

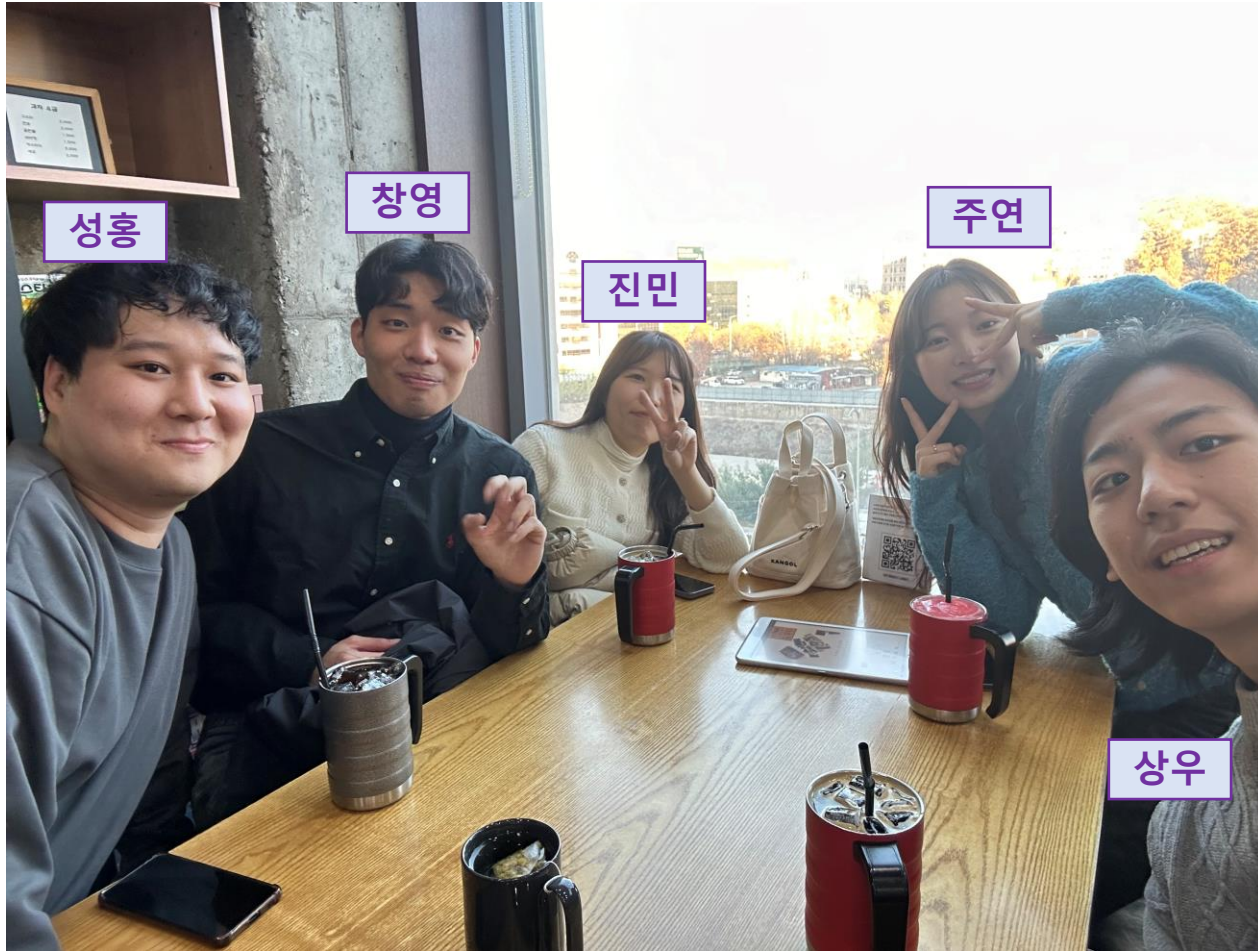
온라인 성능은 우리가 더 좋을지도..

Recsys-02 TMI조

목차

- 팀 소개
- 프로젝트 개요 및 학습 목표 정의
- 프로젝트 진행 방식
- 아이디어와 가설 설정
- 실험 결과 및 해석
- 잘한 것 & 아쉬웠던 것 & 배운 것
- 궁금한 것

팀 소개



TMI 조

: 많은 데이터에서 사용자에게 필요한 정보를 추천하겠다

- 우리 조의 자랑
 - 가족같은 분위기
 - 매우 수평적인 구조
- 컨셉
 - 질 수 없잖아

프로젝트 개요 및 학습 목표 정의

프로젝트 개요

- 주제: 소비자들의 책 구매 결정에 도움을 주기 위한 개인화된 상품 추천 대회
- 배경
 - 일반적으로 책을 읽는 데에는 다른 콘텐츠 대비 많은 시간이 필요함. 소비자가 높이 평가할만한 책을 제시하는 것이 클릭 확률, 구매 확률보다 소비자에게 더 큰 가치를 제공하는 것일 수 있음.
 - 플랫폼에서 기록 가능한 정보를 통해 사용자가 책에 대해 어떤 평가를 내릴지 예측할 수 있다면, 플랫폼에서 소비자의 책 구매 결정에 도움을 줄 수 있음.
 - 대회에서는 사용자와 책의 특성 정보 및 사용자가 책에 대해 부여한 평점 정보를 활용하여, 사용자가 모르는 책에 대해 어떤 평점을 부여할지 예측하는 알고리즘을 개발하는 것이 목표
- task: 개인화 추천을 위한 책 **평점 예측** 문제

