DKT 랩업 리포트

▼ **목차**

1. 프로젝트 개요

Level2 Deep Knowledge Tracing 대회에 참여하여 다양한 엔지니어링 및 머신러닝 방법론으로 실험한 결과와 배운 점을 기록함.

대회 주제

Deep Knowledge Tracing(DKT)는 시간에 따른 유저의 지식 상태를 모델링하는 교육 AI의 한 분야. 이번 대회는 유저의 지식 상태를 예측하는 것에서 나아가, **마지막에 등장하는 문제를 맞출지 예측하는 태스크.**

대회 성능 평가 지표

본 대회는 이진분류 문제이기 때문에 Accuracy 하나만으로 평가를 하기에는 잘못된 평가가 되기 때문에 AUROC(Area Under the ROC curve)와 Accuracy 함께 사용함.

데이터

데이터 명세

컬럼명	설명
userID	사용자의 고유번호
assessmentItemID	문항의 고유번호
testId	시험지의 고유번호
answerCode	사용자가 해당 문항을 맞았는지 여부
Timestamp	해당문항을 풀기 시작한 시점
KnowledgeTag	문항 당 하나씩 배정되는 태그

2. 팀 구성 및 역할

팀 한줄 소개

- 서신이삼조
- 트러블메이커

팀 목표

- 개인적 목표 추구 및 원활한 소통을 통한 팀 전체의 성장

팀 문화

- **페어 프로그래밍**: 2명 단위로 태스크를 지정하여 태스크의 완성도를 높이려고 노력함
- **매일 진행 상황 공유**: 어디까지 했고, 어디에서 어려움을 겪고 있는지 투명하게 공유함
- **게더타운에서 코어타임 동안은 상주하기**: 어려움이 있을 때 동료 찬스!
- **제출 규칙**: 매일 인당 1회/11시 이후 자율적으로 제출
- **오프라인 미팅**: 매 주 3명 이상 참여 시 진행

역할

캠퍼	역할
서동은	• ML modeling, Hyper parameter tuning • Saint 기반 모델링
신상우	• LastQuery, LightGBM 모델 베이스라인 구축, HPO • LastQuery 기반 모델링
이주연	• ML modeling, Ensemble, OOF, CV • LSTM, LastQuery, LGBM 기반 모델링
이현규	• Feature Engineering, EDA • Sasrec 기반 모델링
이현주	• ML modeling, Hyper parameter tuning • LightGCN 기반 모델링
조성홍	• Feature Engineering • LGBM HPO

