



Wrap - Up Report, Recsys 05

Team 우리 한번 믿오조

Team 소개

- | 이호준_T5168
- | 김동환_T5028
- | 박상우_T5081
- | 박예림_T5088
- | 임소영_T5172

목차

- 1-1. 프로젝트 개요
 - 프로젝트 주제
 - 데이터 개요
 - 활용 장비 및 재료
 - 프로젝트 구조 및 사용 데이터셋의 구조도
- 1-2. 프로젝트 팀 구성 및 역할
- 1-3. 프로젝트 수행 절차 및 방법
 - EDA
 - 모델 탐색
 - 모델 고도화
- 1-4. 프로젝트 수행 결과
- 1-5. 자체 평가 의견
 - 잘한 점
 - 시도했으나 잘되지 않았던 점
 - 아쉬운 점
 - 프로젝트를 통해 배운 점
- 2. 개인 회고
 - 2-1. 이호준_T5168
 - 2-2. 김동환_T5028

2-3. 박상우_T5081

2-4. 박예림_T5088

2-5. 임소영_T5172

1 - 1 . 프로젝트 개요

▼ 프로젝트 주제

| Book Rating prediction

유저의 책 평점 데이터 및 유저 메타 데이터, 책 메타 데이터를 바탕으로 어떤 책을 더 선호할지 예측하는 Task

▼ 데이터 개요

- 306,795 건의 평점 데이터 (train_rating.csv)
 - user_id, isbn, rating
- 149,570 개의 Book 데이터 (books.csv)
 - isbn, book_title, book_author, year_of_publication, publisher, img_url, language, category, summary, img_path
- 68,092 명의 User 데이터 (user.csv)
 - user_id, location, age

▼ 데이터 구조

▼ 환경 및 툴

개발환경 (AI Stage Server)

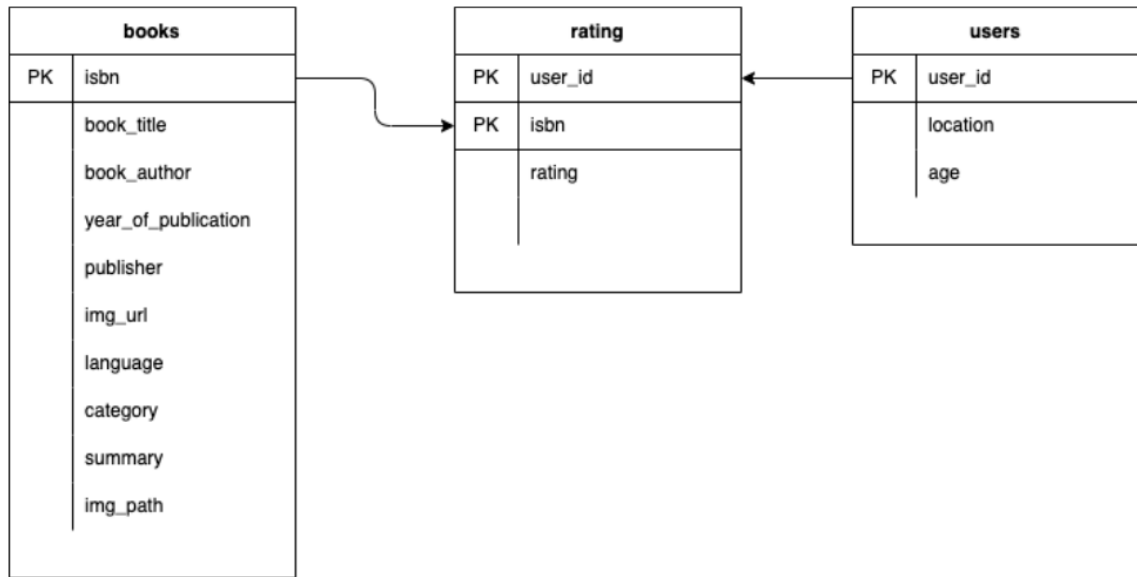
- OS : Ubuntu 18.04.5 LTS
- GPU : Tesla V100-SXM2-32GB

협업

- Github
- Slack
- Zoom
- 주로 오프라인 !

Tools

- Python
- Pytorch
- Optuna
- Scikit-learn



1 - 2 . 프로젝트 팀 구성 및 역할

- 이호준_T5168
 - 파이프라인 조율 및 앙상블
- 김동환_T5028
 - DL, Context 모델 총괄 및 tuning
- 박상우_T5081
 - Catboost 모델 고도화
 - 앙상블 및 미세 블렌딩
- 박예림_T5088
 - EDA 및 데이터 전처리
 - 모델 실험
- 임소영_T5172
 - EDA 및 데이터 전처리
 - 모델 실험