학습 목표

- 대회 점수에 집착하지 않는다
- 처음 접하는 대회이니, 분업보다는 각자 end-to-end를 경험해본다
- 대신 서로 시도해본 것들에 대해 계속 공유하고, 질문하고, 토론한다
- 모델 하나를 깊게 파고 이해한다
- 궁금하면 일단 시도해본다

협업 방식 (1/2)

- 1주차에는 개인 가설 설정, 실험의 반복
- 2주차에는 그 내용을 병합하고 앙상블
- 베이스라인 활용, Github Issue 기반 작업
- 데일리 스크럼/피어세션 활용해 핵심 Feature PR 코드 리뷰 후 병합

1주차: 개인 작업 및 실험 수행

- 환경 설정 및 베이스라인 코드 병합
- 개인별 관심 모델 구현 후 공유
- 제출 방식: 인당 2회씩

2주차: 개인 작업 내용 병합 및 심화

- 제출 방식: 인당 2회 + 11:20PM 이후 남는 경우 선착순
- 최종 제출 모델 선정: 개인별 SOTA 모델 기준 가능한 조합 앙상블

10일	11일	12일	13일	14일	15일	16일
	탐색적 데이터 분석					
			서버 할당, 환경 설정			
			개인 실험 및 공유			
17일	18일	19일	20일	21일	22일	23일
개인 실험 및 공유		개인 작업 병합 및 앙상블				
				대회 종료		

협업 방식 (2/2)

Github Issue 생성

[FEAT] 베이스라인 코드 올리기 #2

Closed 7 tasks done GangBean opened this issue last week · 1 comment

GangBean commented last week · edited by ChangZero

- ✓ 베이스라인 코드 올리기
- ✓ 각자 모델 리팩토링하기
 - ✓ 장영 - NCF [FEAT] NCF #7
 - ✓ 주연 - CNNFM [FEAT] CNN_FM #3
 - ✓ 성룡 - WDN [FEAT] WDN #8
 - ✓ 진민 - FM [FEAT] FM #6
 - ✓ 상우 - FFM [FEAT] FFM #5

GangBean changed the title CNN_FM [FEAT] 베이스라인 코드 올리기 last week

하위 이슈 생성

FFM #5

Closed 2 tasks done GangBean opened this issue last week · 0 comments · Fixed by #28

GangBean commented last week · edited by sangwoonol

Background

- 베이스 코드에 포함된 FFM 모델의 구조를 이해하고 리팩토링한다.

To do

- ✓ FFM 모델 이해하기
- ✓ FFM 모델 리팩토링하기

작업 브랜치 생성

feat/5-ffmpeg

코드 리뷰 및 병합



PR 요청

feat: FFM 모델 구현 #5 #28

Merged GangBean merged 6 commits into main from feat/5-basic_FFM last week

Conversation 2 Commits 6 Checks 0 Files changed 28

sangwoonol commented last week

Overview

- FFM 모델을 베이스라인 코드 기준 리팩토링해 구현했습니다.

