I'm planning to use GIN (Graph Isomorphism Network) for my Postgraduate Project B.

## What is Graph Isomorphism?

- Two graphs are isomorphic if there's a one-to-one mapping between their nodes such that edges are preserved.  
- Basically, they look different on paper but are actually the same structure.  
- Example: A triangle with nodes A-B-C is isomorphic to one with nodes X-Y-Z if connections are same.

## Why Use Machine Learning?

- Classical methods work (like VF2), but not always efficient for large graphs.  
- ML (especially Graph Neural Networks) can learn the patterns and structures without manually checking all permutations.  
- GNNs are now popular in bioinformatics, chemistry, social networks, etc.

## What is GIN (Graph Isomorphism Network)?

- GIN is a type of Graph Neural Network (GNN).  
- It's designed to be as powerful as the WL (Weisfeiler-Lehman) test, which is a standard for graph structure matching.  
- GIN uses SUM aggregation to gather info from neighboring nodes.  
- Uses MLP (multi-layer perceptron) to update node features.  
- Formula:  
 h\_v(k) = MLP((1 + eps) \* h\_v(k-1) + sum(h\_u(k-1))) for u in neighbors of v

## Dataset (Synthetic)

- I’m planning to start with synthetic datasets I created using NetworkX.  
- Each pair has 2 graphs:  
 - One isomorphic pair (same structure, shuffled node labels)  
 - One non-isomorphic pair (different structures)  
- Label = 1 for isomorphic, 0 for non-isomorphic  
- Planning to later move to real datasets like MUTAG, ENZYMES (from PyTorch Geometric).

## Implementation Plan

1. Use NetworkX to generate graph pairs.  
2. Convert them to PyTorch Geometric format.  
3. Build a GIN model (probably 3-5 layers).  
4. Train with cross-entropy loss.  
5. Evaluate with accuracy or F1 score.  
6. If time permits, compare GIN with other GNN models like GCN or GAT.

## My Goal

- To build a model that can take in two graphs and tell if they are isomorphic.  
- Understand how graph neural networks actually learn structural information.  
- Maybe try visualizing node embeddings after training.