[RecSys] Book Rating Prediction 회고_개

작성자 : 추천시스템 8조

이나현_T4146　전해리_T4191

정의준_T4200　조원준_T4211

채민수_T4217

[RecSys] Book Rating Prediction

사용자의 책 평점 데이터를 바탕으로 사용자가 어떤 책을 더 선호할지 예측하는 태스크입니다.

#부스트캠프4기 #추천시스템 #비공개대회

▼ [프로젝트 개요]

책과 관련된 정보와 소비자의 정보, 그리고 소비자가 실제로 부여한 평점, 총 3가지의 데이터 셋(users.csv, books.csv, train_ratings.csv)을 활용하여 각 사용자가 주어진 책에 대해 얼마나 평점을 부여할지에 대해 예측

▼ Input / Output

Input

- `train_ratings.csv` : 각 사용자가 책에 대해 평점을 매긴 내역
- `users.csv` : 사용자에 대한 정보
- `books.csv` : 책에 대한 정보
- `Image/` : 책 이미지

Output

- `test_ratings.csv` 의 사용자가 주어진 책에 대해 매길 것이라고 예상하는 평점

▼ 학습 데이터, 평가데이터

학습 데이터

- `books.csv` : 149,570개의 책(item)에 대한 정보를 담고 있는 메타데이터입니다.
- `users.csv` : 68,092명의 고객(user)에 대한 정보를 담고 있는 메타데이터 입니다.
- `train_ratings.csv` : 59,803명의 사용자(user)가 129,777개의 책(item)에 대해 남긴 306,795건의 평점(rating) 데이터 입니다.

평가 데이터

- `test_ratings.csv` : 26,167명의 사용자(user)가 52,000개의 책(item)에 대해 남길 것으로 기대하는 76,699건의 평점(rating)을 예측합니다.

▼ [프로젝트 팀 구성 및 역할]

- **조원준(팀장)** : 대회 목표 설정 및 협업 시스템 제안, 데이터 기반 가설 검증 및 모델 하이퍼 파라미터 튜닝.
- **채민수(부팀장)** : 데이터 EDA, 머신 러닝 모델 피쳐 튜닝, 모델 최종 앙상블
- **전해리(일정관리)** : 대회 일정 관리, 컨텐츠 기반 모델 설계 및 실험, NCF 모델 실험 및 튜닝
- **이나현(실험기록)** : 데이터 전처리 및 DL계열 모델에 context 계열 모델의 feature 적용
- **정의준(실험기록)** : 모델학습 지원 기능 개발, WDN 모델 실험 및 튜닝

▼ **[프로젝트 수행 절차 및 방법]**

Sun	Mon	Tue	Wed	The	Fri	Sat
24	25	26	26	27	28	29
			베이스라인 파악			데이터 EDA
30	31	1	2	3	4	5
데이터 EDA		모델 실험 및 모델 튜닝		최종 제출	Wrap up	
가설 설계 및 가설 검증			모델 앙상블			

- **대회 1주차**

  베이스라인 코드를 함께 리뷰하고 데이터 EDA를 진행하였습니다.

- **대회 2주차**

  1주차에 이어서 데이터 EDA를 진행하면서 가설을 세우고 이를 검증하였습니다.

  데이터 파트와 모델 파트로 역할을 분담한 후 각각 실험을 진행하였습니다.

  최종 모델의 파라미터를 적용하여 모델을 앙상블 하였습니다.

▼ **협업 툴**

git : 코드 공유 및 형상관리 등 유연한 개발을 위해 사용한 툴

slack : 팀원 간 소통, 슬랙봇을 이용해 실험 결과 공유

notion : 프로젝트 전체적인 관리 및 기록

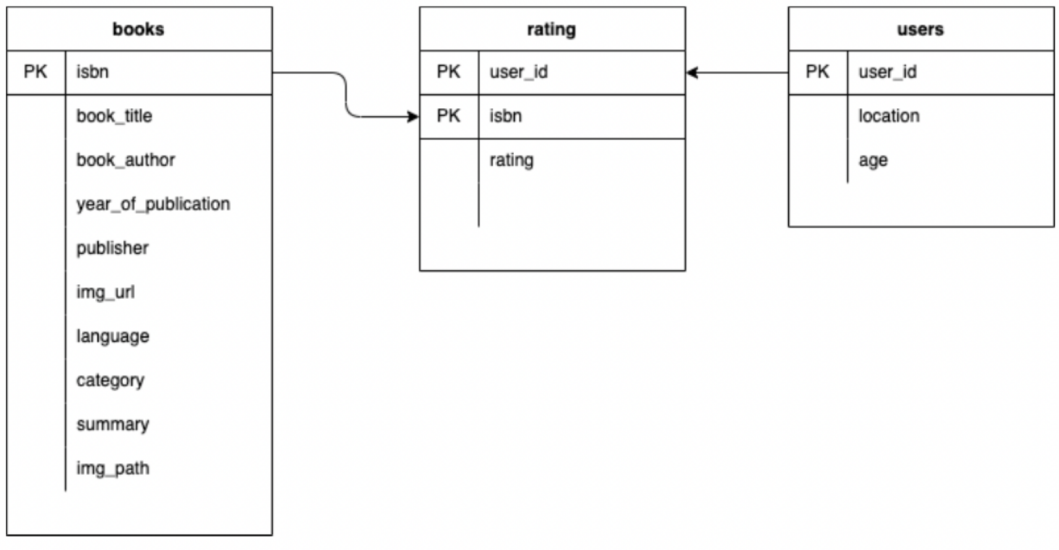
trello : 팀원 간 현 진행 상황 공유

google sheets : 모델 파라미터 실험 결과 기록 및 공유

zoom : 원활한 소통을 위한 화상회의 툴

▼ **[프로젝트 수행 결과]**

- **탐색적 분석 및 전처리**
  - 학습 데이터 소개
    - 개요



- 학습 데이터

  users.csv에는 68,092명의 고객 정보가 담겨있습니다.

  books.csv에는 149,580개의 책 정보가 담겨있습니다.

train_ratings.csv에는 306,795건의 평가 정보가 담겨있습니다.

users.csv

user_id	location	age
8	timmins, ontario, canada	NaN
11400	ottawa, ontario, canada	49.0
11676	n/a, n/a, n/a	NaN
67544	toronto, ontario, canada	30.0
85526	victoria, british columbia, canada	36.0

train_ratings.csv

user_id	isbn	rating
8	0002005018	4
67544	0002005018	7
123629	0002005018	8
200273	0002005018	8
210926	0002005018	9

books.csv

isbn	book_title	book_author	year_of_publication	publisher	img_url	language	category	summary	img_path
0002005018	Clara Callan	Richard Bruce Wright	2001.0	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	en	['Actresses']	In a small town in Canada, Clara Callan reluct...	images/0002005018.01.THUMBZZZ.jpg
0060973129	Decision in Normandy	Carlo D'Este	1991.0	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	en	['1940-1949']	Here, for the first time in paperback, is an o...	images/0060973129.01.THUMBZZZ.jpg
0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999.0	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	en	['Medical']	Describes the great flu epidemic of 1918, an o...	images/0374157065.01.THUMBZZZ.jpg

- 테스트 데이터

  테스트 데이터는 학습 데이터의 train_ratings.csv와 동일한 column을 가지고 있고, rating은 0으로 채워져있습니다.

