```
DeepCARSKit/
├── conda/
│   └── ....
├── config.yaml
├── dataset/
│   ├── depaulmovie/
│   │   ├── depaulmovie.inter
│   │   └── ReadMe.html
```

Dữ liệu trong tệp **depaulmovie.inter** có cấu trúc như sau, với mỗi dòng đại diện cho một tương tác giữa người dùng và một bộ phim:
user_id: Mã định danh duy nhất cho người dùng.
item_id: Mã định danh cho bộ phim (có thể là mã IMDB).
rating: Điểm đánh giá của người dùng cho bộ phim (kiểu dữ liệu float).
time: Thời gian hoặc bối cảnh khi tương tác xảy ra (ví dụ: Weekday - ngày trong tuần).
location: Địa điểm nơi người dùng đã xem bộ phim (ví dụ: Cinema).
companion: Người dùng đi cùng xem phim (ví dụ: Alone - một mình).
contexts: Kết hợp của thời gian, địa điểm, và người đi cùng, dùng để biểu diễn ngữ cảnh toàn diện của tương tác.
uc_id: Mã định danh kết hợp giữa người dùng và ngữ cảnh tương tác (user-context ID).
Ví dụ:Dòng đầu tiên mô tả người dùng 1003 đã xem bộ phim với mã tt0454876, đánh giá bộ phim với điểm 1. Bộ phim được xem vào ngày trong tuần tại rạp chiếu phim, và người dùng đã đi xem phim một mình. Ngữ cảnh tương tác là Weekday_Cinema_Alone và mã kết hợp người dùng với ngữ cảnh là 1003_Weekday_Cinema_Alone.

```
│   └── tripadvisor/
│       ├── ReadMe.html
│       └── tripadvisor.inter
```

Dữ liệu trong tệp **tripadvisor.inter** có cấu trúc như sau:
user_id: Mã định danh duy nhất của người dùng dưới dạng chuỗi token.
item_id: Mã định danh của địa điểm hoặc dịch vụ trên TripAdvisor mà người dùng đã đánh giá.
rating: Đánh giá của người dùng cho địa điểm hoặc dịch vụ, có thể từ 1 đến 5 sao.
trip: Loại chuyến đi mà người dùng thực hiện (ví dụ: SOLO, FAMILY).
contexts: Bối cảnh liên quan đến chuyến đi (ví dụ: trong trường hợp này là "SOLO").
uc_id: Mã định danh kết hợp giữa người dùng và bối cảnh chuyến đi, được tạo bằng cách nối user_id và contexts.
Ví dụ về dữ liệu:B74836EEEB5B363EDE997A88C3B85A38, 92488, 4, FAMILY, FAMILY, B74836EEEB5B363EDE997A88C3B85A38_FAMILY
Người dùng với mã B74836EEEB5B363EDE997A88C3B85A38 đã đánh giá địa điểm hoặc dịch vụ có mã 92488 với 4 sao trong một chuyến đi gia đình.

```
├── deepcarskit/
│   ├── config/
│   │   └── configurator.py
```

Lớp **CARSConfig** mở rộng từ Config và bao gồm hai hàm chính:
_get_model_and_dataset: Xác định mô hình và dữ liệu từ tham số đầu vào hoặc cấu hình ngoài. Nếu không có tham số đầu vào, nó lấy giá trị từ cấu hình ngoài.
_get_final_config_dict: Kết hợp cấu hình nội bộ và ngoài, và chọn các chỉ số phù hợp dựa trên loại nhiệm vụ (xếp hạng hoặc lỗi).

```
├── data/
│   ├── dataloader/
│   │   └── general_dataloader.py
```

FullSortEvalDataLoader: Khởi tạo với các tham số cấu hình, tập dữ liệu, bộ chọn mẫu, và tùy chọn sắp xếp.
LabledDataSortEvalDataLoader: Tăng tốc quá trình tính toán bằng cách chỉ trả về phần tương tác của người dùng, các mục tích cực và đã sử dụng, không trả về các mục tiêu.
Khởi tạo: Nhận cấu hình, tập dữ liệu, bộ chọn mẫu, và tùy chọn sắp xếp. Xử lý dữ liệu tương tác của người dùng, phân loại các mục tích cực và lịch sử.
Phương thức chính:
_get_multidict: Tạo từ điển các mục tương tác của người dùng theo bối cảnh và đánh giá, phân loại các mục theo đánh giá giảm dần.
_next_batch_data: Trả về dữ liệu batch tiếp theo, bao gồm lịch sử và các mục tích cực.

```
│   ├── dataset/
│   │   └── dataset.py
│   └── utils.py
```

