National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Nishant Nayak
Student ID: X22248242

School of Computing
National College of Ireland

Supervisor: Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Nishant Nayak

Student ID: X22248242

Programme: MSc in Data Analytics **Year:** 2024-2025

Module: MSc Research Project (MSCDAD_C)

Lecturer: Prof. Christian Horn

Submission Due Date: 11/08/2025

Project Title: Leveraging Heterogeneous GNNs for User Behaviour Analysis and Node Classification in Movie Recommender Systems

Word Count: 1227 **Page Count:** 9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Nishant Nayak

Date: 11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nishant Nayak
x22248242

1 Introduction

This configuration manual delineates comprehensive guidelines regarding the establishment of the operational environment, the installation of requisite dependencies, and the execution of the project entitled "Leveraging Heterogeneous GNNs for User Behaviour Analysis and Node Classification in Movie Recommender Systems." The objective is to promote reproducibility and guarantee the uninterrupted implementation of the developed models and analytical frameworks.

2 System Requirements

2.1 Hardware Requirements

Component	Minimum Requirement	Recommended	Notes
CPU	Intel i5	Intel i5 (4-core) or higher	For efficient data processing & training
RAM	8 GB	16 GB	For handling large datasets & models
GPU	Intel® UHD Graphics 630 , AMD	NVIDIA GPU with CUDA support	For faster model training (deep learning)

Figure 1: Hardware Configuration on Local Machine.

2.2 Software Requirements

Software/Tool	Version/Specification	Purpose / Description
Operating System	Windows 10/11, Ubuntu 20.04+	Core OS for running the project
Python	3.8 or higher	Main programming language
Jupyter Notebook / Lab	Latest (recommended)	Interactive development environment

3 Package Requirements

The Python packages were installed using pip and conda commands in Jupyter Notebook and Visual Studio Code, respectively. The packages installed are listed below:

- pandas
- numpy
- torch
- torch_geometric
- scikit-learn
- matplotlib
- seaborn
- IPython

The torch_geometric and related dependencies can be installed by following the instructions at:
https://pytorch-geometric.readthedocs.io/en/latest/notes/installation.html

Library Name	pd/Import Name	Description / Usage
pandas	pd	Data manipulation and analysis; reading and processing CSV/tabular data
numpy	np	Numerical operations, array processing, random number generation
torch	torch	PyTorch core for deep learning and tensor computations
torch.nn	nn	Neural network layers, loss functions, activation functions (PyTorch)
torch.nn.functional	F	Functional interface for neural network operations (e.g., relu, dropout)
torch_geometric.data	HeteroData	For creating and handling heterogeneous graph data structures
torch_geometric.nn	HeteroConv, SAGEConv, Linear	Graph neural network layers and architectures for GNN modeling
torch_geometric.transforms	RandomLinkSplit	Data transformation for train/val/test splitting in link prediction tasks
torchinfo	torchinfo	Model summary and architecture visualization (like model.summary() in Keras)
torch_geometric.transforms	RandomLinkSplit	Data transformation for train/val/test splitting in link prediction tasks
sklearn.metrics	roc_auc_score, accuracy_score, recall_score	Evaluation metrics for classification/recommendation tasks

Library Name	pd/Import Name	Description / Usage
sklearn.metrics	mean_absolute_error, mean_squared_error, r2_score	Regression metrics for model evaluation
sklearn.manifold	TSNE	Dimensionality reduction and visualization using t-SNE
sklearn.cluster	KMeans	Clustering algorithm for unsupervised learning (e.g., user/movie clusters)
sklearn.decomposition	PCA	Principal Component Analysis for reducing feature dimensions
sklearn.metrics.pairwise	cosine_similarity	Similarity computation between user/movie embeddings
matplotlib.pyplot	plt	Plotting graphs and visualizations
matplotlib.patches	mpatches	Custom legends, shapes in matplotlib plots
matplotlib.lines	mlines	Custom lines/markers in matplotlib plots
seaborn	sns	Advanced statistical data visualization built on matplotlib
IPython.display	display, HTML	Enhanced display of HTML and rich content in Jupyter Notebooks
os	os	Operating system interfaces; path, file operations

4 Dataset Description

The dataset utilized in this project consists of user movie ratings and metadata, which were obtained by downloading a zip file from a public source shown in Figure 2. After downloading, the dataset was unzipped and stored locally in the D:/ directory. The primary files used were movies.csv and ratings.csv, which were then cleaned to remove inconsistencies and missing values. Subsequently, these files were merged to form a single structured dataset suitable for analysis and model training. The preprocessing steps ensured that the data was ready for the feature engineering and modeling phases.