  테스트 데이터는 76,699건의 예측해야하는 user_id와 isbn조합이 담겨있습니다.

- 데이터 전처리
  - 전체

    (indexing) 모든 rating을 제외한 모든 column값을 인덱싱 처리해주었습니다.

  - users

    (fill in) age의 결측값에 대해선 평균을 취해주었습니다.

    (split & fill in) 쉼표로만 구분되었던 location정보를 country, state, city로 나눈 후 city를 기준으로 country와 state의 결측값을 채워주었습니다.

  - books

    (refine) : 동일한 publisher의 다른 이름을 동일하게 해주었습니다.

    (bounding) : 카테고리들을 상위 카테고리로 묶어주었습니다.

- 데이터 분석
  - sparsity

    최대가능한 평점 기록 개수 : 유저수 * 책수 = 68069 * 149570 = 10181080330

    주어진 평점 기록 개수 : 306,795

    sparsity : 1 - 주어진 평점 기록 개수 / 최대 가능한 평점 기록 개수 = 99.996%

  - train 데이터와 test 데이터의 분포

    Adversarial Validation의 방법을 통해 train 데이터와 test 데이터의 데이터분포의 유사도를 확인해보았습니다.

    'user_id' / 'isbn'/ 둘다 인 세가지의 경우에 비교를 해 본 결과 모두 0.5에 비슷한 분포를 가지고 있다는 것을 확인할 수 있었습니다.

- **모델개요**
  - FM : SVM 과 Factorization Model의 장점을 결합한 모델
  - FFM : 입력 변수를 필드(field)로 나누어, 필드별로 서로 다른 latent factor를 가지도록 factorize
  - NCF : Neural Net 기반의 architecture인 Neural Collaborative Filtering

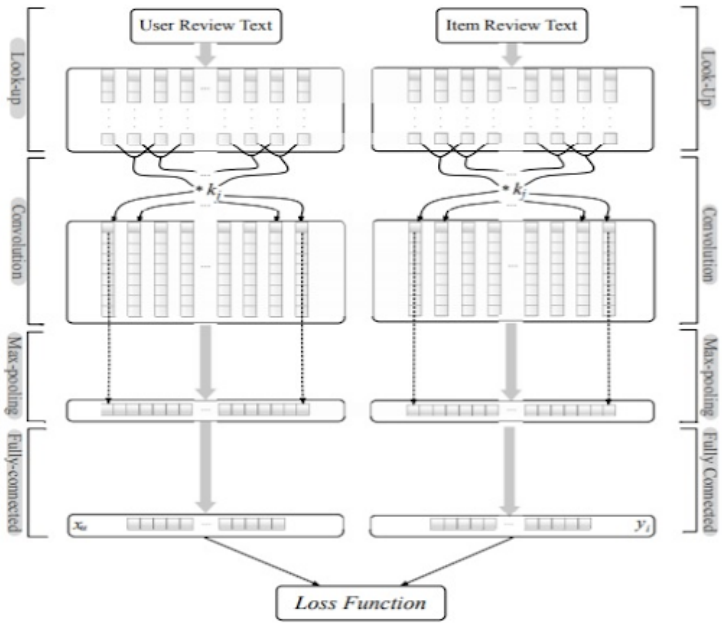
- WDN : 선형적인 모델과 비선형적인 모델을 결합한 모델
- DCN : Cross Network를 통해 feature를 교차 학습함으로 써, 여러 feature 간 특성이 같이 고려된 추천을 하는 모델
- CNN_FM : 시각적 선택을 고려하기 위한 CNN모델과 Feature interaction을 고려할 수 있는 FM 모델을 결합한 모델
- DeepCoNN : User와 Item의 Text 컨텐츠를 2개의 병렬 TextCNN 구조를 통해 User와 Item간의 임베딩을 학습하는 모델

- **모델 선정 및 분석**

1. DeepCoNN, CNN_FM 모델 선정 및 분석
   EDA를 통해 평점 데이터가 99.996% Sparse한 데이터셋인 것을 확인하였습니다. 이런 Cold Start 문제를 활용하기 위해 텍스트나 이미지와 같은 비정형 데이터를 활용하여 성능 개선을 하고자 해당 모델을 사용하였습니다.
2. FFM 모델 선정 및 분석
   책에 대한 텍스트 데이터와 이미지 데이터가 한정적임으로 인해, DeepCoNN과 CNN_FM을 통한 Cold Start 개선이 어느정도 이상 되지 않았습니다. 이 문제를 해결하기 위해, 비교적 다양한 피처들이 존재했던 User에 대한 Context 정보와 Books에 대한 Context 정보를 활용하는 것 이었습니다. 그래서 FFM 모델을 사용하였습니다.

- **모델 평가 및 개선**

1. DeepCoNN 모델 평가 및 개선



   Books의 Summary Text 데이터를 메인으로 사용하는 Text 기반 모델입니다. 2개의 병렬 TextCNN 구조를 활용하여 하나는 User가 읽은 책에 대한 Summary 정보, 다른 하나는 책 자체의 Summary 정보를 활용하였습니다. Summary 정보를 books의 title로 채워보는 등 다양한 데이터 프로세싱으로 성능을 개선시켜 보았습니다. 하지만 데이터를 채우는거로는 큰 성능 개선이 되지 않았고, EMBED_DIM과 LATENT_DIM 등의 파라미터를 활용한 성능 개선의 폭이 더 컸습니다.