1 - 3 . 프로젝트 수행 절차 및 방법

▼ EDA & Data Preprocessing

전처리 방식

Users

- location
 - location 컬럼 분해 ⇒ location_country, location_state, location_city
 - location_city 컬럼에 대해 라벨 인코딩 진행 후, KNNImputer 라이브러리를 사용해 결측치 대체. k=5로 설정하여 결측치를 포함하는 샘플 주변의 5개 샘플들의 평균으로 결측치를 모두 대체함.
 - location_city 값에 따라 해당 city에서 자주 등장하는 state와 country로 location_state, location_country 결측치 대체. 하지만 city 정보만 존재하고, state와 country가 존재하지 않는 경우, 주어진 데이터로는 채우기 힘들어 'unknown'으로 처리함. (약 400명의 user들)
- age 결측치
 - 처음엔 나라별 최빈 나이 값으로 결측치를 채웠었으나, 해당 나라의 유저들이 일괄적으로 같은 값으로 결측치가 대체되다 보니 특정 나이에만 분포가 몰리는 문제 발생
 - 따라서 location 컬럼과 함께, KNNImputer 라이브러리를 사용해 결측치 대체

Books

- 언어 결측치
 - 1차 : 출판사별 최다 빈도 언어
 - 2차 : ISBN 의 앞 두글자 추출 후 최다 빈도 언어
 - 3차 : 최다 빈도 언어
- 카테고리 결측치
 - ▼ 1차 : resnet 을 활용하여 표지 이미지에서 카테고리 추출 → 약 40,000개 보완
 - img_path 에 null 값은 없었지만, 막상 image 를 띄웠을 때 파일이 없었던 경우가 있었음 → 표지 이미지를 텐서로 찍어보고, length 가 1인 경우 null 값으로 처리해줌

- 표지O-카테고리O 관계를 training 데이터로 사용하여 resnet 을 학습시킨 뒤, 표지O-카테고리X 관계 predicting 하였음. 이를 통해 상위 카테고리를 채움

```
books.isna().sum()
✓ 0.0s Python
```

isbn	0
book_title	0
book_author	1
year_of_publication	0
publisher	0
img_url	0
language	0
category	68851
summary	67227
img_path	41802
category_high	25939
dtype:	int64

- ▼ 2차 : author 에서 카테고리 추출 → 약 15,000개 보완
→ 작가별로 자주 쓰는 카테고리가 있음을 확인

```
William Shakespeare
category_high
drama      117
others      25
fiction     13
english     11
poetry       6
humor        2
book         1
art          1
character    1
child        1
Name: count, dtype: int64
—
Stephen King
category_high
fiction     124
others      21
american     9
war          2
fantasy      2
animal       1
adventure    1
literature    1
art          1
man          1
child        1
dog          1
business     1
```

```
BA = list(books[books['category_high'].notna()][book_author'].value_counts().index)

for author in BA:
    if not (books[books['book_author'] == author]['category_high'].value_counts().empty):
        books.loc[(books['img_path'].isna()) & (books['category_high'].isna()) & (books['book_author'] == author), 'category_high'] = books[book

✓ 35m 26.1s Python
```

```
books.isna().sum()
✓ 0.0s Python
```

isbn	0
book_title	0
book_author	1
year_of_publication	0
publisher	0
img_url	0
language	0
category	68851
summary	67227
img_path	41802
category_high	10833
dtype:	int64

포맷터 autopep8(7) 설치되어 있지 않습니다. 설치하시겠습니까?

▼ 3차 : publisher 에서 카테고리 추출 → 약 10,000개 보완

```
pub = books['publisher'].unique()

for p in pub:
    if not (books[(books['publisher'] == p) & (books['category_high'].notna())]['category_high'].value_counts().empty):
        books.loc[(books['publisher'] == p) & (books['category_high'].isna()), 'category_high'] = books[(books['publisher'] == p) & (books['category_high'].notna())]['category_high'].mode()[0]

books.isna().sum()

isbn          0
book_title    0
book_author    1
year_of_publication  0
publisher      0
img_url        0
language       0
category      68851
summary       67227
img_path      41802
category_high  424
dtype: int64
```

▼ 4차 : 나머지 값은 모두 'others' 로 치환해 보완

```
books.loc[books['category_high'].isna(), 'category_high'] = 'others'

books['category_high'].value_counts()

category_high
fiction      106209
others       22016
science      2439
art          1590
business     1528
...
imaginary      8
gay            7
geschichte     5
mystery        3
detective       1
Name: count, Length: 82, dtype: int64

books.isna().sum()

isbn          0
book_title    0
book_author    1
year_of_publication  0
publisher      0
img_url        0
language       0
category      68851
summary       67227
img_path      41802
category_high    0
```

Column 추가

Ratings

- users - `rating_count` : 각 user의 Rating 횟수 column

- books - `book_count_na` : sample마다(book마다) 결측치 개수를 센 컬럼
- books - `book_rating_count` : rating 된 아이템(book)의 개수를 값으로 가지는 컬럼
- books - `book_rating_mean` : 책별 평균 rating을 값으로 가지는 컬럼

▼ 모델 탐색

▼ FM

- 성능이 무난했던 모델. 베이스라인 기준 rmse 2.4 튜닝 시에 2.2 달성
- 다만 FFM과 거의 동일한 메커니즘 때문에 추후 고도화는 진행하지 않았음

▼ FFM

- 성능이 무난했던 모델. 베이스라인 기준 rmse 2.4 튜닝 시에 2.2 달성
- 상대적으로 오버피팅이 덜하고 Latent vector을 특성 추출로 활용할 수 있는 여지가 있어 오버피팅이 심한 DCN 모델과 결합하여 사용
- FFDN 모델 기준으로 튜닝 시 rmse 2.17까지 달성