3. 프로젝트 수행 절차 및 방법

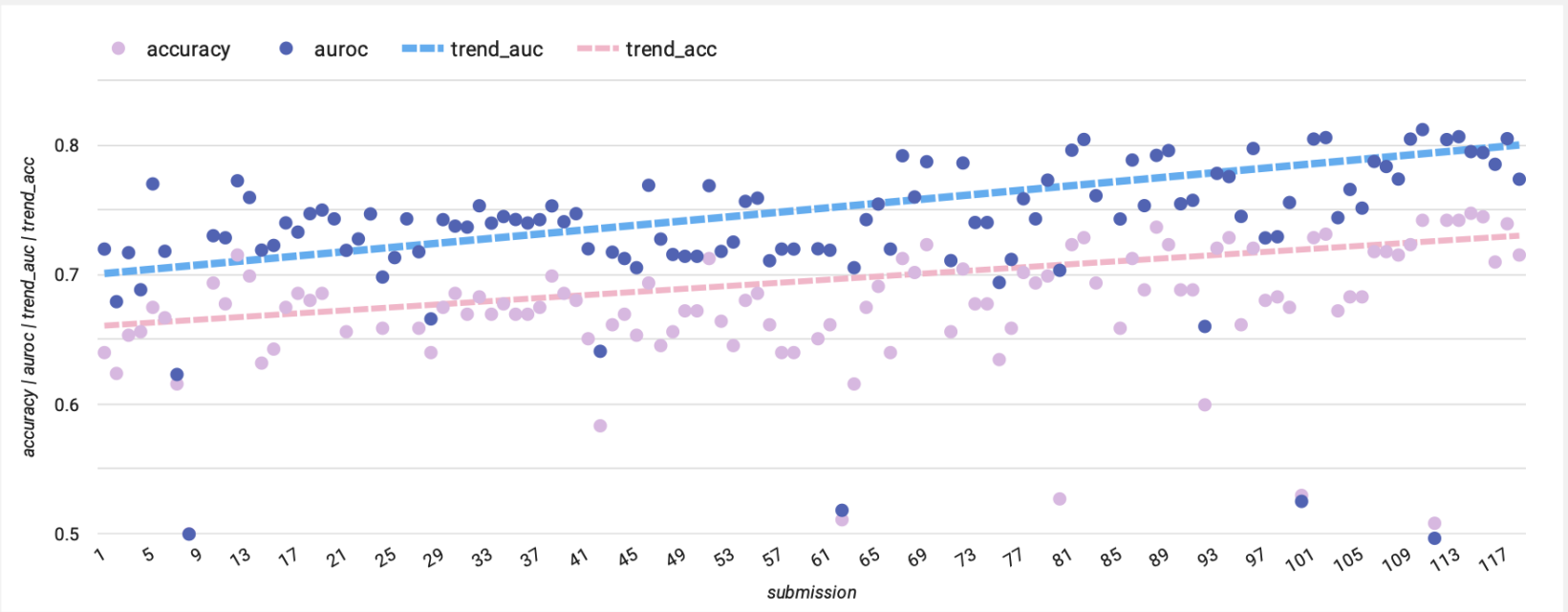
프로젝트 타임라인

제출기록

RECSYS-01 DKT CONTEST REPORT

	Total	Charlie	Judy	Hyelia	Tatlock	Eddie	Noel
제출 횟수	118	17	39	3	10	8	35
max AUC (private)	81.2%	80.5%	81.2%	77.4%	80.4%	80.6%	80.6%

제출 회차별 AUROC, Accuracy 변화



4. 사용한 모델들

1) Feature Engineering

개요 및 주안점

본 대회는 유저와 아이템의 정보를 토대로 유저/아이템 간의 협응 관계를 찾는 것이 아닌, 아이템 시퀀스를 기준으로 다음 시퀀스를 분류하는 이진 분류 문제였기에 **아이템 정보 및 시퀀스 정보의 추출** 에 집중해 새로운 feature를 추가함.

Experiments

Exp1: 문제의 특징 정보-제한 시간

배경 및 기대효과 하나의 test에 여러개의 assessmentItem이 할당된 형태. 하나의 시험에 할당되는 **제한 시간**이 존재하지 않을까 생각. 각 문제를 풀기 시작한 시작시간 정보(Timestamp)가 존재하기에, 제한 시간 근처에 풀기 시작한 문제는 정답률이 떨어지지 않을까 생각. 따라서, 정답 여부를 구분 짓는 중요 정보로 활용될 것이라 가정. 각 테스트별로 전체 유저의 **‘(마지막 문제 풀이 시점) - (최초 문제 풀이 시점) 의 중앙값’**을 제한 시간으로 가정.

결과 및 해석 성능 향상이 있었으나, 큰 폭의 향상은 아니었음. 가정했었던 각 **테스트별 제한 시간 근처 item의 정답률**을 테스트 평균 정답률과 비교해보면 해석이 가능할 것으로 기대.

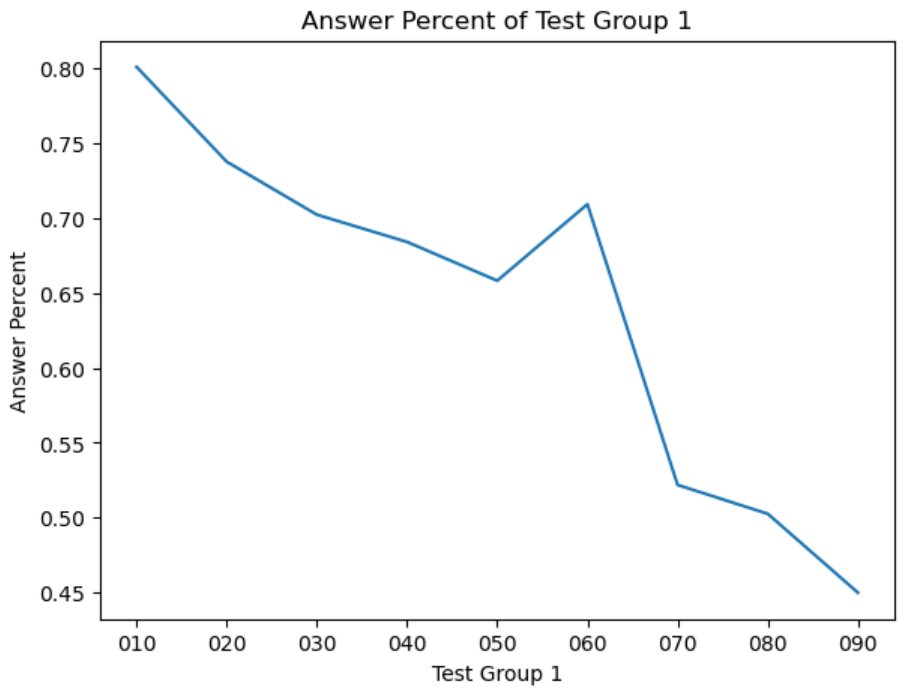
Type	train AUC	valid AUC	etc
Numeric	0.7509	**0.7420**	default valid auc 0.7411