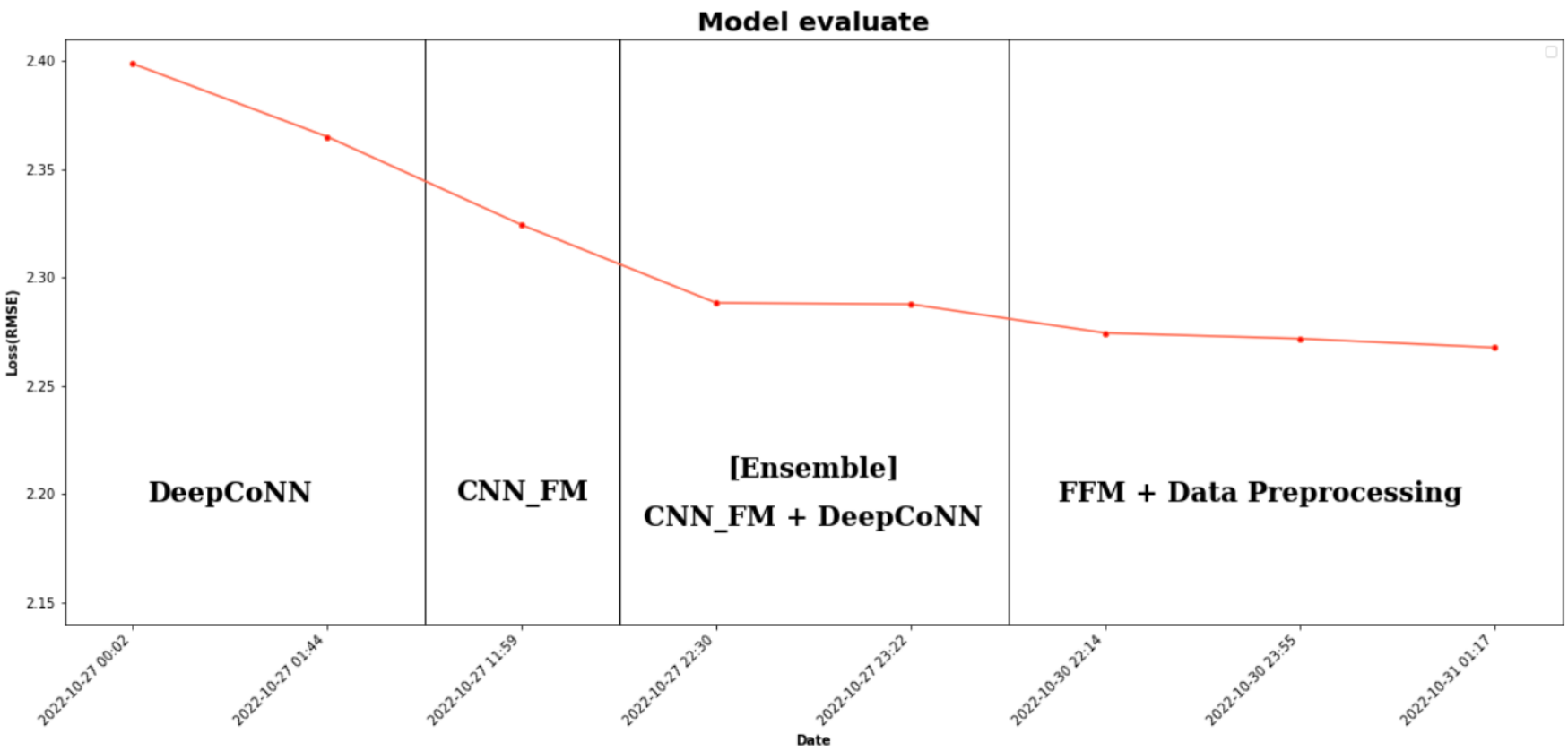
2. CNN_FM 모델 평가 및 개선
   책의 표지 이미지로 부터 특징을 추출해 Cold Start 문제를 완화할 수 있는 이미지 기반 모델로, 이미지 처리를 하는 CNN 모델과 Latent Factor를 통해 Feature interaction을 고려할 수 있는 FM 모델을 결합한 모델입니다. 이미지 표지 컬럼의 경로에는 결측치가 없었지만, 이미지 자체가 보라색으로 채워져있거나 하는 이상 데이터들이 꽤 많았습니다. 이미지 데이터를 추가로 스크래핑하는것은 허용되지 않아서, 역시 하이퍼 파라미터를 조정하며 성능 개선을 해보았습니다.
3. FFM 모델 성능 개선
   FFM모델은 확실히 FM 모델보다 성능이 좋았습니다. FM은 모든 latent vector간 interaction을 하나의 벡터로 표현하기 때문에 생기는 한계를, 필드마다 latent vector를 만들어 개선했기 때문입니다. Users에 존재하는 location, age 정보와 books의 category, publisher, language등의 피처들을 다양한 방식으로 processing 하며 성능을 개선시켰습니다.
4. DCN / NCF 모델 성능 개선

   두 모델 모두 조치에 피쳐로 ‘user_id’와 ‘isbn’만을 가지고 동작하였습니다. 추가적으로 피처를 더해준다면 성능의 개선이 이루어 질수 있다고 판단하여 다양한 피처를 추가하고 결과를 비교하는 방식으로 가설을 검증했습니다. 이때 2개 이상의 피처를 추가할 경우 성능 향상은 이루어 지지만 하나의 피처만을 추가했을때 보다 상승폭이 낮아서 하나의 피쳐만을 사용하도록 진행했습니다. 두개의 모델 모두 location을 추가했을때 대략 2.8% 성능 향상이 이루어졌습니다.
5. WDN 모델 성능 개선