Cuối file doc

create_dataset(config):Tạo đối tượng dataset dựa trên cấu hình và loại mô hình.
save_split_dataloaders(config, dataloaders):Lưu các dataloader đã phân chia vào một tệp.Đường dẫn lưu tệp được xác định bởi cấu hình và tên mô hình.
load_split_dataloaders(saved_dataloaders_file):Tải các dataloader đã phân chia từ tệp lưu.
data_preparation(config, dataset, save=False):Phân chia dataset thành các tập huấn luyện, xác thực, và kiểm tra.Tạo dataloader cho từng giai đoạn và tùy chọn lưu kết quả phân chia.
get_dataloader(config, phase):Trả về lớp dataloader tương ứng với mô hình và giai đoạn ('train' hoặc 'evaluation').
get_used_ids(config, dataset):Trả về các ID mục đã sử dụng cho mỗi người dùng, để phục vụ cho việc tạo mẫu tiêu cực.
get_AE_dataloader(config, phase):Hàm tùy chỉnh để lấy lớp dataloader cho các mô hình VAE dựa trên giai đoạn.
create_samplers(config, dataset, built_datasets):Tạo các bộ chọn mẫu (sampler) cho huấn luyện, xác thực và kiểm tra dựa trên cấu hình và dữ liệu đã phân chia.

```
├── evaluator/
│   ├── base_metric.py
```

AbstractMetric:Lớp cơ sở cho tất cả các chỉ số metric.
TopkMetric: Lớp cơ sở cho các chỉ số top-k.
Phương thức used_info lấy ma trận bool cho các mục tích cực.
Phương thức topk_result ánh xạ giá trị metric theo k vào dạng từ điển.
Phương thức metric_info tính giá trị của metric (phải được triển khai).
LossMetric:Lớp cơ sở cho các chỉ số dựa trên tổn thất và AUC.
Phương thức used_info lấy điểm dự đoán và nhãn thực.
Phương thức output_metric tính toán và trả về giá trị metric.
Phương thức metric_info tính giá trị metric dựa trên điểm dự đoán và nhãn thực (phải được triển khai).

```
│   ├── collector.py
```

CARSCollector : Thu thập tài nguyên đánh giá từ dữ liệu và mô hình.
Phương thức eval_batch_collect: Thu thập và cập nhật các dữ liệu đánh giá như item top-k, điểm số, và nhãn dữ liệu từ đầu ra của mô hình và dữ liệu batch.

```
│   └── evaluator.py
```

class Evaluator(object):: Kiểm tra độ chính xác của tham số và tổng hợp kết quả các chỉ số.
Phương thức evaluate: Tính toán tất cả các chỉ số dựa trên dữ liệu đầu vào, cập nhật kết quả vào từ điển, và tính thêm chỉ số F1 nếu cần.

```
├── model/
│   ├── ae/
│   ├── context_recommender.py
```

ContextRecommender Methods:
__init__: Khởi tạo các tham số và cấu trúc cho mô hình gợi ý dựa trên ngữ cảnh, bao gồm việc xác định các trường dữ liệu và cấu hình embedding.
embed_float_fields: Nhúng các cột dữ liệu số thành tensor embedding với kích thước embedding_size.
getContextSituationList: Trả về danh sách các tình huống ngữ cảnh từ dữ liệu tương tác.
getContextSituationDict: Trả về từ điển các tình huống ngữ cảnh từ dữ liệu tương tác.
embed_token_fields: Nhúng các cột dữ liệu token thành tensor embedding.
embed_token_seq_fields: Nhúng các cột dữ liệu chuỗi token và tổng hợp chúng theo phương thức chỉ định (max, sum, mean).
double_tower_embed_input_fields: Nhúng các trường dữ liệu theo cách double tower, phân chia thành các phần cho người dùng và mục tiêu.
concat_embed_input_fields: Kết hợp tất cả các embedding của các trường dữ liệu thành một tensor duy nhất.
embed_input_fields: Nhúng tất cả các trường dữ liệu (cả token và số) và trả về tensor embedding cho từng loại trường.

```
│   ├── fms/
│   │   ├── deepfm.py
```