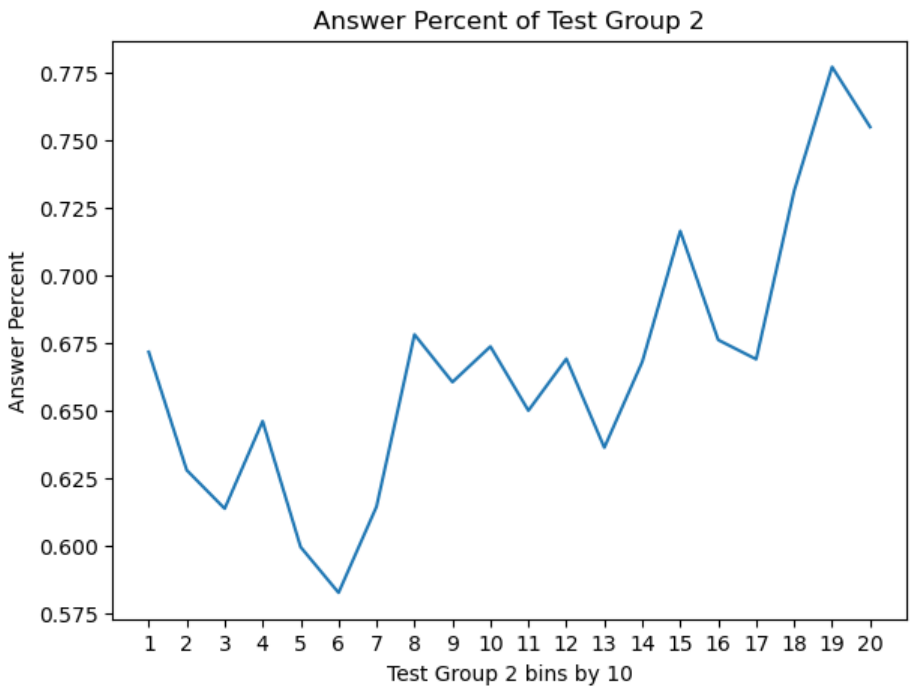
Exp2: 문제의 특징 정보-TestID 분리

배경 및 기대효과 TestID는 순차적인 숫자인 userID 와 달리 (한 자리의 문자 + 9자리의 숫자) 로 구조화된 문자열 값. 구조화된 문자열 값이라는 점에서 구조적인 의미를 가질 것이라 가정. 문자는 모두 ‘A’ 로 동일, 9자리 숫자를 3자리씩 나눴을 때 중간 세자리 값은 모두 ‘000’ 으로 동

일. 첫 번째 3자리 숫자를 test_group_one, 마지막 3자리 숫자를 test_group_two 의 새로운 feature 생성.

결과 및 해석 Numeric/Categorical 두 가지 형태로 추가했을 때 모두 성능 향상됨. 정확히 테스트의 어떤 정보를 담고 있는지까진 해당 실험만으론 추론이 불가능하지만, EDA를 통해 추가한 feature 분류별 통계 정보를 확인하면 추론이 가능하지 않을까 예상. 실제로 test_group_one 별 정답률을 확인해봤을 때, 숫자가 큰 그룹일수록 정답률이 낮아짐을 확인. test_group_two 에서는 test_group_one 만큼의 큰 경향성은 확인할 수 없었지만 10단위로 잘랐을 때, 정답률에 어느정도 차이는 존재함을 확인.