▼ DeepCoNN

- 텍스트를 이용한 모델로 성능이 그리 좋지는 않았다. 예측 결과값도 상대적으로 다른 모델에 비해서 튀는 값도 많았고 텍스트에 결측치가 많아 상대적으로 성능이 좋지 못했던 모델
- 다만 텍스트를 활용한다는 측면이 다른 모델이 캐치하지 못하는 점을 학습할 수 있어서 FFDN과 CNN_FM과 섞어서 사용을 해보았으나 RMSE 2.18 정도로 큰 향상을 없었음

▼ CNN_FM

- 이미지를 이용한 모델로 이 역시 그리 성능이 좋지는 않았다. 다만 이 과정에서 추출한 Latent 벡터를 활용할 수 있는 여지가 있었던 모델
- 다만 텍스트를 활용한다는 측면이 다른 모델이 캐치하지 못하는 점을 학습할 수 있어서 FFDN과 결합해 사용을 해보았으나 rmse 2.17 정도로 큰 향상은 없었음

▼ FFDN

- DCN 모델을 기반으로 발전한 모델.
- DCN은 베이스 라인 코드 기준으로 가장 성능이 좋았음(RMSE 2.2)

- 다만 오버피팅이 심해 이를 보완하기 위해서 FFM과 결합해서 튜닝 후 2.15까지 달성
- 추가로 Stack 구조와 Parallel 구조 모두 시도를 해보았지만 거의 비슷했음
- 가장 성능이 좋아 이 모델과 Cat Boost를 앙상블 및 K-fold 후 2.109 달성

▼ DeepFM

- DeepFM 모델이 없어서 추가해 실험
- DCN과 더불어서 가장 좋은 성능. 튜닝 후 rmse 기준 2.17
- 다만 DCN과 비슷하지만 조금 떨어지는 성능으로 추후 이용은 하지 않았음

▼ Catboost

- User와 Books 메타 데이터를 merge하고 na를 -1를 채운 것 만으로도 rmse 2.1563 달성
- 베이스라인 모델 중 가장 높은 성능이므로 쪽 고도화하기로 결정
- EDA 팀이 데이터 Update 할 때마다 모델에 학습 시켜 rmse 2.1320까지 성능 고도화
- 여러 Feature를 생성하여 성능 향상을 노렸지만, data leakage 때문인지 user의 rating count를 제외한 다른 생성 feature 들은 전형적인 overfitting을 유도함
- 이후 하이퍼파라미터 최적화 및 10 fold로 generalization performance를 증가시켜 rmse 2.1214 달성

▼ 모델 고도화

▼ 튜닝

- 기본적으로 Hyperparameter 최적화 라이브러리인 Optuna 사용
- tune.py 파일로 파이프라인에 공유되게 함





▼ 교차 검증

- Generalization performance 극대화를 위해 kfold.py 파일을 생성
- Regression Task이지만, 평점의 분포가 고르지 않기에 stratified K-fold를 사용해 데이터 불균형을 해소하려 노력함

▼ 앙상블

- 복수 모델간 앙상블을 진행할 때, 교차검증을 활용하여 train에 대한 데이터를 예측한 결과와 선형회귀를 이용하여 각 모델의 앙상블 계수를 측정함
- 단순 앙상블 계수를 사용하는 것 외에, 유저별 클러스터링을 통하여 개인별 앙상블 계수를 측정함
- 이외에도 임의로 weight를 지정해 여러 모델을 앙상블

1 - 4 . 프로젝트 수행 결과

4 (1 ▾)	RecSys_05조	   	2.1094	44	4d
------------	------------	---	--------	----	----

최종 모델 : CatBoost(10fold), FFDCN(10fold) ensembled model

총 14 팀 중 rmse 2.1094 로 4등을 기록하였음.

Public 기준 3등(RMSE 2.1126) 이었는데 Private 데이터에서는 4등을 해 조금 아쉬웠음. 이후 1, 2팀의 발표를 들어보니 우리 팀과 접근 방법 또한 유사하고, 오히려 시도해 본 아이디어는 우리가 더 많은 것 같아서 아쉬운 하지만 결과에 승복하기로 함. 😊

1 - 5 . 자체 평가 의견

▼ 잘한 점

- 아이디어가 생기면 고민하지 않고 일단 실험한 점
- 팀원 간 의견을 활발히 공유하고 역할 분담을 확실히 한 점
- 다양한 아이디어를 생각해보고 공유한 점
- 다른 사람들도 내가 사용한 코드를 쉽게 사용할 수 있게 OOP oriented code를 작성한 점
- 여력이 남는 팀원이 적극적으로 추가 할 일을 물어보고 수행하면서

▼ 시도했으나 잘 되지 않았던 점

- 최종적으로 사용할 모델 및 데이터를 선정하였는데, 이후 드라마틱한 성능 향상은 실패하였음
- 유저 개인 별로 앙상블 weight를 다르게 설정을 해보려고 했으나 시간이 부족해서 실패하였음
- 마찬가지로 앙상블 weight를 단순한 MLP 레이어를 통해서 학습을 하려고 했으나 실패하였음
- context, text, img 모든 요소를 결합해 학습한 FFDCN + CNN_FM + DeepCoNN을 결합해보려고 했으나 큰 향상은 없었음