DeepFM Methods:
__init__: Khởi tạo mô hình DeepFM với các lớp FM, DNN, và các tham số như kích thước ẩn của MLP và xác suất dropout.
_init_weights: Khởi tạo trọng số cho các lớp nhúng và tuyến tính bằng phương pháp Xavier và đặt bias bằng 0.
forward: Tính toán dự đoán bằng cách kết hợp các embedding từ FM và DNN, bao gồm cả lớp tuyến tính cuối cùng.
calculate_loss: Tính toán hàm mất mát dựa trên dự đoán của mô hình và nhãn thực tế từ dữ liệu tương tác.
predict: Dự đoán đầu ra bằng cách gọi phương thức forward.

```
│   │   └── fm.py
```

FM Methods:
__init__: Khởi tạo mô hình Factorization Machine (FM) với các lớp và hàm mất mát phù hợp dựa trên loại đánh giá (ranking hay rating).
_init_weights: Khởi tạo trọng số cho các lớp nhúng bằng phương pháp Xavier.
forward: Tiến hành tính toán dự đoán bằng cách kết hợp các embedding đầu vào và tính toán với lớp FM cùng với tuyến tính đầu tiên.
calculate_loss: Tính toán hàm mất mát dựa trên dự đoán của mô hình và nhãn thực tế từ dữ liệu tương tác.
predict: Dự đoán đầu ra bằng cách gọi phương thức forward.

```
│   ├── layers.py
```

Lớp này khởi tạo các bảng nhúng cho các loại đặc trưng khác nhau (token, chuỗi token, số thực) và xác định trường nhãn dựa trên cấu hình. Nó cũng bao gồm việc khởi tạo các trọng số và bias, với bias được định nghĩa là một tham số học được.

```
│   └── neucf/
│       ├── neucmf0i.py
```

các biến thể của mô hình NeuCF (Neural Collaborative Filtering).

```
│       └── neucmfww.py
├── properties/
│   ├── model/
│   │   ├── AFM.yaml
│   │   ├── AutoInt.yaml
```

```
│   │   │   ├── .....
│   │   │   └── xgboost.yaml
│   │   ├── overall.yaml
│   │   └── quick_start_config/
│   │       ├── context-aware.yaml
│   │       ├── knowledge_base.yaml
│   │       ├── sequential.yaml
│   │       └── sequential_embedding_model.yaml
│   ├── quick_start/
│   │   └── quick_start.py
```

eval_folds: Hàm này đánh giá mô hình qua các fold của cross-validation, huấn luyện mô hình trên tập huấn luyện và kiểm tra trên tập validation tương ứng, rồi trả về điểm số và kết quả tốt nhất của fold đó.
run: Hàm này thực hiện toàn bộ quy trình huấn luyện và đánh giá mô hình trên dữ liệu, bao gồm chuẩn bị dữ liệu, huấn luyện mô hình, và ghi log kết quả, hỗ trợ cả cross-validation và huấn luyện đơn lẻ.
update_best_log: Hàm này so sánh kết quả mới với các log cũ để xác định nếu kết quả mới tốt hơn, cập nhật hoặc lưu lại log mới nếu nó có hiệu suất tốt hơn.
objective_function: Hàm này dùng trong việc tinh chỉnh siêu tham số, thực hiện huấn luyện và đánh giá mô hình với cấu hình đã cho, và trả về điểm số và kết quả validation tốt nhất.
load_data_and_model: Hàm này tải mô hình đã lưu và tùy chọn là dữ liệu và dataloaders. Nó kiểm tra sự tồn tại của các tập tin và tải hoặc tạo chúng, sau đó nạp trạng thái mô hình từ tập tin và trả về các thành phần cần thiết.

```
│   ├── trainer/
│   │   └── trainer.py
```