Change Log

-

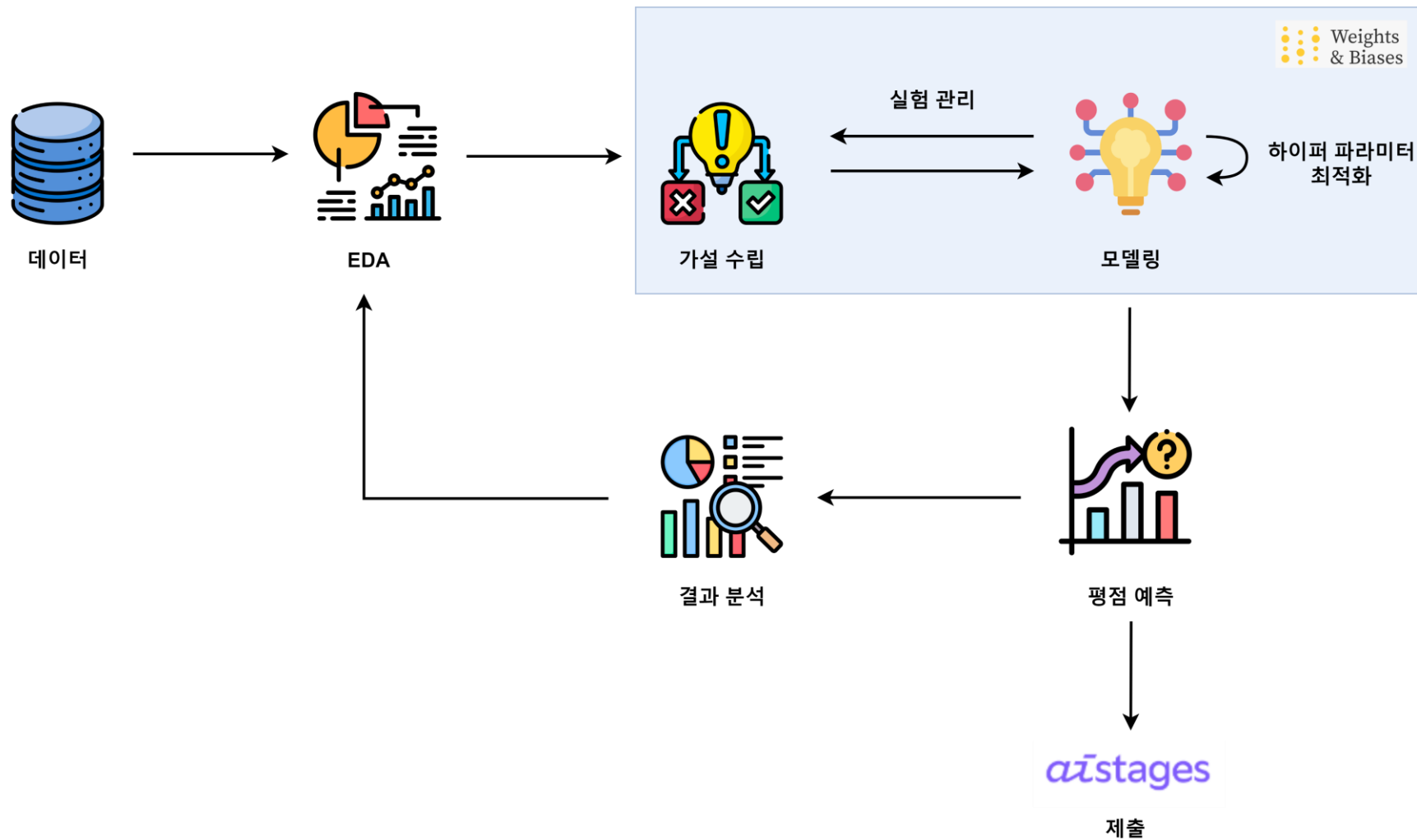
To Reviewer

- 28, 33라인: for 문 대신 리스트 컴프리헨션을 사용했습니다.
- 46, 50라인: FieldAwareFactorizationMachineModel에서 Features.Linear와 FieldAwareFactorizationMachine에 중복으로 존재하던, 입력 데이터 x에 월드의 시작 인덱스를 더해주는 작업을 한 번에 처리하도록 코드를 수정했습니다.
- 24라인: torch.nn.Embedding에 입력 자원으로 들어가는 sum(field_dims)을 따로 self.input_dim이라는 하나의 변수로 선언해서 사용했습니다.