상황에 따라 유동적인 업무를 수행한 점

- 높은 성적을 받은 점
- 팀원의 말에 귀기울이고, 서로 격려하고 응원한 점

- 깃허브를 사용해 팀원 모두 적극적으로 코드를 공유하며 작업하려 했으나, 깃허브 사용에 미숙하고 시간을 많이 투자하지 못하는 상황이었어서 깃허브 업로드 과정에서 생기는 문제(gitignore 등)를 해결하지 못하고 간단한 공유만 하게 되었음

▼ 아쉬운 점

- 강의를 다 듣고 대회를 시작하느라 초반 결과 제출 횟수를 다 쓰지 못한 점
- 이론적인 지식, 조사에 기반하지 않고 단순히 직관적으로 아이디어를 떠올리고 생각해 성공하지 못한 점
- 시간이 부족해 좀 더 아이디어를 고도화하지 못한 점
- 학습 과정을 wandb나 TB같은 툴을 이용해 모니터링하지 못한 점
- 좀 더 다양한 전처리 기법을 사용해 다양한 데이터를 만들었다면 좋았을 것
- 앙상블 모델을 빨리 구현하지 못해 코드 실행시간이 부족했던 것
- 좀 더 체계적으로 다같이 모델 & 앙상블 실험을 했다면 성능이 더 개선되었을 것

▼ 프로젝트를 통해 배운 점

- 데드라인이 정해져 있는 프로젝트에서는 계획을 확실히 세우는 것이 중요한 점
- 팀 프로젝트에서 개개인의 뛰어난 실력보다 팀원 간의 의견소통이 훨씬 더 중요한 점
- 팀 프로젝트에서 깃을 활용하고 관리하는 방법
- 도메인의 특성을 파악하고 배운 개념 및 이론들을 실제 프로젝트에 적용하고 활용하는 방법
- 빠른 실행력의 중요성, 체계적인 워크스페이스 관리의 중요성을 깨달음

2. 개인 회고

▼ 2.1 이호준_T5168

맡은 역할 | 파이프라인 조율 및 앙상블

파이프라인 관리

1. Git

Git Repository를 중심으로 프로젝트 프로그램을 관리하는 역할을 맡았다. 초기 Git Repository형태를 생성하고, git push와 pull을 자동화한 python 파일을 작성했다.

Git을 관리할 때, 각자의 branch를 생성하는 것이 아닌 팀원 모두 main에서 진행하였다. 팀원들이 서로 의사소통을 자주 하기 때문에 Pull Request를 관리하는 팀장 역할이 필요없으며, 대회기간이 짧은 만큼 branch 충돌로 인한 문제를 미연에 방지하고자 해당 방식으로 Git을 사용했다.

새로운 Commit이 추가되면 Git이 잘 작동하는지 체크하고, 오류가 생기거나 파일이 손상되는 경우 수정하여 Git이 잘 작동할 수 있도록 관리하였다.

2. Preprocessing the data to match the model

EDA를 맡은 팀원이 새로운 버전의 데이터를 공유하면 Context, DL 등의 모델이 해당 데이터를 읽고 모델 변수를 생성할 수 있도록 프로그램을 정리하는 역할을 맡았다. 특히 이 부분이 어려웠는데, 오류를 찾기 위해 모델이 생성되고 데이터를 학습하고 예측하는 전 과정을 모두 실행하면서 어떤 부분에서 오류가 발생했는지 탐색해야 했기 때문이다.

앙상블

1. 선형 회귀를 활용한 기본 앙상블

EDA를 맡은 팀원들이 꾸준히 버전업을 하고, 모델을 맡은 팀원이 성능 좋은 모델을 탐색하여 낮은 RMSE를 보였다. 앙상블을 통해 성능이 좋은 모델들을 결합하여 더 좋은 성능을 낼 수 있으므로, Weight를 주는 최적의 방식을 공부했다. 성능이 좋기로 유명한 AutoML들을 구현한 서드파티에서 어떠한 알고리즘을 사용하였는지 탐색했다. 그 중 구현이 간단한 선형회귀를 통해 Weight를 주는 아이디어를 구현하여 좋은 성능을 보였으며, 이를 통해 팀 제출 중 2등의 결과물을 생성했다.

2. 유저별 클러스터링 & 유저별 Weight

모델 자체에 Weight를 줌으로써 RMSE를 낮출 수 있었으나 드라마틱한 성능 변화를 보이지 않았다. 그 이유는 모든 유저가 같은 특성을 갖지 않기 때문이다.(Heavy User 등) 따라서 유저별 클러스터링을 통해 각 유저 그룹마다 Weight를 측정하여 앙상블을 적용시킨 코드를 구현하여 드라마틱하게 성능을 향상시키는 것을 확인했다.

총평 및 아쉬웠던 점

리더보드 3위(RMSE 2.1126), 최종 4위(2.1094)로 높은 성적을 거두었다. 높은 성적을 거둔 것은 각자의 역할을 충실히 수행한 팀원들 덕분이다. 다만 각자의 역할이 서로 상이하여, 서로 의견을 나누고 아이디어를 공유하는 것이 제한되는 단점이 있었다. 모두 공통된 역할을 수행하는 것보다 전문적으로 분업하는 것이 좋다고 생각하나, 서로 소통할 수 있도록 개인이 주 역할, 부 역할로 2가지 이상의 역할을 수행하는 방법을 시도해보면 좋을 것이라 생각한다.