__init__: Hàm khởi tạo của lớp CARSTrainer, thiết lập các đối tượng cần thiết cho việc thu thập đánh giá và đánh giá mô hình, bao gồm eval_collector và evaluator.
_labled_data_sort_batch_eval: Hàm xử lý đánh giá dữ liệu theo batch cho các dữ liệu đã được gán nhãn. Nó tính toán điểm số dự đoán cho các item trong batch và loại trừ các item không cần thiết từ lịch sử của người dùng.
evaluate: Hàm đánh giá mô hình trên dữ liệu đánh giá. Nó nạp mô hình tốt nhất nếu cần, chọn phương pháp đánh giá dựa trên loại dữ liệu, và trả về kết quả đánh giá dưới dạng từ điển các chỉ số.

```
│   └── utils/
│       ├── logger.py
```

RemoveColorFilter: Lớp này kế thừa từ logging.Filter, được sử dụng để loại bỏ mã màu ANSI khỏi thông điệp log trước khi ghi vào file log.
set_color: Hàm này áp dụng màu sắc và tô sáng cho thông điệp log dựa trên màu được chọn từ danh sách màu. Nó trả về thông điệp log được định dạng với mã màu ANSI.
init_logger: Hàm khởi tạo một logger để hiển thị thông báo trên màn hình và đồng thời ghi nó vào file log. Nó định dạng các thông báo log với màu sắc và thiết lập mức độ log dựa trên cấu hình.

```
│       └── utils.py
```

get_model: Hàm này tự động chọn lớp mô hình dựa trên tên mô hình (model_name). Nó tìm kiếm mô hình trong hai nhóm module: deepcarskit.model và recbole.model. Nếu không tìm thấy mô hình trong nhóm đầu tiên, nó sẽ tìm trong nhóm thứ hai. Nếu không tìm thấy mô hình nào phù hợp, nó sẽ ném ra lỗi ValueError.
get_trainer: Hàm này tự động chọn lớp huấn luyện viên (trainer) dựa trên loại mô hình (model_type) và tên mô hình (model_name). Nó cố gắng nhập lớp huấn luyện viên từ module deepcarskit.trainer. Nếu không tìm thấy, nó sẽ chọn lớp huấn luyện viên phù hợp từ module recbole.trainer dựa trên loại mô hình.
model_submodule_recbole = ['general_recommender', 'sequential_recommender', 'knowledge_aware_recommender', 'exlib_recommender']
model_submodule_deepcarskit = ['ae', 'fms', 'neucf']

```
├── images/
│   ├── intro-img1.jpg
│   └── NeuCMF.png
├── log/
│   ├── best/
│   │   └── tripadvisor... .log
│   ├── tripadvisor-NeuCMFii-Aug-19-2024_17-18-20.log
```

Thư mục log/ chứa các file log của quá trình huấn luyện, trong đó có thư mục best/ để lưu các mô hình tốt nhất và log tương ứng.
Thư mục log_tensorboard/ lưu các dữ liệu phục vụ cho TensorBoard để trực quan hóa quá trình huấn luyện.

```
├── log_tensorboard/
│   └── tripadvisor-NeuCMFii-Aug-20-2024_11-07-12/
│       ├── 1724126862.3869784/
│       │   └──
├── requirements.txt
├── run.py
├── saved/
│   ├── NeuCMFii-Aug-20-2024_11-07-24_f1.pth
│   ├── NeuCMFii-Aug-20-2024_11-07-24_f2.pth
│   └── ....
├── setup.py
```

cấu hình cho việc cài đặt gói deepcarskit, bao gồm thông tin về yêu cầu thư viện phụ thuộc (install_requires), cài đặt bổ sung (extras_require), và các thuộc tính khác như tên gói, phiên bản, mô tả, và tác giả.

```
└── style.cfg
```