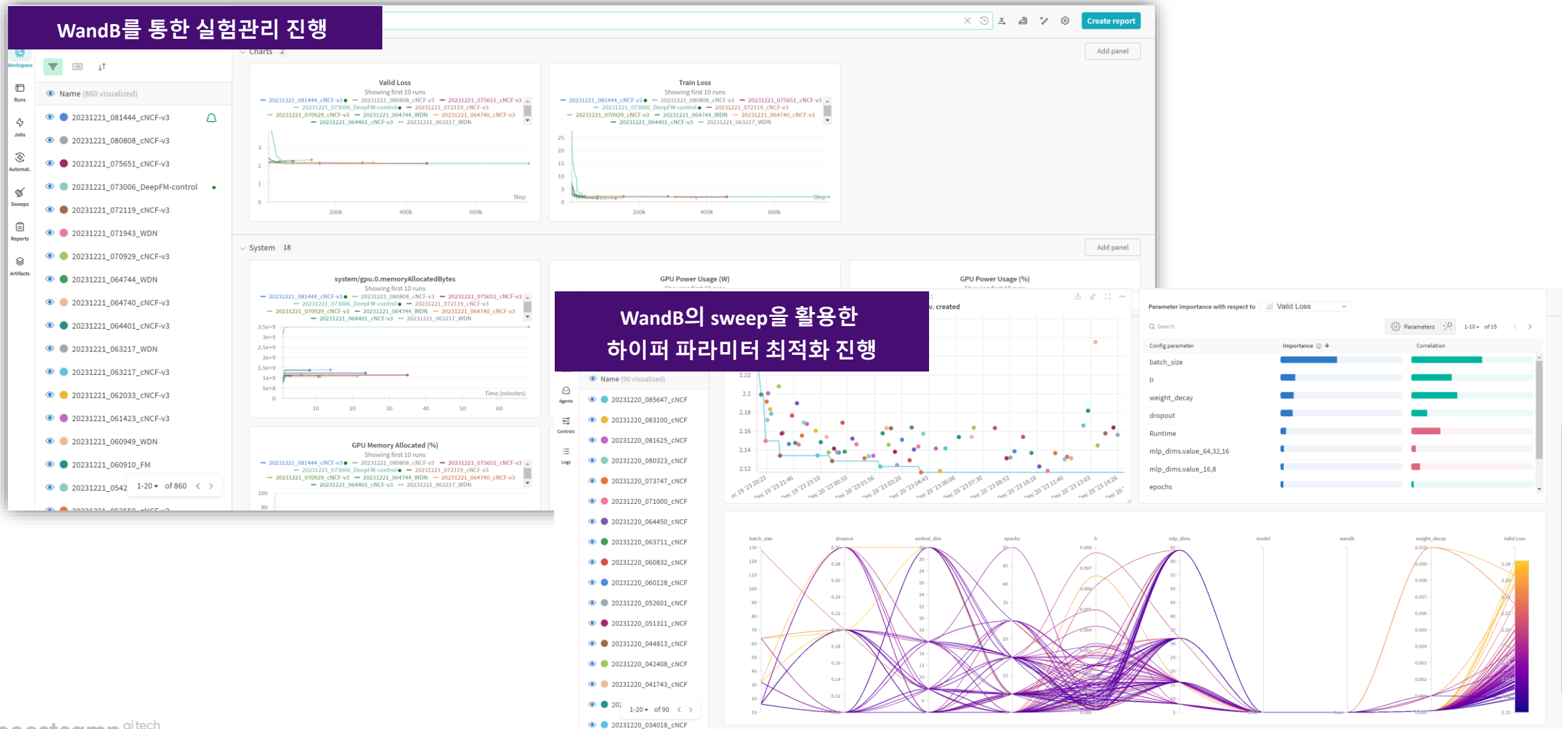
작업 진행



프로젝트 파이프라인



실험 관리 및 하이퍼파라미터 최적화



EDA

평점에는 사용자 ID, 책 ID, 평점 정보만 있다.

평점을 부여한 순서, 시간적 정보는 사용할 수 없겠다

서로 다른 사용자 약 7만 명, 책 15만 권, 학습 데이터 30만 개, 테스트 데이터 77만 개 (학습 데이터의 1/4)
콜드 유저가 test 유저의 31%, 콜드 아이템이 test 책의 38%

**ID 정보만 사용하는 전통적인 CF로는 테스트 데이터를 커버하기 어렵겠다.
이 때문에 베이스라인의 모델들이 대체로 context-aware 모델이었지 않았을까?
이에 반해 book의 특징이 많으니 컨텍스트 정보로 잘 활용해보아야겠다**

책의 카테고리 데이터를 들여다보니, 빈도가 50개에 못미치는 범주가 여럿 있다

모델 성능 향상에 도움이 안될 것 같다. 빈도 수가 높은 범주만 남도록 전처리가 필요하겠다

일부 특징의 결측 비율이 매우 높다.

결측을 어떻게 처리하느냐에 따라 모델의 성능이 달라질 수도 있겠다

...

아이디어와 가설 설정 - 데이터 측면

입력 특징의 많아지면 성능이 항상 향상할까?

- H1: user의 location에서 city 와 state를 제거하면 성능이 떨어질까?

연속형 변수를 어떻게 카테고리화하는 것이 성능 향상에 더 올라갈까?

- H2: age에서 빈도가 적게 발생하는 범주를 하나의 범주로 처리하면 성능이 올라갈까?
- H3: user age기준으로 각 bin의 frequency를 평준화하면 성능이 올라갈까?
- H4: 10년 단위로 binning하면 성능이 떨어질까? (70대 이후는 합침)

결측치/이상치의 처리 방법이 모델 성능에 영향을 줄까?

- H5: category 데이터의 결측을 na 범주를 생성하여 할당하는 방법과 최빈값으로 대체하는 것 중 어떤 것이 더 효과적일까?
- H6: 빈도가 너무 낮은 카테고리의 사용이 모델 학습에 방해가 되지는 않을까?
- H7: 빈도가 50 이하인 category를 결측으로 대체하였을 때 성능이 향상될까?
- H8: category 재분류: 상위 카테고리로 기존 카테고리들을 묶어서 재분류하면 성능이 올라갈까?

추가로 사용 가능한 데이터는 없을까?

...

실험 결과 - 데이터 측면 (1/2)

입력 특징의 개수와 성능 향상이 항상 비례할까? **X** 연속형 변수를 어떻게 카테고리화하는 것이 성능 향상에 더 좋을까? **별 차이 없다**

H1: user의 location에서 city와 state를 제거하면 성능이 떨어질까? **X**



H2: age에서 빈도가 적게 발생하는 범주를 하나의 범주로 처리하면 성능이 올라갈까? **X**



H3: age 기준으로 각 label의 frequency를 평준화하면 성능이 올라갈까? **X**



H4: 10년 단위로 binning하면 성능이 올라갈까? **X**



실험 결과 - 데이터 측면 (2/2)

결측치/이상치의 처리 방법이 모델 성능에 영향을 줄까? **준다**

H5: category 데이터의 결측을 na 범주를 생성하여 할당하는
방법과 최빈값으로 대체하는 것 중 어떤 것이 더 효과적일까? **범주 생성**



