🔍 Summary of What the Code and Model Do

The project builds a **recommendation system** that combines:

- **Graph Neural Networks (GNNs)** for learning from item-item relationships (global information), and
- **Propositional logic reasoning** for user-specific behavior patterns (local, symbolic reasoning).

This hybrid model is called **GNNLR** and is designed to make **more intelligent recommendations** by combining *pattern learning* with *logical inference*.

🔧 How Data Is Represented — Especially Graphs

📦 Input Data

- The dataset is typically from **MovieLens-100k** or Amazon reviews.
- Each user has a **sequence of interactions with items** (e.g., movies watched, products reviewed).

🔁 Graph Construction

Instead of traditional user-item bipartite graphs, the GNNLR model builds an **item-item graph**:

- For every user's interaction history (e.g., items `[v1, v4, v3]`), it creates **edges between adjacent items only**.
- This forms a **dense item-only graph**, where:
  - **Nodes** = items (e.g., movies, products)
  - **Edges** = adjacency in user history (e.g., if v1 and v4 are watched consecutively)
  - **Weights** = how frequently two items are adjacent across all users

📌 **Why this matters**: it captures **temporal and co-occurrence patterns**, while keeping the graph compact.

How the Formulas and Algorithms Are Applied

1. Graph Neural Network (GNN) Layer

Used to learn embeddings for each item node from the item-item graph.

Formula:

$$X' = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} X \Theta$$

Where:

- $\hat{A} = A + I$: adjacency matrix + self-loops
- $\hat{D}$: degree matrix
- X: item features
- Θ: learnable weights

➡️ This propagates information across neighboring items.

2. Propositional Logic Conversion

User's past interactions are converted into **logic expressions**:

If a user saw items `[v1, v4, v3]` and we are predicting their preference for v3:

$$(v1 \rightarrow v3) \lor (v4 \rightarrow v3) \lor (v1 \land v4 \rightarrow v3)$$

Using logic rules, this becomes:

$$(\neg v1 \lor v3) \lor (\neg v4 \lor v3) \lor (\neg v1 \lor \neg v4 \lor v3)$$

➡️ This structure can **reason** whether a user would logically prefer an item.

3. Neural Logic Modules

Instead of hard logic rules, logic operators like **¬ (NOT)** and **∨ (OR)** are implemented as **neural networks** (MLPs).

For NOT:

¬ei=MLP(ei)¬e_i = \text{MLP}(e_i)

For OR:

ei∨ej=MLP(ei⊕ej)e_i \vee e_j = \text{MLP}(e_i \oplus e_j)

These modules output a **vector representing the logic expression**, which is then compared with a **truth vector** to determine if the recommendation should be made.

4. Prediction and Training

A special vector *TT* represents "logical truth". If a logic expression evaluates to a vector close to *TT*, it's likely the user would like the item.

Final similarity score:

Sim(el,T)=sigmoid(ϕ·el·T∥el∥·∥T∥)\text{Sim}(e_l, T) = \text{sigmoid}\left(\phi \cdot \frac{e_l \cdot T}{\|e_l\| \cdot \|T\|} \right)

Loss includes:

- **Pairwise ranking loss**
- **Logic rule loss**
- **Regularization**

🧠 Overall Architecture in Plain Terms

1. **Data Input**: Loads user-item interaction data.

2. **Graph Construction**: Builds an item-only graph from user histories.
3. **Feature Learning**: Uses GNNs to embed each item in vector space.
4. **Logic Reasoning**:
    a. Converts user history into propositional logic
    b. Applies neural logic layers (NOT, OR) to reason over preferences
5. **Prediction**:
    a. Compares logic expression output to a "truth" vector
    b. Ranks items by how "true" the logic expression seems
6. **Training**:
    a. Optimizes using negative sampling and logic-aware loss

☑ Takeaways

- **GNNLR** = Graph Neural Network + Logic Reasoning
- **Innovative** in using logic expressions for explainable recommendation
- **Efficient graph**: item-item only, adjacency-based, temporal
- **Neural logic ops**: powerful, trainable NOT and OR networks
- **Flexible**: Can be swapped with different GNN types (GCN, GAT, etc.)