Dataset.py

class Dataset(object):
_get_innerid_from_rawid(self, field, rawid):Trả về innerid tương ứng với rawid trong từ điển field2token_id của trường cụ thể.
_get_rawid_from_innerid(self, field, innerid):Trả về rawid tương ứng với innerid trong từ điển field2token_id của trường cụ thể.
_get_uid_from_usercontexts(self, innerid_usercontexts): Lấy uid từ innerid_usercontexts bằng cách phân tích chuỗi ngữ cảnh người dùng.
_get_context_tuple_from_usercontexts(self, innerid_usercontexts): Chuyển đổi innerid_usercontexts thành tuple chứa các giá trị ngữ cảnh.
_get_context_fields(self): Trả về danh sách các cột ngữ cảnh, ngoại trừ uid, iid, ucid, label, và rating.
_from_scratch(self): Tải và xử lý dữ liệu từ đầu, khởi tạo các thuộc tính và nạp dữ liệu từ file.
_get_preset(self): Khởi tạo các thuộc tính cần thiết từ cấu hình như đường dẫn dữ liệu và thông tin các trường dữ liệu.
_get_field_from_config(self): Khởi tạo tên các trường dữ liệu chung từ cấu hình, bao gồm các trường như USER_ID_FIELD, ITEM_ID_FIELD, LABEL_FIELD, và TIME_FIELD.
_data_processing(self): Tiền xử lý dữ liệu, bao gồm lọc dữ liệu, ánh xạ lại ID, xử lý giá trị thiếu, chuẩn hóa và khởi tạo trọng số tải trước.
_data_filtering(self): Lọc dữ liệu, bao gồm loại bỏ các user_id hoặc item_id bị thiếu, loại bỏ các tương tác trùng lặp, và lọc dữ liệu theo giá trị.
_build_feat_name_list(self): Xây dựng danh sách tên các đặc trưng (feat_name_list) từ dữ liệu đã tải lên.
_get_download_url(self, url_file, allow_none=False): Trả về URL tải dữ liệu từ file cấu hình YAML hoặc None nếu cho phép.
_download(self): Tải dữ liệu từ URL, giải nén và đổi tên tập tin nếu cần.
_load_data(self, token, dataset_path): Tải dữ liệu bao gồm các đặc trưng tương tác, người dùng và item từ thư mục dữ liệu.
_load_inter_feat(self, token, dataset_path): Tải các đặc trưng tương tác từ file, xử lý dữ liệu nếu danh sách file benchmark được cung cấp.
_load_user_or_item_feat(self, token, dataset_path, source, field_name): Tải các đặc trưng của người dùng hoặc item từ file, xác minh sự tồn tại của các trường cần thiết.
_load_additional_feat(self, token, dataset_path): Tải các đặc trưng bổ sung nếu có chỉ định trong cấu hình.
_get_load_and_unload_col(self, source): Phân tích cấu hình để xác định các cột cần tải và không tải theo nguồn dữ liệu.