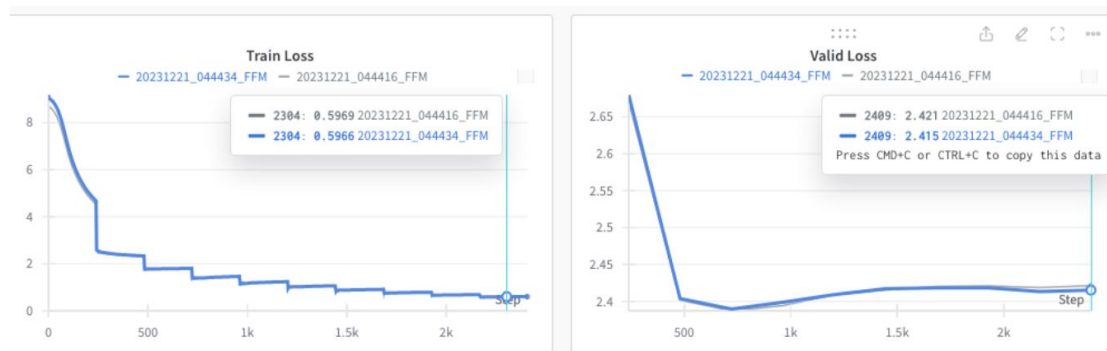
H7: 빈도가 50 이하인 category를 결측으로 대체하였을 때 성능이
향상될까? **○**



H6: 빈도가 너무 낮은 카테고리의 사용이 모델 학습에 방해가 되지는 않을까? **×**



H8: category 재분류: 상위 카테고리으로 기존 카테고리들을 묶어서 재분류하면 성능이
올라갈까? **○**



아이디어와 가설 설정 - 모델 측면

Factorization Machines: 특성 간의 상호작용을 학습할 수 있지만 선형적으로 계산되기 때문에 조금은 한계점이 있을 수 있다고 생각

Field-aware Factorization Machines: 특성을 필드로 구분하고, 필드 간 상호작용을 잠재 벡터에 모델링하므로, FM 보다 특성 간 상호작용을 더 섬세하게 모델링하여, 주로 카테고리 데이터에서 강점이 있을 듯

DeepFM: FM으로 특성 간 선형 상호작용을, DNN 파트를 통해 비선형 상호작용을 모델링. 복잡한 상호작용을 가진다면 적합할 것.

DeepFFM: FFM의 섬세하다는 강점과 DNN의 비선형 상호작용 모델링을 결합하면, DeepFM이 좋은 성능을 내는 환경에서 높은 성능을 낼 수 있을지도.

CNN-FM: 취향과 평가는 다르니까 책의 평가를 예측하는 데에 책 표지는 큰 영향이 없을 것으로 보임.


NCF: NCF는 컨텍스트 없이 user, item의 상호작용만을 학습하기 때문에, 콜드 유저나 아이템에 대해서는 높은 성능을 내기 어렵지 않을까?

cNCF: 컨텍스트 정보를 결합하여 학습한다면, NCF의 성능을 보완할 수 있지 않을까?

WDN: Wide 와 deep component의 반영 비율을 조절하면 일반화 성능 관점에서 영향이 있을까? Deep component의 비율을 높이면 test 데이터와 loss 차이가 줄어들까?

CatBoost: 이미지, 텍스트를 제외한 다른 정보들은 모두 1차원으로 구성된 tabular 데이터이므로, 정형 데이터, 특히 카테고리 데이터에서 높은 성능과 안정성을 보이는 Boosting 모델 여기서도 높은 성능을 낼 수 있지 않을까

실험 결과 - FM series

모델명	Train	valid	public	private
FM	2.047	2.186	2.1889	2.1868
FFM 	1.9417	2.1637	2.1693	2.1676
DeepFFM	2.188	2.218	2.2077	2.2098
DeepFM	2.126	2.215	2.1797	2.1824
DeepFM-summary	2.06	2.123	2.1819	2.1781

FFM --> DeepFFM

- 표현력을 높이기 위해 기존 FFM모델에 MLP 추가
- 성능향상이 크게 없고, 특정 조건에서는 오히려 성능이 떨어짐
- FFM과 MLP가 같은 임베딩을 공유해서 이것이 FFM의 학습을 방해하기 때문이라고 판단

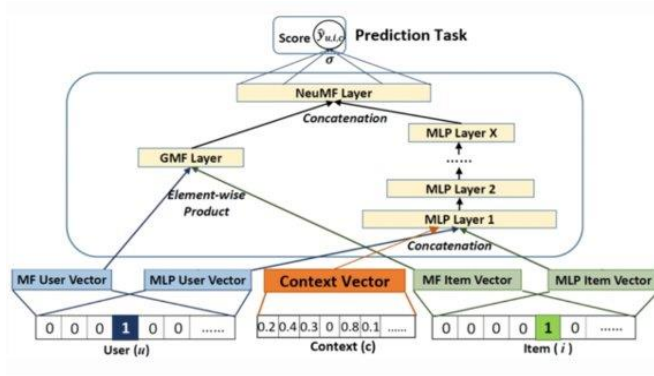
DeepFM vs. DeepFM-summary

- summary를 BERT embedding한 뒤, DNN term의 입력에 추가함
- Val loss 는 개선되었으나, public score는 아님. 근데 private는 개선됨...(?)