유저별 클러스터링 및 개인별 추천 구현이 늦어져 최종 프로젝트에 해당 분석결과를 적용시키지 못했다. 성능 자체는 드라마틱하게 향상되었으나, 오버피팅인지의 여부를 파악하는 방법 또한 구현하지 못했다. 제한된 시간 안에서 최대의 성능을 내는 연습이 상당히 필요하다고 느꼈다.

▼ 2.2 김동환_T5028

맡은 역할 | 모델링 및 튜닝

모델 조사 및 실험

먼저 베이스 라인 코드에 있는 모델들을 조사해보고 돌려보면서 어떤 특징이 있나 조사를 해보았다. 일단 대부분의 데이터가 범주형이고 유저와 아이템 특징이 있어서 둘의 상호작용을 포착할 수 있는 Context Aware 모델이 좋을 것이라고 생각을 해 DeepFM과 DCN을 고려했다. 그리고 일단 기본 베이스라인으로 돌려보았을 때 대부분의 모델이 1~2 Epoch 기준으로 오버피팅이 되는 것을 보고 Tree 기반의 모델이 조금 더 나을 것 같아서 Cat Boost를 모델 후보군에 추가를 했다.

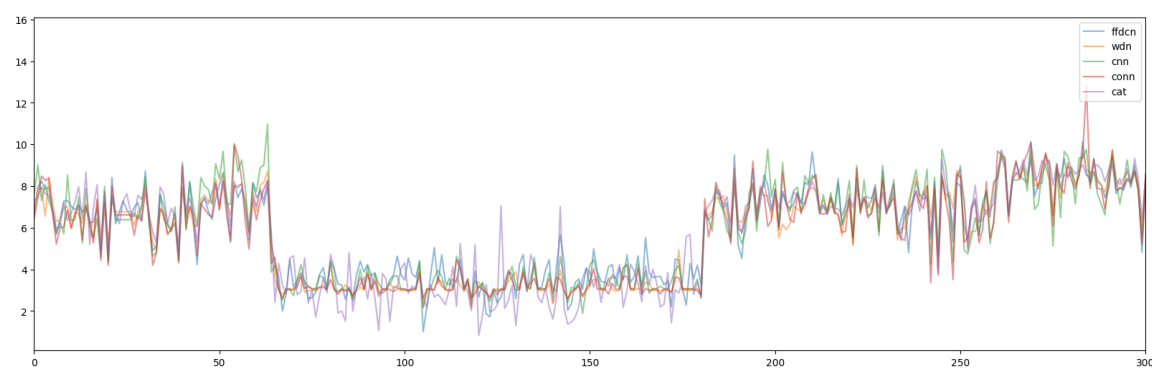
모델 추가 및 모니터링

일단 기본 모델 외에 다른 모델들을 main.py에 돌아갈 수 있도록 추가를 했다. DeepFM과 DCN Parallel를 추가를 했는데 Parallel은 Stack구조보다 Parallel이 좀 더 이 데이터에 적합할 것 같아서 추가를 해보았다. 이후에 실험을 했을 때 딥러닝 모델 중에서는 DCN, DCN Parallel, DeepFM이 가장 좋은 편이었다. 다만 앞서 언급한대로 딥러닝 기반 모델은 오버피팅되는 경향이 있어 FFM 모델을 합친 FFDCN과 FFDCN Parallel 모델을 또 추가했다. 그 결과 일반 DCN은 2.18대인 반면 FFDCN Parallel과 Stack 모델이 튜닝 후에 약 2.15~2.16 정도로 성능이 가장 좋았다. 또한 추가한 모델들이 다른 실행 파일들에 돌아갈 수 있도록 pipeline을 수정했다.

Null 값이 많아서 이런 데이터는 overfitting된 모델이 잘 예측을 못하나 싶어서 Feature로 Null Count를 한 번 추가를 하거나 평점이 유독 이상한 유저가 있으면 유저의 평균 평점을 추가해보았지만 크게 변화가 없거나 오히려 떨어졌다.

모델의 예측 결과값을 그래프로 찍어보았을 때 DeepFM과 DCN 기반 모델들의 결과는 대부분 비슷하였으나 CNN FM이나 DeepCoNN은 조금 다른 양상이 보여서 이를 섞은 FFM + DCN + CNN FM 과 FFM + DCN + CNN_FM + DeepCoNN인 하이브리드 모델을 추가해보았다. 다만 결과는 2.18~2.17 대로 그렇게 좋지는 않았고 앙상블을 해도 향상이 있지는 않았다.

이후 몇 가지 더 시도를 해보았으나 모델을 아무리 튜닝을 하고 수정을 해도 Cat Boost의 성능을 뛰어넘지는 못했다. 물론 K-Fold를 이용하면 성능이 더 좋아졌지만 이는 Cat Boost도 마찬가지였고 이 과정에서 좀 한계를 맞이했던 것 같다. 시간이 부족해서 하지 못했지만 오히려 좀 더 단순한 모델인 SVM이나 MF, CB, CF를 이용해서 앙상블을 하거나 Hybrid로 결합을 한 것이 좋을 수도 있다고 생각을 했고 ML 기반의 DT나 Clustering같은 것도 시도를 하지 못한 것이 아쉽다.



