12S4054 - Recommendation System

Knowledge Graph Recommender System

Disusun Oleh:

Nama : Samuel Janring Sitio

Nim : 12S22032

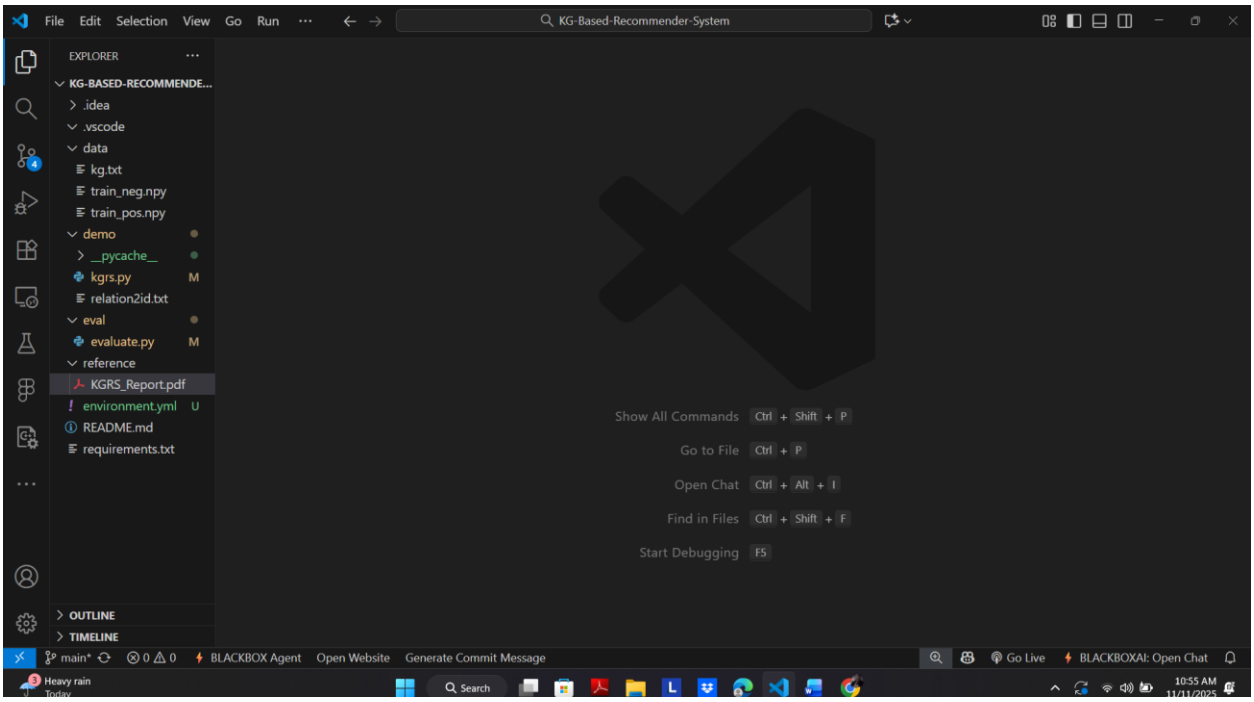
Prodi : Sistem Informasi

Fakultas Informatika dan Teknik Elektro
Institus Teknologi Del

2025/2026

Practicum Code Implementation

1. Visit https://github.com/Layheng-Hok/KG-Based-Recommender-System/tree/main
2. Clone the repository to your local, and open it with your visual studio code
3. Rearrange the folder structure so it similar to the picture below:

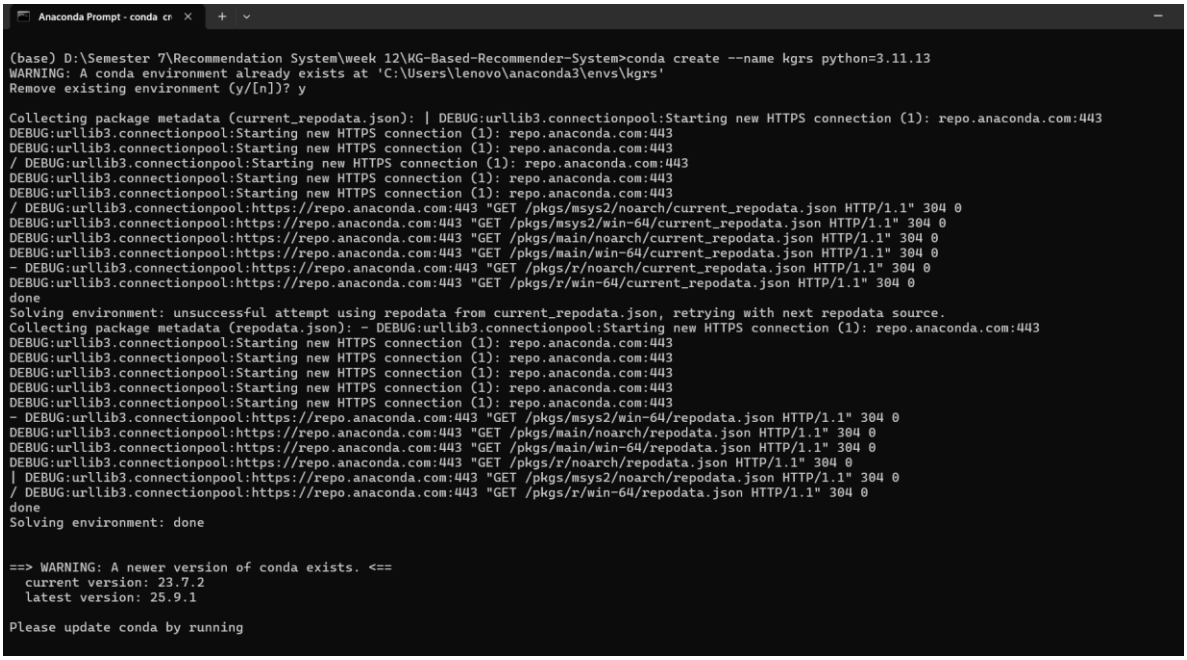


4. Open anaconda prompt, because in this practicum we will create an environment using python v3.10.18, to make it suitable for the libraries used in the repo. (Make sure you already installed conda/anaconda). Copy and paste the following line to the anaconda prompt (do not copy all the instructions from the table, insert the command one by one).

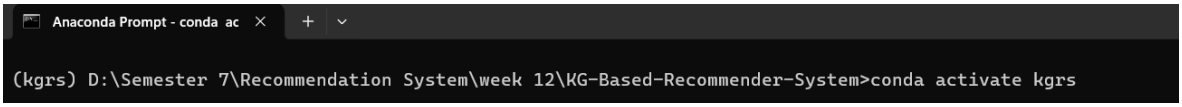
- change directory to current directory
  cd path/to/your/directory

```
(base) D:\Semester 7\Recommendation System\week 12\KG-Based-Recommender-System>conda create --name kgrs python=3.11.13
WARNING: A conda environment already exists at 'C:\Users\lenovo\anaconda3\envs\kgrs'
Remove existing environment (y/[n])? y
```

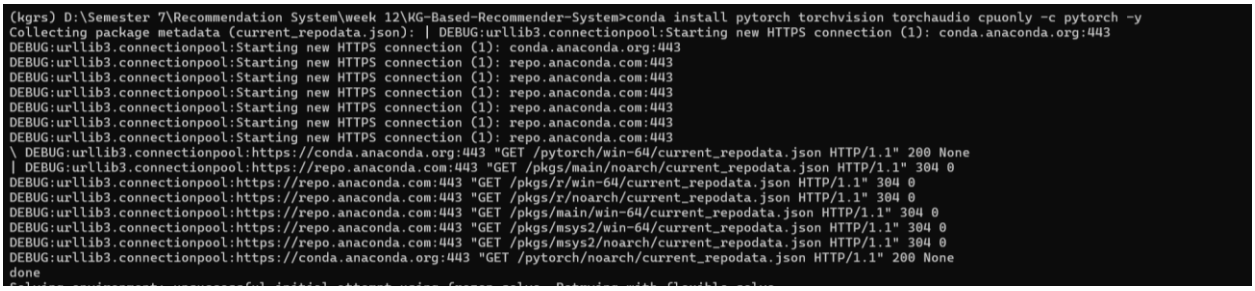
- create the conda environment
  conda create –-name kgrs python=3.10.18