_load_feat(self, filepath, source): Tải các đặc trưng từ file vào DataFrame, xác định kiểu dữ liệu và cột cần tải.
_set_alias(self, alias_name, default_value): Cài đặt alias cho các trường dữ liệu dựa trên cấu hình.
_init_alias(self): Khởi tạo alias cho user_id và item_id, kiểm tra tính hợp lệ của alias và cập nhật các trường còn lại.
_user_item_feat_preparation(self): Sắp xếp user_feat và item_feat theo user_id và item_id, điền giá trị thiếu sau.
_preload_weight_matrix(self): Chuyển các đặc trưng trọng số tải trước thành ma trận numpy.ndarray với kích thước phù hợp.
_fill_nan(self): Điền giá trị thiếu trong các trường dữ liệu với giá trị mặc định hoặc trung bình của dữ liệu gốc.
_normalize(self): Chuẩn hóa các trường dữ liệu float theo công thức chuẩn hóa, nếu cấu hình yêu cầu.
_filter_nan_user_or_item(self): Loại bỏ các dòng có giá trị NaN trong các trường user_id và item_id của user_feat và item_feat. Cũng loại bỏ các dòng tương ứng trong inter_feat nếu user_id hoặc item_id bị thiếu.
_remove_duplication(self): Loại bỏ các tương tác trùng lặp trong inter_feat. Nếu có trường time_field, dữ liệu sẽ được sắp xếp theo thứ tự tăng dần của trường này trước khi loại bỏ trùng lặp.
_filter_by_inter_num(self): Lọc người dùng và mặt hàng dựa trên số lượng tương tác của chúng. Giữ lại chỉ những người dùng và mặt hàng có số lượng tương tác trong khoảng được chỉ định. _get_illegal_ids_by_inter_num(self, field, feat, inter_num, inter_interval=None): Xác định các ID không hợp lệ dựa trên số lượng tương tác, nếu số lượng tương tác nằm ngoài khoảng cho phép. Trả về một tập hợp các ID không hợp lệ.
_parse_intervals_str(self, intervals_str): Chuyển đổi chuỗi khoảng cách thành danh sách các khoảng, mỗi khoảng được định dạng bằng tuple với các điểm đầu và cuối.
_within_intervals(self, num, intervals): Kiểm tra xem một số có nằm trong các khoảng đã cho không.
_filter_by_field_value(self): Lọc các thuộc tính dựa trên giá trị của chúng, chỉ giữ lại những giá trị nằm trong các khoảng giá trị đã định.
_reset_index(self): Đặt lại chỉ số cho tất cả các thuộc tính trong feat_name_list.
_del_col(self, feat, field): Xóa một cột khỏi feat và cập nhật các từ điển liên quan.
_filter_inter_by_user_or_item(self): Loại bỏ các tương tác mà người dùng hoặc mặt hàng không có trong các thuộc tính tương ứng.
_set_label_by_threshold(self): Tạo nhãn 0/1 cho các tương tác dựa trên ngưỡng giá trị từ config['threshold'] và xóa cột giá trị nếu cần.
_get_remap_list(self, field_list): Tạo danh sách ánh xạ cho các thuộc tính trong cùng một không gian ánh xạ.
_remap_ID_all(self): Ánh xạ lại tất cả các thuộc tính kiểu token theo alias và các thuộc tính còn lại.
_concat_remaped_tokens(self, remap_list): Nối các giá trị token từ remap_list và trả về các token nối cùng điểm phân tách.
_remap(self, remap_list): Ánh xạ lại các token sử dụng pandas.factorize và cập nhật các thuộc tính với ID mới.
_change_feat_format(self): Đổi định dạng của các thuộc tính từ pandas.DataFrame sang Interaction.
unique(self, field): Trả về các giá trị duy nhất của một trường sau khi ánh xạ.
num(self, field): Trả về số lượng token khác nhau cho các trường kiểu token hoặc 1 cho các trường kiểu số thực.
fields(self, ftype=None, source=None): Trả về danh sách các tên trường dựa trên loại và nguồn.
float_like_fields: Trả về danh sách các trường kiểu số thực hoặc dãy số thực.
token_like_fields: Trả về danh sách các trường kiểu token hoặc dãy token.
seq_fields: Trả về danh sách các trường kiểu dãy token hoặc dãy số thực.
non_seq_fields: Trả về danh sách các trường kiểu token hoặc số thực không phải dãy.
set_field_property(self, field, field_type, field_source, field_seqlen): Đặt thuộc tính mới cho một trường.
copy_field_property(self, dest_field, source_field): Sao chép thuộc tính từ trường nguồn sang trường đích.
field2feats(self, field): Trả về các thuộc tính liên quan đến trường cụ thể.
user_counter: Trả về bộ đếm số lần xuất hiện của các người dùng khác nhau trong inter_feat.
item_counter: Trả về bộ đếm số lần xuất hiện của các mục khác nhau trong inter_feat.
user_num: Trả về số lượng token khác nhau của trường người dùng (uid_field).
item_num: Trả về số lượng token khác nhau của trường mục (iid_field).
user_context_num: Trả về số lượng token khác nhau của trường ngữ cảnh người dùng (ucid_field).
inter_num: Trả về số lượng bản ghi tương tác.
avg_actions_of_users: Trả về số trung bình các bản ghi tương tác của người dùng.
avg_actions_of_items: Trả về số trung bình các bản ghi tương tác của các mục.
avg_actions_of_user_context: Trả về số trung bình các bản ghi tương tác của ngữ cảnh người dùng.
sparsity: Trả về độ thưa thớt của tập dữ liệu, tính bằng 1 trừ đi tỷ lệ tương tác so với số lượng người dùng và mục.