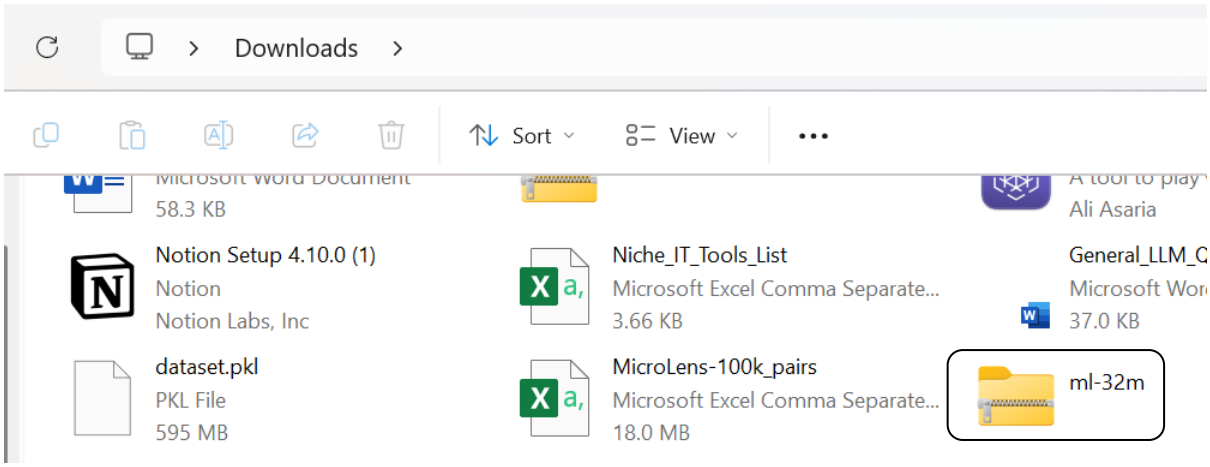

Fig 2 Download history

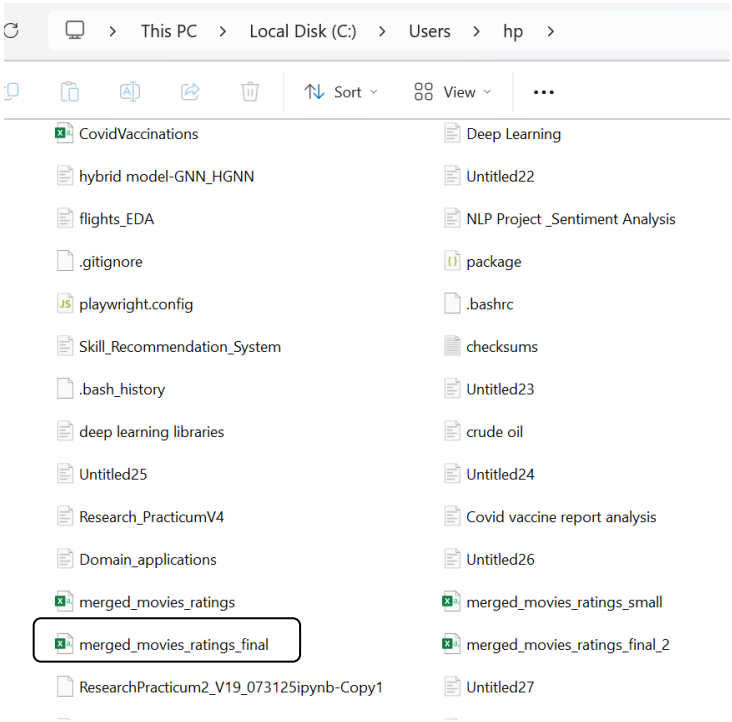

Fig 3 Notebook's local storage location

Working Directory: By default, Jupyter reads from the current working directory (CWD). To verify it:

```
import os
print(os.getcwd())
```

Ensure all dataset files are placed in the same folder as the notebook or update paths accordingly.

Download Dataset: https://files.grouplens.org/datasets/movielens/ml-32m.zip[1]

5 Installation

1. Create Virtual Environment (Optional but Recommended):
   python -m venv venv
   source venv/bin/activate # On Windows: venv\Scripts\activate

2. Install Required Packages: pip install pandas numpy scikit-learn matplotlib seaborn torch torch-geometric torchinfo

3. Verify Installation:

   ```
   import torch
   import torch_geometric
   import pandas as pd
   import numpy as np
   import sklearn
   import matplotlib
   import seaborn
   import torchinfo
   import IPython

   print("Torch:", torch.__version__)
   print("PyG:", torch_geometric.__version__)
   print("pandas:", pd.__version__)
   print("numpy:", np.__version__)
   print("sklearn:", sklearn.__version__)
   print("matplotlib:", matplotlib.__version__)
   print("seaborn:", seaborn.__version__)
   print("torchinfo:", torchinfo.__version__)
   print("IPython:", IPython.__version__)
   ```

4. Dataset Configuration

   - Dataset: MovieLens or custom interaction dataset (movies, ratings)
   - Extract Files: Unzip the downloaded file and place the CSVs in your working directory (e.g., D:/).
   - Data Cleaning: Remove or filter users with excessively high activity (e.g., more than 300 ratings) to avoid bias from "bulk voters".
   - Handle missing or inconsistent values in genres and other columns.
   - Data Merging: Merge movies.csv and ratings.csv on movieId to create a unified dataset containing both user interactions and movie features.