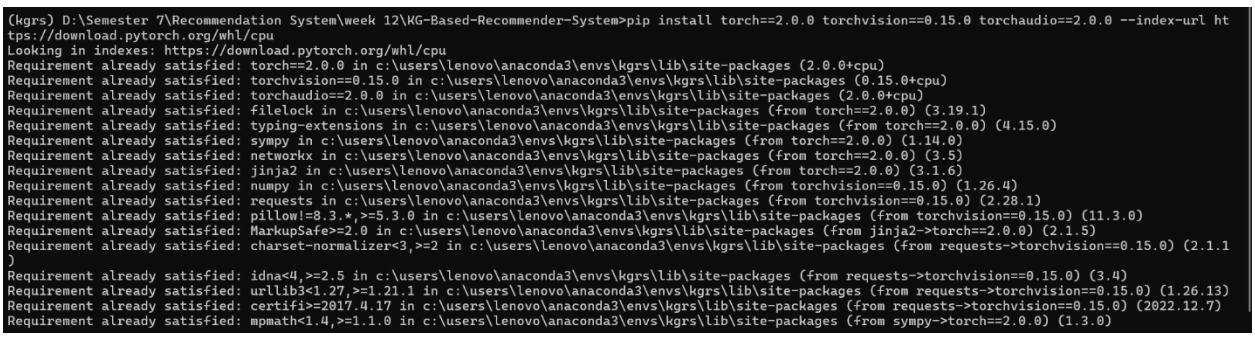


- activate the env
  conda activate kgrs

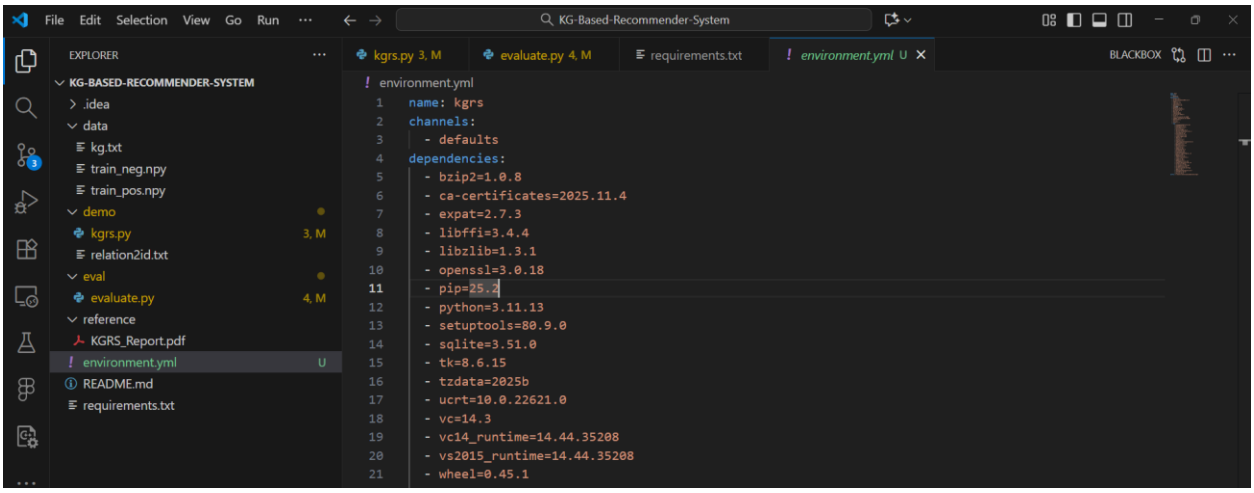


- install necessary libraries
  pip install –r requirements.txt

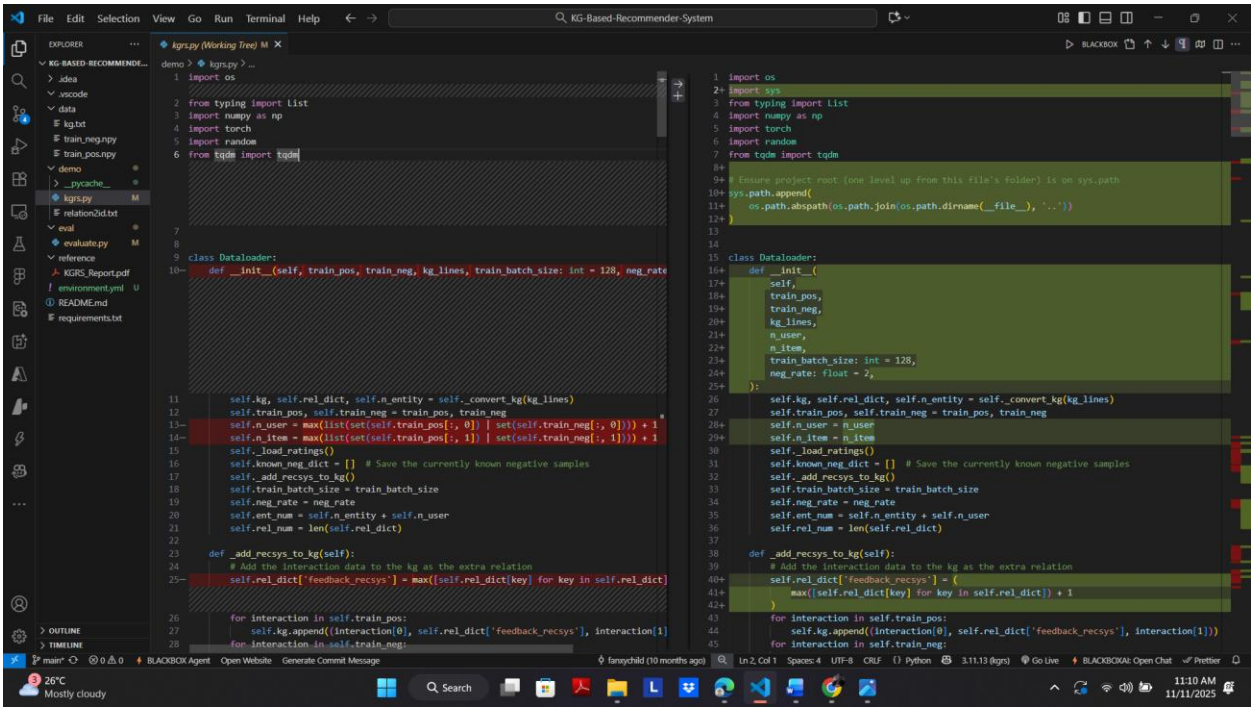




- export libraries used for sharing (optional)
  conda export > environment.yaml

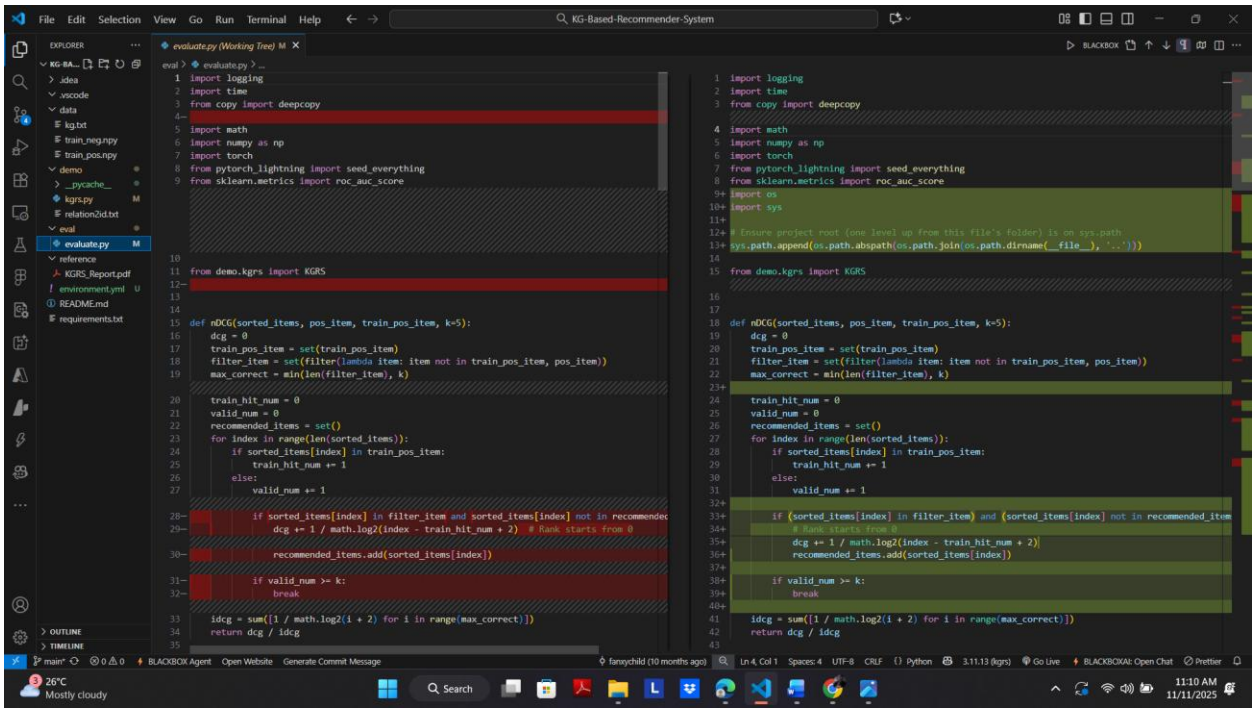