튜닝

기본 모델 구조는 우리 데이터 셋에는 학습이 잘 이루어지지 않았다. 또한 FFDCN 같은 경우도 임의로 만들었기 때문에 적합한 모델 구조를 알 수가 없었다. 따라서 다양한 실험을 통해서 적합한 모델 구조를 찾을 필요가 있어서 Optuma를 이용한 실험을 할 수 있도록 설계했다. 따라서 main.py의 구조를 본 떠서 다른 팀원들도 쉽게 튜닝을 할 수 있도록 tune.py을 만들었다. 실행 시에 json 파일로 tuning option을 지정해서 learning rate, weight decay 이외에도 모델 구조도 튜닝을 할 수 있도록 했다. 그 결과 2.2~2.3 정도에서 대부분 2.18, 좋은 것은 2.15까지 떨어졌다. 다만 Bayesian이나 Grid Search도 시도해보면 좋을 듯. 또한 실험 인자가 많았기 때문에 좀 더 진행을 했으면 더 떨어졌을 수도. 좋은 튜닝과 그렇지 않은 튜닝 모델의 결과의 차이가 컸기 때문.

앙상블

일단 Tabnet 논문에 딥러닝 기반 모델과 Tree 기반 모델을 결합하면 성능이 좋아진다고 하여 단순하게 각각 10 K-Fold를 한 Catboost와 FFDCN을 1:1로 앙상블을 했다. 그 결과 2.11 대로 가장 좋은 성능이 나왔다. 그리고 유저 별로 예측을 잘 하는 모델이 다를 것이다 생각을 해 유저 개인 별로 앙상블 계수를 다르게 하는 모델을 구현해보면 어떨까 생

각을 했다. Pretrained된 모델들의 예측값을 뽑아놓고 $N_users \times N_models$ 크기의 임베딩을 만들어서 이 임베딩을 이용해 각 유저 별 앙상블 계수를 계산을 할 수 있지 않을까 싶어서 구현 후 실험을 해보았지만 Loss가 4대로 줄지 않았다. 아마 이는 내가 유저와 다른 상호작용을 고려하지 않아서 그런 것 같다. 유저가 평점을 주는 대에는 유저와 책, 유저와 기타 요인이 있을텐데 이를 무시하고 단순히 모델과 유저만 고려하다보니 이렇게 나온 것 같다. 또한 단순한 MLP 레이어를 실험을 해보았지만 2.4 정도에서 줄지 않았다. 이도 앞서 상기한 이유 때문이라고 생각을 한다. 다음 번에는 조금 더 이 부분을 발전시켜보고 싶다.

총평 및 아쉬운 점

먼저 리더보드 3위, 최종 데이터 공개 후 4위로 12개 팀 중 상대적으로 높은 순위를 달성했다. 그리고 1,2등의 발표를 들었을 때 우리가 했던 것과 동일해서 아쉬웠다. 그리고 우리가 시도한 점이 좀 더 많았는데 아마 좀 더 제대로 했다면 더 좋지 않았을까 하는 아쉬움이 있었다. 모델링이나 여러 시도를 해보는 과정에서 정확한 이론이나 수식에 기반하지 않은 직관적인 감으로, 빠른 속도로 이런 거 저런 거를 시도해보았어서 이 부분이 좀 부족했던 것 같다. 다만 시스템 아키텍처 위에서 작동하도록 여러 코드를 작성한 경험 덕분에 OOP oriented 코드를 구현하는 실력은 좀 늘어서 좋았던 점 중에 하나다. 나만이 아니라 다른 사람들도 간단하게 동일하게 코드를 실행할 수 있어야한다는 것이 중요하다는 것을 느꼈다. 결과적으로 아쉬움이 조금 있지만 좋은 결과를 얻었고 이렇게 집중해서, 이런 환경에서 팀과 같이 집중한 경험은 처음이라 좋았던 것 같다.

▼ 2.3 박상우_T5081

말은 역할 | Catboost 모델 고도화 및 앙상블

Why Catboost?

기존 BaseLine 코드를 사용한 모델 중, 눈에 띄는 성능이 나오는 모델은 없었다. 따라서 여러 모델을 시험해 보던 중, 단순히 User와 Book을 Merge하고 결측치를 -1로 채워준 baseline catboost 모델의 rmse가 약 2.15로 다른 모델에 비해 월등히 높은 성능을 보여주었다. 이는 아마도 전체 Column 중, year와 age를 제외한 모든 변수가 범주형 이고 일종의 정형 데이터의 형식을 띄고 있어 이러한 데이터 형식에 좋은 성능을 보장하는 Gradient Boosting Machine이 잘 작동하는 것으로 보였다. 따라서 Catboost를 전담해 Hyperparameter 튜닝 및 여러가지 feature를 generation 해 성능을 끌어올리는 역할을 맡았다.

