparation**

* Download the **ENZYMES dataset** using PyTorch Geometric.
* Perform **exploratory data analysis** (number of nodes, edges, features, labels).
* Split the dataset into **train (80%), validation (10%), and test (10%)** sets.

**3. Implement GCN Models**

**(a) GCN by Hand (Without Using PyG)**

* Manually define the **graph convolution operation**.
* Implement forward propagation using adjacency normalization and matrix multiplications.
* Use **global mean pooling** for graph-level classification.

**(b) GCN using PyG**

* Implement a GCN using torch\_geometric.nn.GCNConv.
* Apply **global mean pooling** for graph classification.

**4. Training Pipeline**

* Train **both models** on the dataset.
* Use a **separate dictionary to store hyperparameters** for easy tuning.
* Print training loss after each epoch.

**5. Model Evaluation**

* Evaluate model performance on the **test dataset**.
* Report **accuracy, precision, recall, and F1-score**.
* Generate and visualize a **confusion matrix**.
* If the problem is binary classification, also compute **ROC Curve & AUC Score**.

**6. Making Predictions & Visualizations**

* Generate predictions on test data.
* Display **predicted vs. true labels**.
* Plot:
  + **Confusion Matrix** (For all classes)
  + **ROC Curve & AUC Score** (For binary classification)

**Submission**

Submit the following:

1. **Jupyter Notebook (.ipynb)**: Include all the code and output.
2. **Results & Analysis**: Briefly describe key findings and observations in a markdown cell.