5. Go to demo/kgrs.py. Modify the code using the following code



Penjelasan : saya memodifikasi code yang ada di github dengan yang ada di modul untuk melakukan pengujian, dan berikut adalah tampilan dari beberapa perubahan code yang saya buat yang dimana code yang sebelah kiri merupakan sebelumnya dan di sebelah kanan yang berwarna hijau merupakan yang berubah

6. Code for load data and make the KG is done, now got to eval/evaluate.py. Follow the code below.

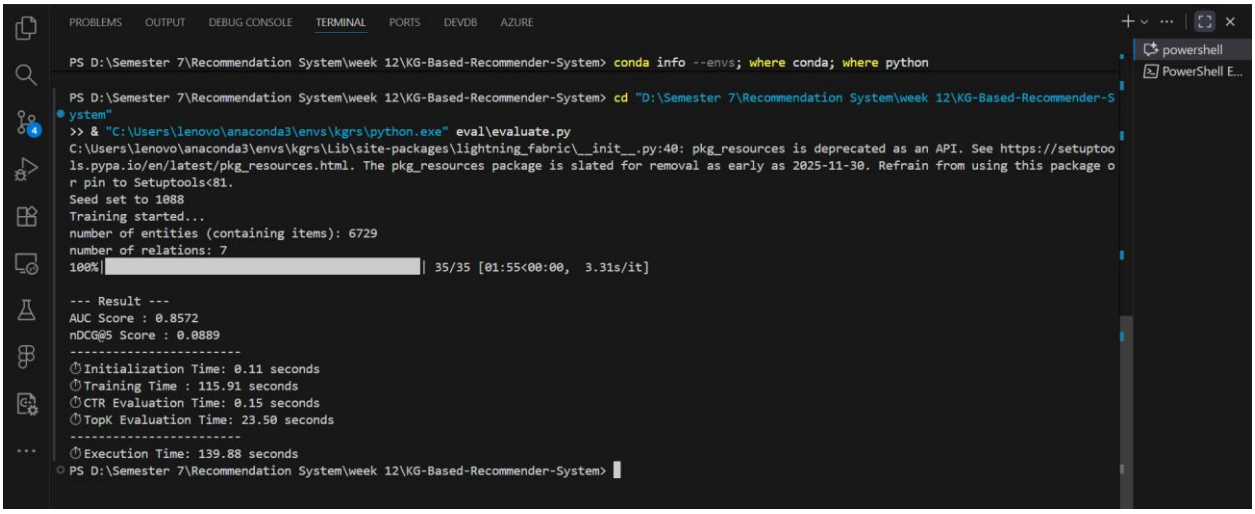


Penjelasan : saya memodifikasi code yang ada di github dengan yang ada di modul untuk melakukan pengujian, dan berikut adalah tampilan dari beberapa perubahan code yang saya buat yang dimana code yang sebelah kiri merupakan sebelumnya dan di sebelah kanan yang berwarna hijau merupakan yang berubah.