실험 결과 - NCF

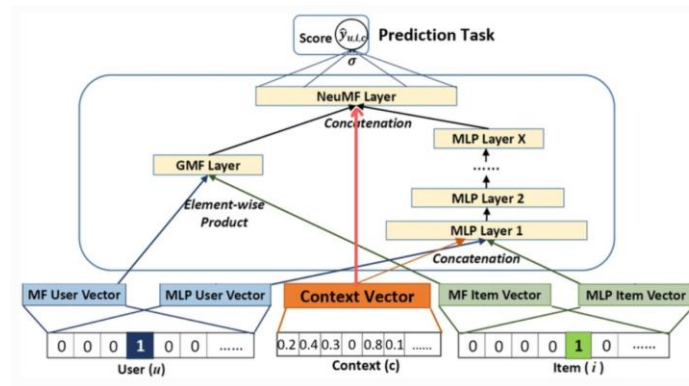
기존 NCF는 데이터가 가지는 희소성으로 인해 과적합이 발생! -> context 정보를 활용해보자!!

cNCF



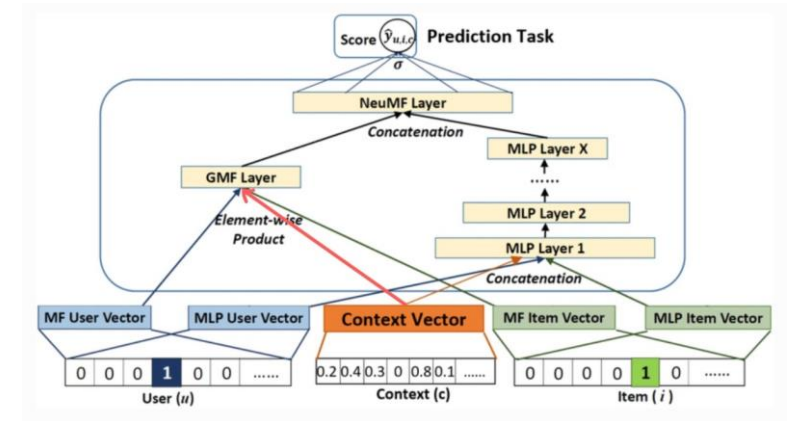
Context 정보를 활용하기 위해 MLP에 입력으로 추가

cNCF-v2



Context 정보를 MLP에 입력으로 주고 MLP의 출력과 GMF, Context latent factor를 concat

cNCF-v3



Context 정보를 GMF와 MLP의 입력으로 활용

모델명	Train	valid	public	private
NCF	1.697	2.29	2.2226	2.23
cNCF	1.821	2.135	2.1743	2.1741
cNCF-v2	2.037	2.122	2.1845	2.1809
cNCF-v3	2.013	2.117	-	-

Summary

- Context 정보를 활용하면 기존 NCF 보다 성능이 보다 더 향상되는 것을 확인했다.
- Context 정보를 많이 활용할 수록 Train과 Valid의 간격이 좁아지는 것을 확인했다.

앙상블 학습

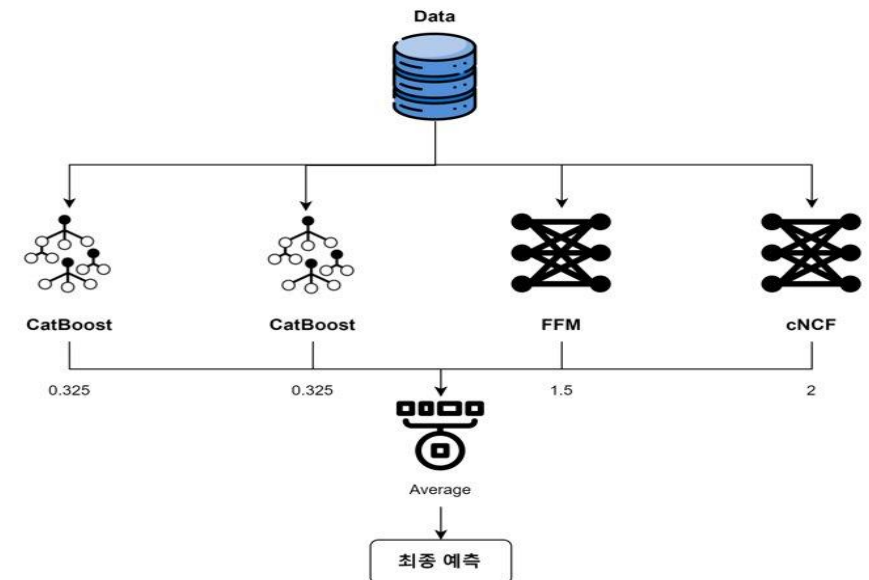
앙상블	public	private
CatBoost(0.5) + cNCF(0.5)	2.1379	2.1348
CatBoost(0.5) + FFM(0.5)	2.1360	2.1309
CatBoost(0.6) + FFM(0.2) + cNCF(0.2)	2.1304	2.1256
FFM(0.5) + cNCF(0.5)	2.1588	2.1593
CatBoost(0.8) + cNCF(0.2)	2.1345	2.1286
CatBoost(0.7)+FFM(0.15)+cNCF(0.15)	2.1316	2.1261
CatBoost(0.7)+cNCF(0.3)	2.1331	2.1299
CatBoost(0.3)+CatBoost(0.3)+cNCF(0.2)+ FFM(0.2)	2.1297	2.1251
CatBoost(0.325)+ CatBoost(0.325) + cNCF(0.15) + FFM(0.2)	2.1300	2.1249



