End-to-End Recommendation Engine (SQL + PyTorch)

Overview

This project implements a realistic recommendation system to predict whether a user will click on an item. It combines a **lean SQL feature pipeline** with a **strong PyTorch neural model**.

Key points: - Raw data stored in SQLite - Minimal SQL for feature aggregation - Neural model with embeddings and MLP for interactions - Evaluation using ROC-AUC under realistic class imbalance

Project Structure

```
recommender-system/
├── sql/
│   ├── schema.sql          # Database schema
│   └── features.sql        # SQL feature view
├── data/
│   └── generate_data.py    # Synthetic data generator
├── dataset/
│   └── rec_dataset.py      # PyTorch Dataset
├── model/
│   └── recommender.py      # Neural model
├── training/
│   └── train.py            # Training loop
├── evaluation/
│   └── evaluate.py         # Evaluation metrics
├── config.py               # Hyperparameters
├── requirements.txt        # Dependencies
└── README.md
```

Database Schema

- **users**: demographic & account features
- **items**: item metadata
- **interactions**: user-item interactions with `clicked` label

Feature Engineering

- Minimal SQL features:
- Interaction counts
- User/item click-through rates (CTRs)

- Most modeling power comes from neural embeddings

Model

- User and item embeddings
- Concatenation with numeric features
- Multi-layer perceptron (MLP) for interaction modeling

Training & Evaluation

- Loss: Binary Cross Entropy with logits
- Optimizer: AdamW
- Metric: ROC-AUC

How to Run

```
# 1. Generate data and populate database
python data/generate_data.py

# 2. Train model
python training/train.py

# 3. Evaluate
python evaluation/evaluate.py
```

Extensions

- Time-based train/val splits
- Negative sampling
- Cold-start evaluation
- Feature ablation studies
- Model checkpoints and experiment tracking

Notes

This project is designed to mirror real-world recommendation systems. SQL handles basic aggregation, while the neural model learns latent user-item interactions. ROC-AUC evaluation ensures meaningful performance measurement under class imbalance.