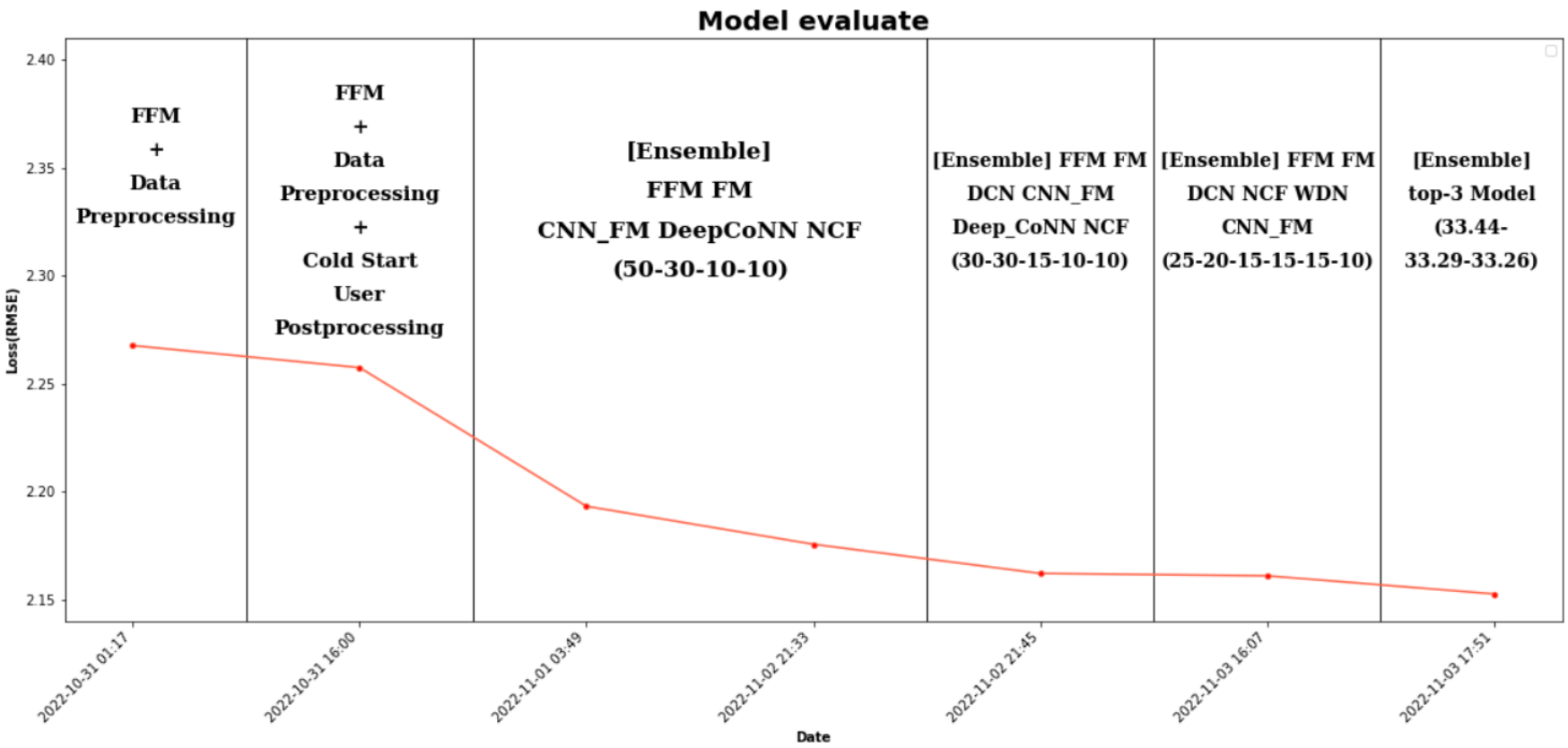
   context model과 같이 feature를 모두 적용하였습니다. overfitting이 쉽게일어나는듯하여 Dropout 값을 Default 0.1 → 0.5로 높여주어 최적값을 찾았습니다. MLP_LAYER는 16의 값으로 여러개 쌓는 값이 가장 잘 나왔습니다. EMB_DIM은 점점 높여가며 val_loss를 비교한 결과 32에서 최적의 결과가 나오는 것을 확인하였습니다.

- **시연결과**

  1주차 결과 :



  2주차 결과 :



  ▼ 세부 정보

  - 10/27 :
    - DeepCoNN(Default) : 2.3987
    - DeepCoNN (EPOCH 600, Batch_Size 256, EMBED_DIM 600, LATENT_DIM 60) : 2.365
    - CNN_FM (EPOCH 600, Batch_Size 256, EMBED_DIM 600, LATENT_DIM 60) : 2.3243
    - [Ensemble] CNN_FM DeepCoNN (70-30) : 2.2883
    - [Ensemble] CNN_FM DeepCoNN (50-50) : 2.2877
  - 10/30 :
    - FFM (Users - Location Processing) : 2.2744
    - FFM (Books - Category Processing) : 2.2718
  - 10/31 :
    - FFM (Books - Category Processing + 200 Epoch) : 2.2677

    - FFM (train set에 없고 test set에 있는 User 후처리) : 2.2575
- 11/1 :
    - [Ensemble] FFM FM CNN_FM DeepCoNN NCF (50-30-10-10) : 2.1933
- 11/2 :
    - [Ensemble] FFM FM CNN_FM Deep_CoNM NCM (30-30-15-10-10) : 2.1757
    - [Ensemble] FFM FM DCN CNN_FM Deep_CoNN NCF (35-30-12-12-7-4) : 2.1622
- 11/3 :
    - [Ensemble] FFM DCN NCF WDN FM CNN_FM (25-20-15-15-15-10) : 2.1611
    - [Ensemble] Top-3 Model (33.44-33.29-33.26): 2.1509

▼ [자체 평가 의견]

프로젝트의 목표는 여러 모델의 최적 옵션을 찾아내어 최대 성능을 끌어올린 다음 각각의 모델을 앙상블하여 최종적으로 높은 정확도를 갖는 모델을 완성하는 것이었습니다. 효율적인 진행을 위해서 다양한 협업툴을 사용했으며, 각 팀원이 책임감을 가지고 맡은 역할을 수행하여 최종적으로 RMSE 2.1509 값을 갖는 모델을 완성하였습니다. 계획한 내용을 대부분 달성하였고, 가설과 검증을 통해 얻은 모델인 점을 고려한다면 계획 대비 달성도는 90%이상으로 볼 수 있는 모델입니다.

다만, Cold Start를 처리하는 부분에서 아쉬운 점이 남았습니다. 데이터 EDA를 하는 과정에서 Cold Start문제를 인지하였고 해결하기 위한 계획을 수립하였으나, 수립된 계획으로 주목할 만한 효과를 얻어내는데 어려움을 겪었습니다. 결국 원하는 만큼의 개선을 이루지 못한 채 마무리하게 되었습니다. 평점을 예측하듯 결측치를 예측하여 채워 넣는 방식을 구상해보았는데 실제로 적용해보지 못한 점이 아쉬움으로 남았습니다.

추가로 베이스 라인 코드에 등장하지 않은 모델을 사용했다면 더 좋은 성능을 낼 수 있었을 것으로 보입니다. 주어진 데이터가 굉장히 sparse한 상태였는데 베이스 라인에 주어진 모델들로 이를 극복해가는 방향으로 프로젝트를 진행했습니다. 그 때문에 주어진 상태에서 최적의 성능을 발휘하였지만, 데이터 특성에 맞는 모델을 사용한 팀과 비교하였을 때 부족한 성능을 보였습니다.

위의 내용을 종합하였을 때 아쉬운 점이 존재하지만, 계획하고 목표한 바를 대부분 달성했기 때문에 성공적이었다고 평가 내렸습니다. 작성된 모델은 사용자 데이터를 고려하여 책에 대한 독자들의 미래 평점을 예측한다는 점에서 프로젝트의 의도에도 부합한다고 할 수 있습니다. 더하여 이 모델에 사용한 데이터는 쉽게 수집할 수 있는 정보이며 추가로 수집된 데이터가 빠르게 반영될 수 있는 구조를 가지고 있기 때문에 실무에서도 충분히 사용 및 개선이 이루어질 수 있는 모델입니다.