Catboost 모델 고도화

기존 스크립트 형식의 파일은 Pytorch의 딥 러닝 모델에 적합하게 코드가 짜여져 있어, Catboost를 파이프라인에 연결하는 작업과 동시에 주피터 노트북으로 모델 고도화를 진행하였다. model 디렉토리에 Catboost model.py 파일을 생성하였고, data 디렉토리에 catboost dataloader.py 파일을 생성하여 기존 파이프라인에 병합시켜 주었다. 이후 다른 팀원이 작업한 tune.py 파일에서도 실행할 수 있도록 병합시켜 주었다.

이와 별개로, EDA 팀에서 데이터를 전처리하여 보내주는 버전 별로 Catboost 모델을 학습하였다. version 4 데이터에서 가장 좋은 성능을 보여주었으며, 이후 업데이트 되는 버전에서는 rmse 값이 큰 차이가 없었다. 각 버전 별로 모델을 학습할 때 마다 Optuna를 통해 각각 다른 Hyperparameter 최적화를 통해 제일 좋은 성능을 도출하려 노력하였다.

여러가지 feature generation 시도 중, 의미가 있었던 것은 user가 평가를 한 횟수인 user's rating count 뿐 이었다. 다른 생성된 feature는 data leakage 때문인지 training rmse는 낮지만 valid rmse가 매우 높은 전형적인 overfitting 현상을 만들어 냈다. 따라서 최종적으로 사용한 catboost 모델에서도 만들어낸 feature 중 user's rating count만 사용하였다.

양상블

팀원들이 수학적 접근으로 최적의 양상블 모형을 만들어내는 것과 별개로, 당장 높은 결과를 도출 해야 함, 제출 횟수가 하루에 10번으로 정해져 있어 소모해야 함 등의 이유로 임의로 가중 평균 낸 양상블 모델을 매일 블렌딩 하였다. 모델이 잘 설명해주는 부분이 다르다는 가정을 전제로, Catboost와 FFM, DCN, FFDCN 등의 상이한 모델과 블렌딩하였는데, 역시나 좋은 성능을 기대할 수 있었다. 모델 파트의 팀원이 결과를 도출하는 즉시 Catboost의 결과와 양상블 하였고, 이를 제출 해 초반부터 리더보드의 상위권에 위치할 수 있었다. 마지막 최종 제출시에는 kfold.py 파일을 생성해 generalization performance를 극대화 하였다. regression task이지만, 점수 분포가 천차 만별이기에 Stratified Kfold를 사용해 데이터 불균형을 해소하려 노력하였다.

총평

평소 여러 경진대회에 참가하였고 좋은 성적을 낸 경우도 있었지만 본 대회만큼 몰입하여 임한 대회는 없었던 것 같다. 팀원과 여러 아이디어를 제시하여 실험해보고 또 좋은 결과를 얻어가면서, 문제 해결력이 상승함과 동시에 AI 엔지니어로서의 즐거움도 살짝 맛볼 수 있었다. 14팀 중 4등으로 좋은 성적을 얻었고, 파이프라인을 제작하면서 OOP 실력도 상승하였지만, 그 무엇보다 의미있는 것은 level 1 팀원과 마지막으로 즐거운 시간을 보낸 것이라 생각한다. 일종의 게임을 하는 듯한 느낌으로, 팀원들의 생각은 모르겠지만 나는.. 즐거웠다. 대회에서 점수를 미세하게 향상 시킬 수 있는 여러 팁을 얻은 것도 큰 자산이라고 생각한다.

▼ 2.4 박예림_T5088

말은 역할 | EDA & 데이터 전처리 및 모델 실험

EDA & 데이터 전처리

데이터를 자세히 뜯어보고 각 컬럼별로 분포를 살펴보고 데이터의 특징을 하나하나 얻어나갔다. 분포가 특정 값에 몰려있는 경우가 많아 이를 고려하여 전처리를 하기 위해 노력했다. 또 팀원들의 의견을 수용하여 여러가지 파생 변수들을 만들기도 했다.

특히 전처리에 있어 그룹별 통계량을 사용하거나 KNNImputer를 사용해보며 다양한 방법으로 결측치를 채우는 데 집중하였고, 데이터를 만지고 버전 업을 할 때마다 모델 성능이 조금씩 향상이 되는 것을 확인함으로써 성취감을 느끼며 작업할 수 있었다.

그리고 location 결측치를 채울 때, city, state, country 중 일부 한 정보만 있으면서 그 샘플이 유일한 경우 다른 샘플들을 참고해서 채울 수 없어 막막했던 때가 가장 기억에 남는다. 동시에 배운 것도 가장 많았던 지점이기도 하다. 그러한 경우에는 외부 데이터도 참고할 수 없기 때문에 의미 있게 잘 채울 수 있는 방법이 존재하지 않는 것으로 판단하고 'unknown' 값으로 일괄적으로 대체해주었다.

모델 실험

모델 파트를 맡으신 분들이 주도적으로 이끌어주시는 방향에 따라 각 모델을 실험적으로 돌려보고 여러가지 앙상블을 시도하며 모델 작업을 도왔다. 프로젝트 막바지에 하이퍼파라미터 튜닝 결과로 나온 베스트 파라미터 조합을 받아, 최종 버전의 데이터로 FFDCN 모델을 돌려 성능이 많이 개선되었을 때, 그리고 그 모델을 CatBoost와 조합해 최고 성능이 나왔을 때 조금이나마 모델 작업에 기여를 한 것 같아 뿌듯함을 느꼈다.