_check_field: Kiểm tra sự tồn tại của các thuộc tính được chỉ định.
join: Kết hợp các tính năng người dùng/mục vào một DataFrame tương tác, nếu có.
__getitem__: Trả về một phần của inter_feat dựa trên chỉ mục, với tùy chọn kết hợp các tính năng người dùng/mục.
__len__: Trả về số lượng bản ghi trong inter_feat.
__repr__ và __str__: Cung cấp chuỗi mô tả cho đối tượng, bao gồm thông tin về số lượng người dùng, mục, bản ghi tương tác, độ thưa thớt của tập dữ liệu và các trường còn lại.
copy: Tạo một bản sao của đối tượng Dataset với tính năng tương tác mới.
_drop_unused_col: Xóa các cột không sử dụng trong mô hình, dựa trên cấu hình unused_col.
_grouped_index: Tạo chỉ số cho các nhóm dựa trên danh sách các khóa, trả về danh sách các chỉ số nhóm.
_calcu_split_ids: Tính toán số lượng phần tử trong mỗi phần khi chia tổng số theo tỷ lệ đã cho. Phần đầu tiên được điều chỉnh để đảm bảo tổng số phần tử chính xác.
split_by_ratio: Chia các bản ghi tương tác thành nhiều phần theo tỷ lệ cho trước, với tùy chọn nhóm theo một trường cụ thể trước khi chia. Trả về danh sách các đối tượng Dataset sau khi chia.
split_by_folds: Chia các bản ghi tương tác thành nhiều phần theo số lượng nếp gấp (folds) cho trước, với tùy chọn nhóm theo một trường cụ thể. Trả về một từ điển chứa các đối tượng Dataset cho từng fold.
_split_index_by_leave_one_out: Chia các chỉ mục theo chiến lược "leave one out", trong đó một số phần tử được giữ lại riêng biệt. Trả về danh sách các chỉ mục sau khi chia.
leave_one_out: Chia các bản ghi tương tác theo chiến lược "leave one out" với chế độ "valid_and_test", "valid_only" hoặc "test_only". Trả về danh sách các đối tượng Dataset sau khi chia.
shuffle: Xáo trộn các bản ghi tương tác tại chỗ.
sort: Sắp xếp các bản ghi tương tác theo trường chỉ định, có thể sắp xếp tăng dần hoặc giảm dần.
build: Xử lý dataset theo các cài đặt đánh giá, bao gồm nhóm, thứ tự và phân chia. Trả về danh sách các đối tượng Dataset đã được xử lý.
save: Lưu đối tượng Dataset vào thư mục checkpoint đã cấu hình, sử dụng định dạng tệp .pth.
get_user_feature: Trả về các đặc trưng người dùng dưới dạng đối tượng Interaction. Nếu không có đặc trưng người dùng, tạo một đặc trưng mặc định.
get_item_feature: Trả về các đặc trưng mặt hàng dưới dạng đối tượng Interaction. Nếu không có đặc trưng mặt hàng, tạo một đặc trưng mặc định.
_create_sparse_matrix: Tạo ma trận thưa từ hai trường nguồn và đích. Ma trận có kích thước (số lượng trường nguồn, số lượng trường đích). Có thể chỉ định định dạng ma trận (coo hoặc csr) và trường giá trị.
_create_graph: Tạo đồ thị mô tả mối quan hệ giữa hai trường nguồn và đích, với định dạng có thể là DGL hoặc PyG. Có thể thêm thuộc tính cho các cạnh nếu có trường giá trị.
inter_matrix: Tạo ma trận thưa mô tả tương tác giữa người dùng và mặt hàng. Có thể chỉ định định dạng ma trận (coo hoặc csr) và trường giá trị.
inter_matrix: Tạo ma trận thưa mô tả tương tác giữa người dùng và mặt hàng. Ma trận có kích thước (số người dùng, số mặt hàng). Có thể chỉ định định dạng ma trận (coo hoặc csr) và trường giá trị.
_history_matrix: Tạo ma trận dày mô tả các bản ghi tương tác lịch sử của người dùng hoặc mặt hàng. Cung cấp ma trận lịch sử, giá trị lịch sử và độ dài lịch sử.
history_item_matrix: Tạo ma trận dày mô tả các bản ghi tương tác lịch sử của người dùng với mặt hàng.
history_user_matrix: Tạo ma trận dày mô tả các bản ghi tương tác lịch sử của mặt hàng với người dùng.
get_preload_weight: Lấy ma trận trọng số đã tải trước, với các hàng được sắp xếp theo ID của các token.
_dataframe_to_interaction: Chuyển đổi pandas.DataFrame thành đối tượng Interaction với các loại dữ liệu được xác định rõ ràng.