<개인 회고>

1. 이나현

나는 내 학습목표 달성하기 위해 무엇을 어떻게 했는가?

저의 학습목표는 “데이터를 많이 다뤄보고, 하나의 통찰을 얻기” 입니다.

이를 위해 저는 corr함수와 heatmap을 이용한 시각화를 통해 user의 feature들과, book의 feature이 rating에 어떤 영향을 미치는지 파악해보고자 했습니다.

또한 Adversarial Validation이라는 방법으로 train데이터와 test데이터의 분포의 유사도를 ‘user_id’, ‘isbn’, ‘user_id와 isbn’의 관점으로 확인해보고자 했습니다.

마지막으로 성능을 올릴 수 있도록 작가와 location 정보에 대해 결측치를 채워보거나, ‘age’의 등분을 다르게 해보는 등 여러 가설을 세우고 검증(rmse)해보았습니다.

위의 세가지의 노력을 해보았으나, 유의미한 통찰을 얻지는 못했습니다. 아직 더 많은 경험이 필요함을 느꼈습니다.

나는 어떤 방식으로 모델을 개선했는가?

이번 대회에서 제가 주로 맡은 역할은 데이터 전처리와 데이터를 이용한 딥러닝 계열 모델 성능향상이었습니다.

딥러닝 계열의 모델은 ‘user_id’와 ‘isbn’정보만을 가지고 rating을 예측하는 것을 발견하고, context 기반 모델처럼 다른 feature들도 모델에 같이 넣어보았습니다. 그 결과 ‘location’과 ‘age’, ‘book_author’를 각각 같이 넣어줬을 때 미세하게 성능을 올릴 수 있었습니다.

이 과정에서 feature를 넣기위해 feature마다 다른 전처리를 하는등 번거로운 과정이 있었습니다. 이를 개선하고자 3줄만 고치면 feature를 적용해 볼 수 있도록 코드를 일반화하도록 노력했습니다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

평점을 한번만 받은 책 또는 평점을 한번만 준 유저는 해당 평점으로 과적합 될수도 있다는 생각이 들어 steam rating 이라는 방법을 응용하여, 조금 더 평균값으로 값을 보정하고자 했습니다.

그 결과, 각 모델의 rmse 값이 이 0.2~3정도 줄어드는(성능향상) 유의미한 효과를 볼 수 있었습니다.

하지만 알고보니 train 데이터를 train과 validation set으로 split하기 전에 rating 값을 변경하여서 validation의 값에도 영향을 주어 성능이 올라갔음을 뒤늦게 알게되었습니다.

이를 통해 target값을 바꿀 수 있는 데이터 전처리는 위의 경우를 주의해서 split하기전에 적용해줘야한다는 것을 깨달았습니다.

전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

터미널 대신 쥬피터노트북에서 실험을 하거나 모델을 돌려서, 결과값을 체계적으로 기록할 수 있었습니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

판다스가 아직 익숙하지않아 데이터에 대한 하나의 가설을 세운 후 확인하는 과정이 오래걸린 점이 아쉽습니다.

생각보다 데이터에 대한 가설이 성능에 큰 효과를 내지 못한 점이 많이 아쉽습니다.

steam rating을 적용하고 validation rmse만 확인할 것이 아니라, 직접 제출을 해보면 조금 더 위의 실수를 빨리 파악할 수 있었을 것 같습니다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

다음 프로젝트 전, 실습코드나 과제코드에 주어지는 데이터 전처리를 눈으로만 보지않고, 직접 다시 짜보면서 pandas에 익숙해지도록 노력해봐야겠습니다.

대회 초기에는 제출횟수가 아깝다고 생각하지말고 제출해서 validation의 것뿐만아니라 실제 성능을 확인해봐야겠습니다.

2. 채민수

- 이번 프로젝트에서 나의 목표는 무엇이었는가?

  첫번째로 참가하는 대회인 만큼 대회 환경에 익숙해지는 것을 목표로 설정했습니다. 특히 서버 환경에서의 작업과 깃허브를 통한 협업에 적응하는 것을 우선에 두었습니다.

  두번째 목표는 주어진 데이터를 정확히 이해하는 것입니다. 데이터의 특성을 파악하고 어떠한 모델이 적합한지, 모델에 적용하기위해 어떠한 처리를 해주어야하는지를 가설을 세우고 검증해나가는 형식으로 파악할 것입니다.

  마지막 목표는 각 모델이 어떤 특성을 가지고 있는지 이해하고 파라미터 조정과 피처 튜닝을 통해 어떻게 성능이 변하는지를 확인하고 싶습니다. 이 과정을 통해 여러가지 모델의 최적의 상태를 찾고, 이들을 앙상블 했을때 성능이 어떻게 변하는지를 확인할 것입니다.

- 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

  먼저 팀원들과 함께 프로잭트 깃허브 사용 규칙을 정하였습니다. 특히 목적에 따른 브랜치를 만들고 해당 브랜치에서 정해진 작업만 수행하는 것으로 협업을 진행했습니다. 가능한 commit을 자주하여 버전 관리와 깃에 익숙해지려고 노력했습니다.

  데이터와 모델에 관한 부분을 가설을 세우고 검증하는 방식으로 특성을 파악하고자 노력했습니다. “이러한 변화를 주면 이러할 것이다”라고 가정한뒤 결과를 확인하는 과정이 모델과 데이터가 변화에 따라 어떤식으로 반응하는지 이해하는데 많은 도움이 되었습니다.

- 나는 어떤 방식으로 모델을 개선했는가? + 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

  성능이 향상될 것으로 기대되는 가설을 세우고 검증하여 가설이 맞다면 적용하고 틀리다면 적용하지 않는 방식을 사용했습니다. 영역은 데이터, 모델, 앙상블 이렇게 3가지로 나누어 적용했습니다.

  데이터 영역에서 무엇이 필요한 데이터인지 확인하는 작업에 집중했습니다. 책을 추천하는데 의미가 없다고 판단되는 데이터를 제거한 후, 성능의 변화를 관찰했습니다. 제거 후 성능이 개선되었다면 추가 과정없이 해당 데이터를 제거하였고 그렇지 않을 경우엔 모델에 필요한 데이터이므로 결측치 처리 등의 과정을 거치며 모델에 적용하기 좋은 형태로 수정했습니다.

FM으로 모델을 고정시키고 users의 location, books의 language와 publisher를 각각 제거해 보았습니다. 이때 location과 publisher의 경우 각각 제거시 2.67%, 1.46%의 성능 개선이 이루어 진것을 확인했습니다. 이때, 두가지 피처를 동시에 제거하면 성능이 2.64% 향상되어 location만 제거했을때 보단 높지 않다는 것을 발견했습니다. 반대로 language의 경우 제거시 -0.05% 성능 하락을 야기했기 때문에 제거하지 않는 것으로 정하였습니다.