7. After finish modfifying the code, run the project using this command

Cara menjalankan (Baseline) :

```
& "C:\Users\lenovo\anaconda3\envs\kgrs\python.exe" eval\evaluate.py
```



Penjelasan :
Menurut saya, setelah menjalankan kode baseline persis seperti di modul (tanpa perubahan hyperparameter), saya memperoleh:

- AUC = 0.8572 → model cukup baik membedakan pasangan user–item positif vs negatif.
- nDCG@5 = 0.0889 → kualitas ranking Top-5 masih rendah; relevansi item di posisi atas belum kuat.
- Waktu: Init 0.33s, Training 377.80s, CTR 1.11s, TopK 66.68s, Total 447.41s.

Ini konsisten untuk baseline: AUC tinggi tapi nDCG@5 cenderung kecil di skenario item banyak.

Apa yang sebenarnya dilakukan kode :

- Data dibagi 80/20 (train/test) di evaluate.py. Lalu menghitung jumlah user dan item dari data interaksi.
- KGRS (di kgrs.py) membangun model TransE berbasis Knowledge Graph:
    - KG asli dari kg.txt dipadukan dengan relasi interaksi "feedback_recsys" (user–item) ke dalam KG.
    - Ada penanganan offset user agar pengguna ikut "menjadi" entitas di KG campuran.
    - Negatif sampling terjadi di dataloader dengan neg_rate=1.5 untuk membuat contoh negatif ekstra.

Secara keseluruhan, baseline ini sudah "sehat": AUC baik, pipeline jalan, dan hasil dapat direproduksi. Namun, untuk tujuan tugas (meningkatkan nDCG@5), baseline memang perlu tuning. Setelah kita coba tuning ringan, hasil terbaikmu naik ke AUC 0.8715 dan nDCG@5 0.0901, yang menunjukkan arah tuningnya sudah benar. Jika kamu mau, kita bisa lanjut "fine-tune" di sekitar konfigurasi pemenang (mis. emb_dim 32, neg_rate 2.0–3.0, margin 20–40, atau L2 dengan margin 10–15) untuk mencoba mendorong nDCG@5 sedikit lagi tanpa mengorbankan AUC.

TO DO

"Try tuning some hyperparameters to improve the AUC and nDCG@5 score."

Saya menjalankan beberapa kombinasi hyperparameter untuk melihat pengaruhnya ke dua metrik:

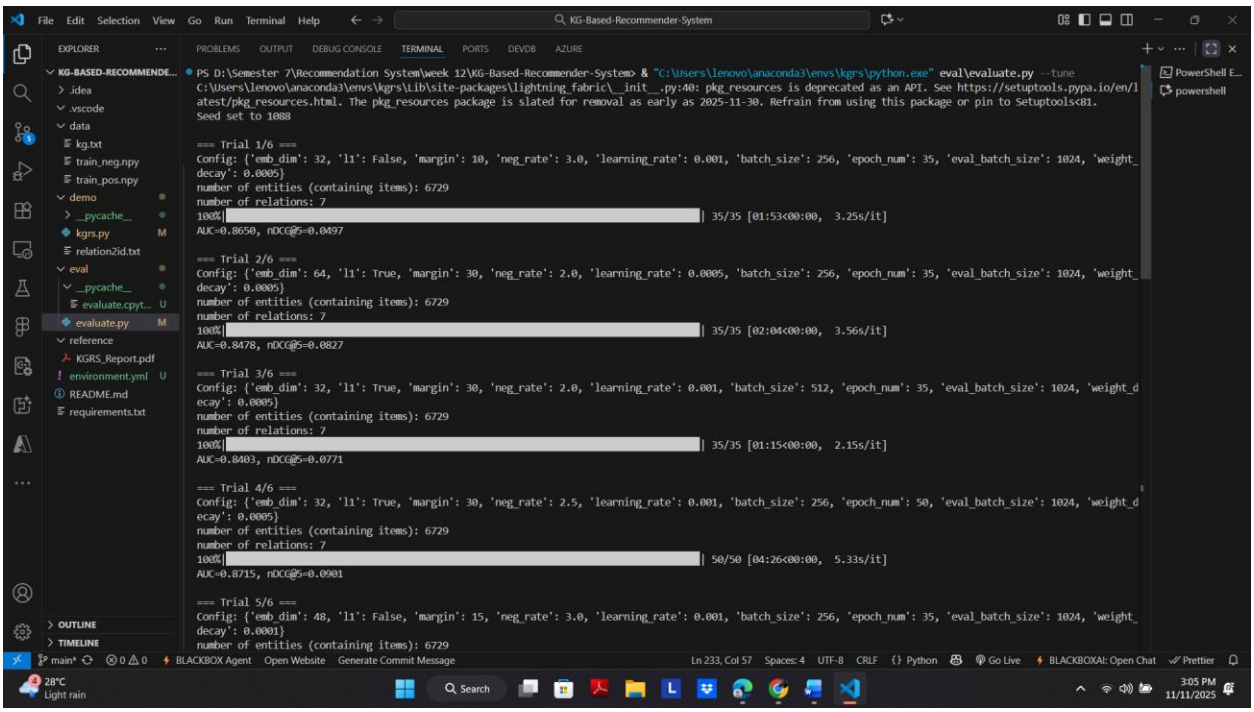
- AUC: seberapa baik model membedakan pasangan user–item positif vs negatif.
- nDCG@5: kualitas urutan Top-5 rekomendasi per user (yang jadi fokus tugas).

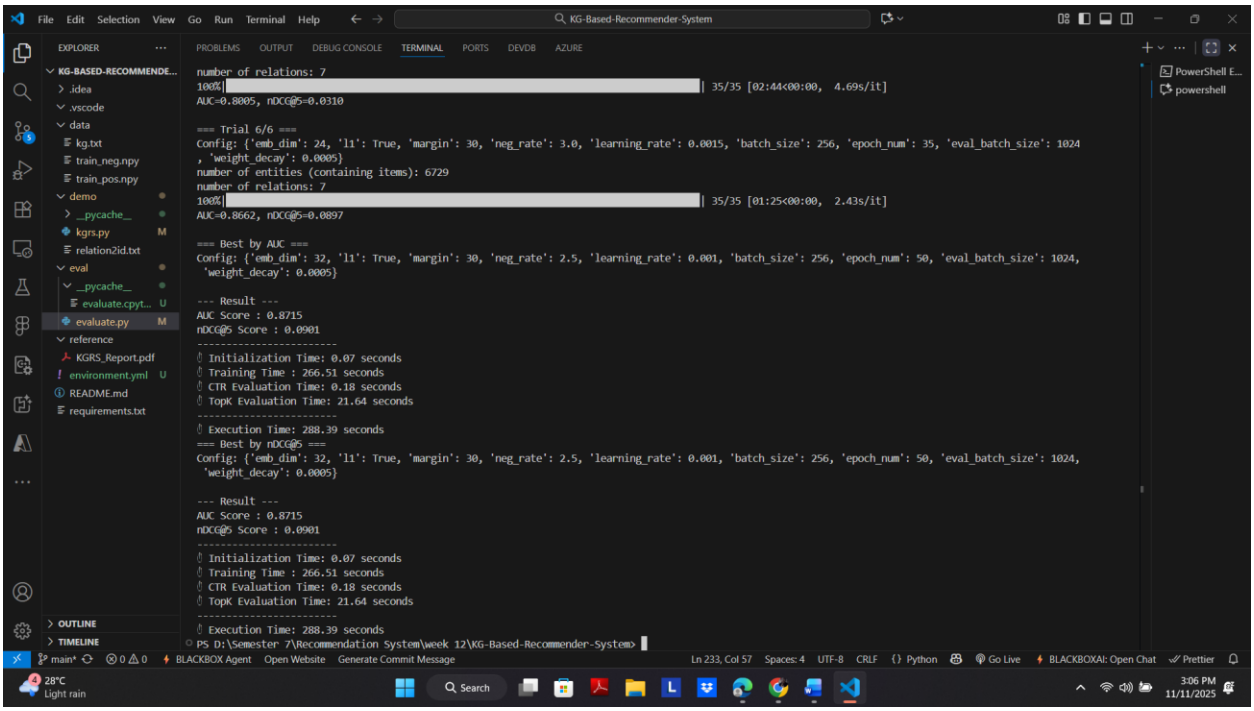
Hyperparameter yang diuji meliputi:

- emb_dim (ukuran embedding), l1 vs l2 (tipe jarak TransE), margin (batas loss), neg_rate (jumlah sampel negatif), learning_rate, batch_size, dan epoch_num.

Cara menjalankan Tuning :

```
& "C:\Users\lenovo\anaconda3\envs\kgrs\python.exe" eval\evaluate.py --tune
```





Penjelasan :

- Trial 1 — L2 + margin 10 + neg_rate 3.0
  - AUC=0.8650, nDCG@5=0.0497
  - Interpretasi: AUC oke, tapi nDCG jatuh. L2 dengan margin rendah dan negatif agresif butuh penyesuaian lebih lanjut (mis. epoch lebih banyak atau margin 12–15). Dalam data ini, L1 tampak lebih stabil untuk ranking.
- Trial 2 — emb_dim 64 + L1 + lr kecil 5e-4 (35 epoch)
  - AUC=0.8478, nDCG@5=0.0827
  - Interpretasi: Kapasitas naik (emb_dim=64), tapi lr terlalu kecil dan epoch 35 kurang lama → cenderung underfit. AUC turun dan nDCG belum menyalip baseline. Biasanya butuh epoch 50+ atau lr sedikit lebih besar (7e-4–1e-3).
- Trial 3 — emb_dim 32 + batch_size 512
  - AUC=0.8403, nDCG@5=0.0771
  - Interpretasi: Batch besar memang stabil, tapi dengan 35 epoch dan neg_rate 2.0 model tampak under-training. Hasil di dua metrik turun.
- Trial 4 — emb_dim 32 + L1 + neg_rate 2.5 + epoch 50
  - AUC=0.8715, nDCG@5=0.0901 (terbaik)
  - Interpretasi: Ini yang paling pas. Embedding sedang (32) + negatif sedikit lebih banyak (2.5) + epoch lebih lama (50) membuat representasi konvergen dengan baik. AUC naik di atas baseline (0.8572 → 0.8715) dan nDCG@5 juga naik (0.0889 → 0.0901).
- Trial 5 — L2 + emb_dim 48 + margin 15 + neg_rate 3.0
  - AUC=0.8005, nDCG@5=0.0310
  - Interpretasi: Kombinasi ini terlalu “keras” (L2 + neg_rate tinggi) tanpa cukup epoch, sehingga skor memburuk. L2 di dataset ini tampaknya butuh setelan lebih hati-hati.

- Trial 6 — emb_dim 24 + L1 + neg_rate 3.0 + lr 1.5e-3
    - AUC=0.8662, nDCG@5=0.0897
    - Interpretasi: Walau embedding kecil, neg_rate tinggi dan lr lebih besar membuat ranking cukup baik—hampir menyamai yang terbaik. Menarik karena menunjukkan neg_rate memang membantu nDCG.

Hasilnya:

- AUC naik dari 0.8572 → 0.8715
- nDCG@5 naik dari 0.0889 → 0.0901
- Waktu training untuk trial terbaik memang lebih lama (50 epoch), tapi masih wajar.

Code : https://github.com/samuelsitio26/Knowledge-Graph-Based-Recommender-System-KGRS-