FFM+cNCF vs. DeepFM

- 공통점: low-order, high-order, context
- 차이점: embedding 공유, GMF 차이
- DeepFM 보다는 성능이 좋고, FFM, cNCF만 한 것보다도 좋음
- NCF, WDN context 추가 시 성능 좋아짐 둘다 high-order

- 5:5 loss > 8:2 loss
- CatBoost 비율 크게 더 좋음



실험 결과 - 그 외


WDN: context 정보로 입력을 추가하고, 각 component의 weight을 조절해 봤습니다.

모델명	Train	Valid	Public
WDN_non_context	1.121	2.391	-
WDN_context_w5:d5	1.424	2.291	-
WDN_context_w3:d7	1.459	2.290	-
WDN_context_w7:d3	1.425	2.285	-

Summary

- Context 정보를 활용하면 WDN에서도 성능이 향상됨
- Wide component 와 Deep component의 weight을 조절하는 것은 유의미한 차이를 발견하지 못함

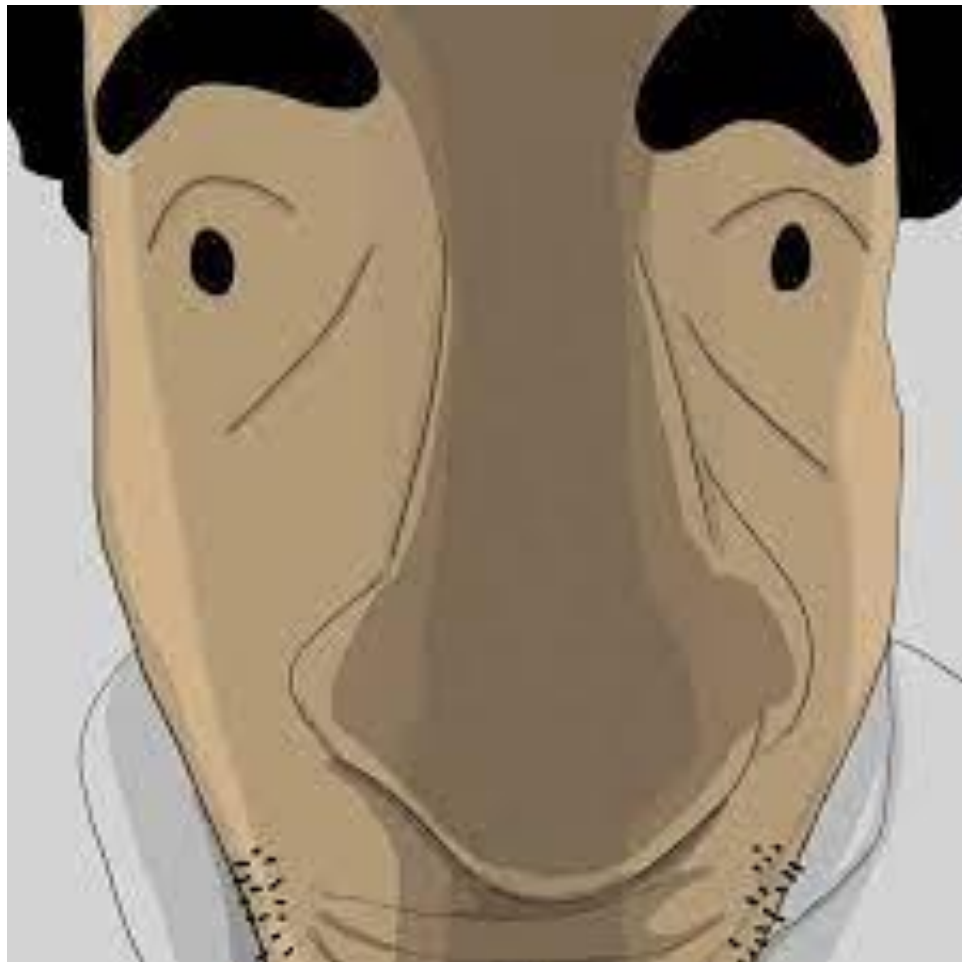
GBDT Model

모델명	Train	Valid	Public
XGBoost	-	2.27	-
LightGBM	-	2.31	-
CatBoost 	-	2.14	-

실험 결과 해석

- **Ordered TS(Target Statistics), ordered boosting** 이 많은 카테고리 데이터를 다루는 데 유리하지 않았을까?
 - 카테고리 데이터에 유용한 FFM도 비교적 높은 성능을 냈으므로
 - 이번 대회에서는 CatBoost의 성능이 가장 좋았다. XGBoost, LightGBM의 성능은 그다지 좋지 않았음
- 데이터의 규모가 매우 큰 편이 아니라 내재된 복잡한 상호작용이 많지 않았던 걸까 ?
 - FM vs. DeepFM, FFM vs. DeepFFM 에서 DNN을 추가해도 성능이 드라마틱하게 좋아지지 않음
- **Context** 의 여부는 모델의 성능에 큰 영향을 줌 ?
 - CatBoost, NCF, WDN 모두 ID 정보만 사용할 때보다 context를 같이 학습할 때 성능이 확실하게 향상됨
- **Summary**는 일반화 성능 향상에 도움이 된다 ? 안된다 ?
 - summary 정보를 추가하였더니, train/val 성능은 크게 향상되었으나 public score, private는 약간 개선됨
 - summary train/val score와 public score의 gap이 커졌으므로, 일반화 성능에는 긍정적인 영향이 아니라고 할 수도 있을까?
- 딥러닝 모델은 하이퍼파라미터에 민감하다 ?
 - 특히, Weight decay, batch_size, learning rate에 따라 학습 경향이 크게 바뀐다.

마무으으으으으리



다음엔 꼭 1등 할꺼야....


부록 - 어려웠던 점 및 아쉬운 점

- 실행 측면