모델 검증의 경우 팀원들과 분업하여 위와 유사하게 진행했다. 저는 DCN과 NCF모델이 'user_id'와 'isbn'만을 사용한 것을 확인하였고, 다른 피처가 추가되면 성능이 향상 되는지를 기록했습니다. 해당 모델들의 경우 피처가 한 가지만 추가되었을때 가장 높은 성능 개성이 이루어 졌습니다. 앞서 확인한 FM 모델과는 반대로 users의 location 피처를 추가하였을때 2.8% 정도 성능이 높아졌습니다.

마지막으로 앙상블을 통해 성능을 최대화 했습니다. 성능이 잘 나온 모델끼리 결합했을때 더 좋은 선능을 보일 것이란 가정에 입각하여 앙상블을 진행했습니다. 이때 softmax에서 차용한 기법을 통해 성능이 좋았던 모델이 더 많이 반영될 수 있도록 조정했습니다. 해당 과정을 통해 RMSE 값을 2.1622에서 2.1509 까지 낮추는데 성공했습니다.

- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

  첫번째로 스스로 세운 가설이 모델 성능을 향상 시키는데 기여한 바가 적다고 느껴졌습니다. 대부분의 가설이 "가설이 맞다면 성능이 개선될 것이다"라는 식이었는데, 가설이 맞는 경우가 많이 없었습니다. 특히 데이터 가설의 경우 어떤 모델에선 가설이 참이지만 다른 모델에서는 거짓인 경우가 많아 일괄적인 적용이 어려웠습니다.

  두번째로 주어진 모델에 너무 집중한 점이 아쉬웠습니다. 데이터의 특성을 파악하고 적합한 모델을 탐색하는 과정에서 제공된 모델 이외의 모델을 적용해보지 못하였습니다. 필요성은 느꼈지만 시간적인 문제로 적용해보는 단계에 도달하지 못 했습니다. 결과적으로 봤을때 적합한 모델이 존재했다는 점이 더욱 아쉬웠습니다.

  저 자신의 구현 능력도 아직 만족할 만한 수준이 아니였습니다. 생각한 내용을 베이스라인 코드에 적용하는데 어려움이 있었습니다. 빠른 작업을 위해 실수 없이 코드를 작성하는 능력도 키워나가야 할 부분으로 느껴졌습니다.

- 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

  대회 후반, 각 모델 별로 최적의 파라미터 및 피처를 찾아내어 더 이상 성능 향상이 이루어 지지 않는 시점이 왔습니다. 다만 만족할 만한 성능이 아니었기 때문에 성능 향상이 필요 했습니다. 저는 지금이야말로 앙상블이 필요한 시점이라고 생각되어 현재까지 팀에서 제안한 모델 중 최상위 3개의 모델을 모아서 앙상블하기로 결정했습니다. 이때 가중치 비율은 softmax에서 착안하여 성능이 좋은 모델일 수록 더 많이 반영될 수 있도록 했습니다. 결과적으로 해당 과정을 통해 RMSE 값을 2.1622에서 2.1509 까지 낮추는데 성공했고 팀의 최종 제출로 사용되었습니다. 이 과정에서 앙상블을 통한 지속적인 개선이 이루어 질 수 있다는 것을 느꼈습니다.

- 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

  베이스 라인 코드에 너무 집중하지 않고 적합한 모델이 있다면 적극적으로 탐색하여 적용할 계획입니다. 또한 데이터의 특성을 파악하기 위해 EDA 단계에서 더 많은 시간을 할애할 것입니다.

3. 조원준

1. **이번 프로젝트에서 나의 목표는 무엇이었는가?**

   데이터를 관찰하고, 모델을 관찰해서, 데이터를 기반으로 가설을 세우고, 이를 검증해가며 모델의 예측 성능을 높혀가는 경험을 해보는것이었습니다.

2. **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

   노션과 트렐로, 구글시트를 통해 가설을 세우고 검증하는 시스템을 팀원들과 함께 만들어보았습니다. 이 시스템에 각자 가설에 대한 검증을 기록하고, 매일 회의를 하며 인사이트를 공유했습니다. 외에도 Github를 통해, 데이터를 고정하고 모델파트의 코드를 수정하는 model branch와 모델을 고정하고 데이터 프로세싱 코드를 수정하는 data branch로 나누어 각자 작업을 했습니다. 저는 이런 체계를 만드는데 기여했고, 이런 체계 안에서 가설을 세우고 검증했으며 공유하는 노력을 했습니다.

3. **나는 어떤 방식으로 모델을 개선했는가?**

   모델을 개선한 방식은 크게 하이퍼 파라미터 수정과 입력 피처 데이터를 수정하였습니다. 주로 데이터를 어떻게 변화시키면 성능이 오를지 가설을 세우고, 검증해가며 데이터를 깊게 이해해가는 방식을 사용했습니다.

4. **내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?**

   저는 Data Centric AI가 매우 중요한 흐름이라고 여겨, 데이터에 대한 인사이트를 키우는 방향으로 주로 가설을 세우고 검증해갔습니다. 그래서 데이터에 대해서는 깊게 이해해볼 수 있었지만, 모델의 성능을 높히는 방향은 부족했던 것 같습니다. 대회에서는 모델기반으로 어떻게 하면 모델의 성능을 높힐지 몰입해보는게 좋겠다는 깨달음을 얻었습니다. 그리고, CV set의 중요성과 단순히 Validation loss가 감소했다고 성능이 증가하여 가설이 검증됐다고 하기 보단, 정확히 어떤 그룹의 validation loss가 낮아진건지, 어떤 조작변인에 의해 낮아진건지를 통해 좀더 깊게 가설을 이해해보고 활용해봐야 겠다는 깨달음을 얻었습니다.

5. **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

git을 통해 데이터를 고정하고 모델을 수정해서 가설을 검증해보는 model branch 팀과 모델을 고정하고 피처 프로세싱을 수정하는 data branch팀을 나누어 개발해보았습니다. 서로 역할 분담이 명확해 졌고, data 수정한 사항과 모델 수정한 사항이 겹치지 않아 성능검증이 수월했습니다. 데이터에서 가설 검증이 명확해진 부분은 model 브랜치에 merge하여 조작 변인의 분리와 합치는게 명확해진다는 점이 좋았습니다. 또한, Trello를 이용해 각자 현재 어떤 일을 진행 중인지, 어떤 업무가 끝났고 검증된 가설이 무엇이있는지를 실시간으로 한 눈에 볼 수 있던게 좋았습니다. 그리고, Slack Bot을 통해 모델이 다 돌면 그 결과 모델과 하이퍼파라미터, 성능이 알림으로 와서, 여러 실험을 하면서도 모델들이 어떻게 돌고있는지 Follow up 하기 수월했습니다.