[1] https://files.grouplens.org/datasets/movielens/ml-32m.zip[1]

- Final Dataset Structure:
    - userId: Unique identifier for each user.
    - movieId: Unique identifier for each movie.
    - rating: User-assigned rating (usually 0.5-5.0).
    - timestamp: When the rating was given (may be omitted for modeling).
    - title: Movie title.
    - genres: Movie genres, pipe-separated.

Structure:

```
├── src/
│   ├── data_preprocessing/          # Data cleaning and merging scripts
│   ├── feature_engineering/         # Feature extraction & engineering
│   ├── model/                       # GNN model definition & training
│   ├── evaluation/                  # Evaluation metrics and plots
│   └── visualization/               # Embedding & cluster visualization
│
├── data/
│   ├── raw/                         # Raw datasets (unzipped .csv files)
│   │   ├── movies.csv
│   │   └── ratings.csv
│   ├── processed/                   # Cleaned & merged datasets
│   └── └── merged_movies_ratings_final.csv
│
├── notebooks/                       # Jupyter notebooks for EDA & prototyping
│   ├── data_analysis.ipynb
│   └── model_experiments_visualization.ipynb
│
├── results/                         # Plots, charts, and output metrics
│
├── requirements.txt                 # Project dependencies
└── README.md                        # Project overview and instructions
```

5. Model Configuration:
    - Model Type: Heterogeneous Graph Neural Network using PyTorch Geometric
    - Main Layers: HeteroConv, SAGEConv, Linear
    - Parameters:

Learning rate: 0.01
Epochs: 100
Hidden dimensions: 82

Model Building: Figure 4 illustrates the execution of the RecommenderGNN class, which delineates a Graph Neural Network (GNN) framework specifically designed for a recommendation system utilizing the PyTorch library. This architecture incorporates two layers of HeteroConv (heterogeneous graph convolution) employing SAGEConv operations to effectively capture the interrelations between users and films within a bipartite graph topology. Each layer adeptly processes user-movie and movie-user edges independently, thereby allowing the model to derive insights from interactions bidirectionally. Following each convolutional operation, a ReLU activation function along with dropout regularization is implemented to mitigate the risk of overfitting. The ultimate representations for each category of node (user and movie) are transformed to the output dimension via a linear mapping. The optimization algorithm employed is Adam, complemented by a learning rate scheduler that is instituted to modulate the learning rate throughout the training process. This class serves as the fundamental component of the recommendation model, facilitating the acquisition of intricate interaction patterns to enhance predictive accuracy.

```python
# ================== 5. GNN Model Definition (2 layers + Dropout) ================== #

class RecommenderGNN(nn.Module):
    def __init__(self, hidden_channels, out_channels, metadata, dropout_prob=0.3):
        super().__init__()
        self.conv1 = HeteroConv({
            ('user', 'rates', 'movie'): SAGEConv((-1, -1), hidden_channels),
            ('movie', 'rev_rates', 'user'): SAGEConv((-1, -1), hidden_channels),
        }, aggr='sum')
        self.conv2 = HeteroConv({
            ('user', 'rates', 'movie'): SAGEConv((-1, -1), hidden_channels),
            ('movie', 'rev_rates', 'user'): SAGEConv((-1, -1), hidden_channels),
        }, aggr='sum')
        self.lin = nn.ModuleDict({
            node_type: Linear(hidden_channels, out_channels)
            for node_type in metadata[0]
        })
        self.dropout = nn.Dropout(p=dropout_prob)

    def forward(self, x_dict, edge_index_dict):
        x_dict = self.conv1(x_dict, edge_index_dict)
        x_dict = {k: F.relu(self.dropout(v)) for k, v in x_dict.items()}
        x_dict = self.conv2(x_dict, edge_index_dict)
        x_dict = {k: F.relu(self.dropout(v)) for k, v in x_dict.items()}
        x_dict = {k: self.lin[k](v) for k, v in x_dict.items()}
        return x_dict

hidden_channels = embedding_dim
out_channels = embedding_dim
model = RecommenderGNN(hidden_channels, out_channels, data.metadata(), dropout_prob=0.3)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=30, gamma=0.5)  # <--- LR SCHEDULER
```

Fig 4 GNN Model Definition (RecommenderGNN Class) with Two SAGEConv Layers and Dropout

6. Visualization and Results

- PCA and t-SNE projections: Visualize user and movie embeddings.
- Cluster Analysis: K-Means and genre-dominant clusters.
- Performance Metrics: ROC-AUC, Accuracy, Recall.

7. Troubleshooting & Known Issues

- Ensure correct PyTorch + CUDA compatibility.
- Large datasets may require increased RAM or GPU resources.
- Use virtual environments to avoid dependency conflicts.

References

"The Python Language Reference," Python documentation. Accessed: Aug. 04, 2025. [Online]. Available: DOI

T. Kluyver et al., "Jupyter Notebooks—a publishing format for reproducible computational workflows," in IOS Press, 2016, pp. 87–90. DOI

"PyTorch: An Imperative Style, High-Performance Deep Learning Library | BibSonomy." Accessed: Aug. 04, 2025. [Online]. Available: DOI