 - 시도해보고 싶은 것들은 많은데 이 시도가 적절한 시도일지 판단하기 어려웠다.
 - 구현 속도가 아이디어이션 속도를 못 따라가 많은 실험을 더 못해본 것이 아쉬웠다.
 - Feature Engineering → Modeling → Hyper Parameter Tuning의 순서로 계획을 했지만 중구난방으로 진행했다.
 - 신뢰할 만한 리소스(공식 문서, 논문 등)를 기반으로 논리를 전개하고 코드로 구현해 검증하는 과정을 계획했으나 잘 이뤄지지 않았다.
 - 각 실험 결과 해석시에 명확한 이론적 근거를 만들지 못했다.
 - GBDT 모델들은 Hyper-Parameter가 너무 많아 최적화를 제대로 하지 못한 것이 아쉬웠다.
- 실험 관리 측면
 - 실험 기록을 꼼꼼히 하지 못했다.
 - EDA 후 각 특징을 정리하지 못했다.
- 대회 접근 측면
 - train, valid loss 는 좋아서 제출했지만 public score 낮은 경우 있어 어떤 결과를 제출해야 좋을지 몰랐었다.
 - test 데이터에 대해서는 횟수가 제한된 제출을 통해서만 확인할 수 있어 가설에 대한 확실한 검증이 어려웠다.
- 기타
 - 최대한 재사용 가능한 코드 구현을 목표로 했으나 잘 이뤄지지 않았다.

다음에는 초기 작업 환경 구성(코드, 실험관리 툴 등)을 구조적이고 구체적인 계획을 해야겠다.

부록 - 잘한 점

- Git과 Github을 활용해 작업을 진행했다.
 - 문제나 이슈 발생시 github issue 로 기록하고 공유하며 빠르게 해결했다.
- 초기 프로젝트 설정과 구현 방식 논의를 빠르게 마쳐 원활한 협업이 가능했다.
- 논문을 참고하면서 Hyper-Parameter 설정을 했다.
- 코드 리뷰와 열띤 대화를 통해 학습했으나 구현해본 모델만큼은 이해도가 매우 크게 향상되었다.
- WandB를 통한 학습 과정 모니터링, 실험 관리와 하이퍼 파라미터 튜닝을 적용했다.
- 각자 시도한 방법, 궁금한점, 인사이트 등 활발한 공유가 이뤄졌다.
- 성능을 보고 아쉬웠던 것도 사실이지만, 각자가 많은 시도를 해봤다는 점에서 나름의 성취가..

순위	팀 이름	팀 멤버	RMSE ⚡	제출 횟수	최종 제출
4 (-)	RecSys_02조		2.1297	78	20h



넬슨 만델라

나의 성공으로 나를 판단하지 말고, 내가 얼마나 넘어지고 또 다시 일어섰는지를 통해 나를 판단하라.

부록 - 궁금했던 것

- 전처리, 모델 종류, 하이퍼파라미터의 각 단계에서 변화를 줄 수 있는 부분이 매우 많은데, 모든 기능을 다 모듈화하여 하나씩 실험을 해봐야하는 것인지, 실험 관리를 어떻게 할 것인지?
- 함께 대회를 진행할 때 코드 관리는 어떻게 하는 게 좋은지
- 실험 결과를 어떻게 모델 관점에서 해석할지 (아래와 같이 접근하면 될지)
 - 모델의 특징이 되는 것의 이론적 배경을 살핀다
 - 이론적 배경을 토대로 모델의 장단점을 유추한다
 - 유추한 모델의 장단점에 관련된 실험 결과를 찾아보거나, 없다면 직접 실험을 통해 검증한다
 - 모델의 장단점을 정리한다
- 우리 나름대로 정리한 결론(?)이 적절한지