아쉬웠던 점

데이터를 만지는 일을 맡아 며칠 동안 EDA & 전처리에 집중해서 하고, 데이터를 완성한 후에는 모델 작업을 도왔는데, 그 시작이 어렵고 막막했던 게 기억이 난다.

어려움에 부딪힐 때마다 질문을 거듭해 모델 작업 환경을 구축해나가고, 모델을 돌려보는 것까지 한 것만으로도 만족스럽긴 하지만, 사용한 모델에 대한 디테일한 공부와 이해를 하고 돌린 것이 아니라 아쉬움이 많이 남는다.

또 막바지에 체계적으로 다같이 모델 실험에 집중해서 더 많은 시도를 했다면 성능을 좀 더 개선할 수 있었을 것 같다.

다음 프로젝트에서 시도해볼 것

좀 더 체계적으로 워크스페이스를 구축하고 관리해보고 싶다.

특히 프로젝트 초반에 시간을 투자해서라도 깃허브 업로드가 잘 되도록 세팅하여, 모두 적극적으로 깃허브를 사용할 수 있으면 한다.

또한 모델에 대한 깊이 있는 이해와 공부를 한 후에 모델을 사용함으로써 효과적으로 모

델 성능을 개선하고, 하이퍼파라미터 튜닝도 직접 시도해보고, 적극적으로 앙상블도 돌려보며 모델 파트에 집중해보고 싶다. 다음엔 데이터 뿐만 아니라 모델도 많이 만져보면서 모델에 대한 감을 많이 익히는 것이 목표이다.

▼ 2.5 임소영_T5172

👁️ 말은 역할 | EDA & 데이터 전처리 및 모델 실험

EDA

데이터를 분석하여, 결측값을 채울 수 있는 방법과 내재되어 있는 특이한 관계성을 파악하려고 노력하였다. users 데이터의 경우 age 결측값이 많아 이를 채울 수 있는 방법에 대해 팀원과 함께 논의해보았다.

books 데이터의 여러 column 중 rating 에 가장 중요하게 작용할 것 같았던 category의 결측값과 언어 결측값을 채우기 위해서 여러 방법을 고안해내었다.

데이터 전처리 : books.csv 카테고리 결측치 보완 한계점

v5 data : resnet 을 학습시켜 총 80개의 상위 카테고리로 예측해 결측치를 채웠다. training 데이터 내 분포가 fiction 에 거의 치우쳐있었던 만큼, 원활하게 학습시킬 수 없었던 점이 아쉬웠다.

v7 data : fiction 을 under sampling 하여 resnet 을 학습시키고, 좀 더 큰 범주인 총 22개의 상위 카테고리로 예측해 결측치를 채웠다. v5 data 보다는 fiction 으로 예측하는 정도가 줄어들긴 했으나 여전히 fiction 데이터가 상당하여 학습이 잘 되지 않았던 것 같다. 미세한 성능 향상은 있었지만, 기대했던 만큼 큰 보폭으로 성능이 향상되지 않았기에 아쉬움이 컸다.

총평 & 아쉬웠던 점

좋은 팀원분들을 만나 대회를 성황리에 끝낼 수 있었다. 대회가 처음인만큼 두려움도 컸고, 나만 모르는 것일까 조금씩 꽤나 스트레스를 받았던 적이 있다. 하지만 팀원들의 따뜻한 격려와 응원 덕분에 모든 것을 이겨내고 결국 해야할 일을 끝낼 수 있었다. 나의 성장에 큰 도움을 준 우리 팀원들에게 다시 한 번 감사하다.

EDA 와 결측치를 채우는 일에 집중하다보니, 모델을 실질적으로 만져보고 개선할 수 있는 기회가 없었다. 데이터 결측치가 채워질수록 모델의 성능이 미세하게 좋아졌기에, 결측치를 완벽히 채우는 task 에만 집중했던 것이 아쉬웠다. 다음 대회가 되어서야 모델을 사용할 수 있을 것 같은데 잘 해낼 수 있을지 걱정된다. 모델 파트를 맡았던 팀원들에게 방법을 디테일하게 물어본 뒤 성장한 모습으로 Level 2 를 맞이하고 싶다.

다음 프로젝트에서 시도해볼 것

모든 팀원들이 각각 다른 역할을 맡아 수행했기에, 서로 모르는 점을 공유할 수 없거나 도움을 줄 수 없었다. 다음 프로젝트에서는 EDA 를 다함께 진행해보고, 각자 다른 시각으로 데이터를 본 결과를 합산해 좀 더 나은 EDA 결과물을 만들어보고 싶다. 이후, 모델링을 각자 또 함께 진행하여 서로의 모델을 보완해줄 수 있었으면 좋겠다!

이번 대회를 진행하며, 겁내지 않고 새로운 것들을 시도해본 경험은 잊을 수 없을 것 같다. 다음 대회에서도 역시 떨지 않고 이것저것 시도해보며 팀에게 도움을 줄 수 있도록 노력해야겠다! 😊