6. **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

   Optuna와 wandb를 사용하여 최적 Hyper Parameter를 편하게 찾아볼 수 있었는데 시도해 보지 못한게 아쉬웠습니다. 또한, 주어진 데이터가 매우 Sparse한 Cold Start이고, 범주형 변수들이 많다는 점을 고려해 부스팅 기반의 알고리즘 특히 CatBoost를 시도해 보지 못한게 아쉬웠습니다.

7. **한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?**

   a. wandb 연결 및 Optuna 사용

   b. 예측을 하는 train set을 그룹별로 묶어서 validation loss가 좋아졌다면 어떤 그룹에서 좋아진건지 시각화 및 분석해보기.

   c. 현재 데이터는 어떤 모델에서 성능이 좋을지 가설을 세워보고 검증해 보는 것.

4. 정의준

1. 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

▼ 가설 설정과 검증 과정

▼ isbn에 따른 language결측치 대체 값

```python
import tqdm
lang_dic = {}

for i in tqdm.tqdm(range(100), smoothing=0, mininterval=1.0):
    isbn = str(i)
    if len(isbn) == 1:
        isbn = '0' + isbn
    if len(books[books['isbn'].str[:2] == isbn].groupby('language').count()['isbn']) != 0:
        lang_dic[isbn] = books[books['isbn'].str[:2] == isbn].groupby('language').count()['isbn'].sort_values(ascending=False).keys()[0]
```

```
 '29': 'fr',
 '30': 'de',
 '31': 'de',
 '32': 'de',
...
 '95': 'es',
 '96': 'es',
 '97': 'pt',
 '98': 'en',
```

- isbn의 첫 두 숫자에 따라 language의 분포가 한 쪽으로 몰려있는 점을 확인하였습니다.
- language의 결측치를 isbn에 따라 채워주었지만 성능 향상에는 영향이 없었습니다.

▼ 나라별 평균으로 age 결측치 채우기

```
# age를 나라별 평균으로 채워줌
# 나라와 age모두 결측값일 경우 전체 평균으로 대체
users = users_save
age_mean = users[users['age'].notnull()].groupby('location_country')['age'].mean()
age_null_idx = users[users['age'].isnull()].index
location_null_idx = users[users['location_country'].isnull()].index

for i in age_null_idx:
    if i not in location_null_idx and users.loc[i,'location_country'] in age_mean.keys() :
        users.loc[i,'age'] = age_mean[users.loc[i,'location_country']]
    else :
        users.loc[i,'age'] = age_mean.mean()

users['location'] = users['location_city'] +','+ users['location_state'] +','+ users['location_country']
users = users.drop(columns = ['location_city','location_state', 'location_country'])
users['location'] = users['location'].fillna('na,na,na')
users.to_csv('./data_age_avg/users.csv',index = False)
```

- val_loss 2.274 → 2.275 미비하지만 성능은 떨어졌습니다.
- 위 가설을 진행할 때 까지만해도 결측치는 최대한 없애주는게 성능에 도움을 줄거라 믿었습니다.
- 멘토님의 말씀과 프로젝트가 끝나고 다른 팀들의 발표를 들으면서 결측치를 꼭 대체하는 방법보다는 다른 관점도 고려해야 한다고 느꼈습니다!

▼ Cold start user의 rating 값 대체

- user정보가 없으니 isbn만을 이용하여 train에서 해당 책 평점의 평균을 불러왔습니다.
- val_loss 2.28 → 2.3으로 성능이 떨어졌습니다.
- 책 정보만을 가지고 대체가 가능할 것이라 생각했지만 user 개인의 특성도 영향력이 크다는 점을 알게 되었습니다.

▼ DeepCoNN 모델 summary값 변경

- 구글링을 통해 영화 추천에서 유사도를 계산할 때 제목과 장르, 감독 등을 한 Column에 모아 Word2Vec 사용하는 과정을 보고 가설을 세웠습니다.
- DeepCoNN이 Books data에 summary의 값을 인자로 BERT모델을 통해 vector로 반환하기 때문에 제목,저자, 언어, 카테고리의 내용을 summary에 합쳐준다면 더 특성을 잘 나타낼 수 있지 않을까 생각하였습니다.
- 결과적으로 성능이 향상되지는 않았습니다.

2. 나는 어떤 방식으로 모델을 개선했는가?

▼ 모델 학습에 기여한 부분

▼ WDN모델의 최적 Hyper Parameter 찾기

EMBED_DIM	32	64	128	256	512
val_loss	2.0897	2.0902	2.1097	2.1096	2.1178

- WDN의 EMBED_DIM을 비교해보며 val_loss가 가장 낮게 나오는 값을 찾았습니다.
- 모두 EPOCH = 5 이전에서 val_loss가 증가하는 경향을 보여 오버피팅이 쉽게 일어나는듯 하여 WDN_DROPOUT의 값을 0.4, 0.5로 높여주었고 더 높은 EPOCH에서 낮은 val_loss를 얻을 수 있었습니다.
- MLP_DIMS는 값을 높여가며 비교해봤지만 낮은 값을 여러개 쌓았을 때 val_loss를 낮출 수 있었고 (16,16,16,16)의 값을 넣었을 때 최저 val_loss값을 얻을 수 있었습니다(2.0897 → 1.9468)

▼ 이전에 학습했던 모델을 이어서 학습할 수 있는 코드 추가

- 조원준님의 의견에 따라 이전에 했던 모델학습을 이어할 수 있는 코드를 추가하였습니다.

```
arg('--PRETRAINED', type=str, default=None, help='pretrain 파일을 불러옵니다.')
arg('--MIN_VAL_LOSS', type=bool, default=True, help='Val_loss가 가장 작을 때를 기준으로 모델을 저장합니다.')
```

- main.py에 argument를 추가해 편하게 쓸 수 있게 했습니다.

```
self.min_val_loss = args.MIN_VAL_LOSS
self.pretrained = args.PRETRAINED
```

```python
if self.pretrained is not None:
    self.model.load_state_dict(torch.load(self.pretrained))
    print('--------------- PRETRAINED_MODEL LOAD ---------------')
```

- PRETRAINED를 main.py의 args로 받아 이전의 모델을 불러와 학습을 할 지 여부를 판단하였습니다.
- pt파일을 인자로 줄 경우 해당 모델의 parameter를 불러온 다음 학습을 시작하게 하였습니다.

```python
if minimum_loss > rmse_score:
    minimum_loss = rmse_score
    if not os.path.exists('./models'):
        os.makedirs('./models')
    torch.save(self.model.state_dict(), './models/{0}_{1}.pt'.format(self.save_time,self.model_name))
```

```python
if self.min_val_loss:
    self.model.load_state_dict(torch.load('./models/{0}_{1}.pt'.format(self.save_time,self.model_name)))
    print('--------------- MIN_VAL_LOSS STATE LOAD ---------------')
```

- 모델마다 min_val_loss의 값을 가지는 파라미터로 예측을 하는 경우와, 학습이 종료된 시점의 파라미터로 예측을 하는 경우가 나뉘어 있었기 때문에 main.py에 MIN_VAL_LOSS인자를 주어 둘 중 선택할 수 있는 기능을 추가하였습니다.

3. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 데이터 분포를 좀 더 자세히 보지 않았던 점

▼ Baseline Code의 모델 외에 사용하지 않았던 점

- Mission에서 더 다양한 모델을 소개해 주었지만 실제 프로젝트 진행에서 사용해봐야겠다고 느꼈을 때는 이미 마지막 날이 되어 사용하지 못했습니다
- 대부분 상위권 팀에서는 catboost 모델을 이용하였기 때문에 아쉬웠습니다.

▼ 구현을 하고 싶었지만 못했던 점

- K-fold CV를 이용한 val_loss를 얻고 싶었지만 baseline code에서 적용하기에는 능력이 부족하여 구현하지 못했습니다.

4. 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 깃허브, 트렐로, 슬랙 봇 등 이번 프로젝트에서 팀 단위의 협업 과정은 만족스러웠고 앞으로도 중요다고 느꼈습니다. 여러 사람의 생각과 진행상황을 정리하고 공유하는 과정은 효율성에서 큰 도움을 준다고 생각했기 때문입니다.
  앞으로도 계속 프로젝트를 진행하며 협업 시스템을 발전시킨다면 다른 팀과의 차별성을 기를 수 있을 것 같습니다!!
- 혼자서 기능을 테스트 하는 것과 남들도 쉽게 알아보고 사용할 수 있게 기능을 만드는 것의 난이도 차이를 크게 체감하였습니다. 좀 더 가독성 좋은 코드를 만드는 방향으로 연습이 필요할 것 같습니다.
- 팀 단위로 프로젝트를 진행할 때 남들에게 말로만 설명하기 어려운 부분은 시각화를 적극적으로 활용하는게 훨씬 더 효율적이라고 느꼈습니다.
- 모델의 성능을 검증할 때 단순 주어진 val_loss만이 아닌 data의 분포, loss를 계산하는 과정, 그리고 loss가 성능을 충분히 대변할 수 있는 지 등에 대한 생각도 논리적으로 고려해야 한다고 느꼈습니다.
- 아직 머신러닝에 대한 기초적인 이론이 실전에 사용할 정도로 능숙하지 않다고 느꼈습니다. 이전에 했던 강의를 다시 보거나 다른 자료를 통해 깊게 학습하는 과정이 필요하다고 생각했습니다.
- 프로젝트 진행도를 한 눈에 볼 수 있는 캘린더 형식의 도구도 찾아본다면 좋을 것 같습니다.

5. 전해리

- **이번 프로젝트에서 나의 목표는 무엇이었는가?**

  저는 그동안 대회나 프로젝트 경험이 없었기 때문에 이번 프로젝트에서 최우선의 목표는 대회를 경험해보는 것이었고 대회를 잘 마치는 것이 목표였습니다. 여기에 더해서 level1에서 배웠던 내용을 대회에 적용해보고 개발시켜보는 것이 목표였습니다.

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  저는 무엇보다 팀원 간의 소통에 노력하였습니다. 대회가 잘 진행이 되려면 무엇보다 팀원들 간의 소통이 중요하다고 생각했기 때문입니다. 서로의 진행 상황이나 실험 결과 등을 공유하며 진행해야 보다 효율적으로 대회를 이끌어 갈 수 있다고 생각했습니다.

- **나는 어떤 방식으로 모델을 개선했는가?**

저는 현 상황에서의 문제점을 찾고, 이를 해결하기 위해 모델을 개선하였습니다. 예를 들어 이번 대회에서는 콜드 스타트 문제가 있었는데, 이를 해결하기 위해 컨텐츠 기반 모델을 개발하려 하였습니다. 또한 모델 튜닝을 통해 최적의 파라미터를 찾는 것에 몰입하였습니다.

- **내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?**

  콜드 스타트 문제를 위해 컨텐츠 기반 모델을 개발했으나, 개발에만 집중하고 이를 적용하기 위한 방법까지는 생각하지 못했습니다. 따라서 수많은 데이터로 인해 메모리 초과 오류가 났고 이를 뒤늦게 개발하여 비록 직접 적용해보진 못했지만 다음에 있을 대회를 위해 모델 개발은 데이터의 위에서 이루어져야 하는 등 많은 깨달음을 얻을 수 있었습니다. 더하여 NCF 모델 실험을 통해 최적 파라미터 값을 파라미터 값에 따른 모델의 변화도 알 수 있었습니다.

- **내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

  데이터 전처리 과정에서 데이터의 결측값을 평균값으로 채워주지 않고, 결측치가 있는 행을 아예 제거하고 예측해보았습니다. 예상 외로 rmse가 높게 나오지 않아 놀랐습니다. 그 전에는 결측치를 제거하는 것에 부정적인 생각이 있었는데, 이를 보고 결측치를 채우는 방법 말고 제거하는 방법도 다시 생각하게 되는 계기가 되었습니다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  대회를 진행하며 마주했던 제일 큰 한계는 '시간적 한계'였습니다. 처음엔 여유가 있다고 생각했으나, 데이터 EDA와 실험을 진행하면서 많은 시간이 소요되었고 결국 콜드 스타트 문제를 뒤늦게 해결하려하다보니 개발한 콘텐츠 기반 모델에서 메모리 초과 오류가 났는데 이를 결국 해결하지 못하였습니다. 다음 대회에서는 일정 관리를 더욱 체계적으로 하여 시간적 한계를 극복해야겠다고 생각했습니다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?**
  다음 대회에서는 데이터 EDA를 보다 타당한 이유로 체계적으로 진행하고, 데이터 파악을 통해 어떤 모델이 가장 적합할 지 파악하는 것을 제일 첫번째 목표로 해야겠다고 판단했습니다. 또한 주어진 베이스라인 코드를 넘어서 강의에서 배운 내용을 적극 활용해봐야겠다고 생각했습니다.