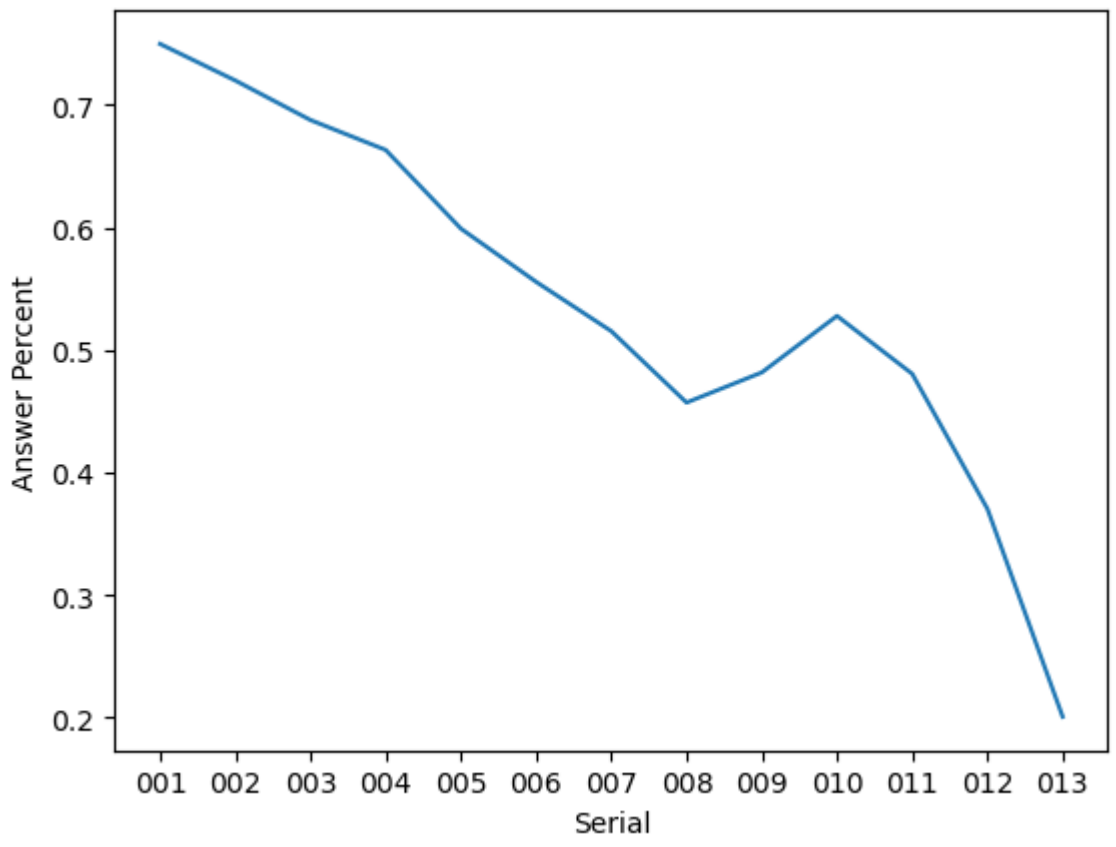


Type	train AUC	valid AUC	etc
Numeric	0.7485	0.7438	default valid auc 0.7411
Categorical(dim 115)	0.7722	**0.7464**	default valid auc 0.7411

Exp3: 문제를 푼 순서 정보-test내 assessment 순번

배경 및 기대효과 assessmentItemId는 testId와 동일한 구조로 되어 있음: A + 숫자 9자리. 다만, (testId에서 중간 숫자 3자리를 제외한 축약된 값) + (순차적인 3자리의 숫자) 의 구조. 그래서 각 문제의 순차적인 숫자가 해당 테스트에서 제공되는 과제의 순서라고 가정하고, 해당 순서에 따라 추론할 수 있는 정보가 있지 않을까란 생각으로 분리해 봄.

결과 및 해석 순번이 높아질수록 정답률이 유의미하게 떨어지는 것을 확인할 수 있었고, 실제 feature 추가 시 모델 성능도 향상됨을 확인함.



Type	train AUC	valid AUC	etc
Categorical(dim 100)	0.7692	**0.7507**	default valid auc 0.7411

2) Sequence Modeling

개요 및 주안점

대회 데이터가 시계열 데이터이기 때문에 Sequence 정보를 반영할 수 있는 LSTM, Transformer와 같은 Sequence 기반의 모델을 사용함. 이들은 주로 자연어 처리 (NLP) 작업을 위해 설계된 deep learning 아키텍처이지만, 같은 시퀀스 데이터를 처리한다는 점에서 dkt 문제를 해결하는데 활용됨

(1) SASRec

시계열 데이터를 효과적으로 이용하기 위해 transformer와 lstm을 활용하였음. 이전 정보와 현재 정보의 연관성을 잘 나타낼 수 있는 형태를 찾고자 함.

model	valid AUC	public AUC
sasreclstm	0.7960	0.8153

(2) LastQueryTrmLSTM

dkt와 비슷한 Ruiid 대회에서 1등 솔루션으로 QK 행렬곱 연산의 복잡도를 O(L^2) 에서 O(L)로 개선한 모델임. 이로 인해 max_seq_len을 늘려서 한 번에 많은 시퀀스를 학습할 수 있기 때문에 확장성이 높아 dkt 문제에 적용하고자 함. 학습 속도 자체도 빠르기 때문에 다양한 실험을 빠르게 할 수 있다는 장점이 있음

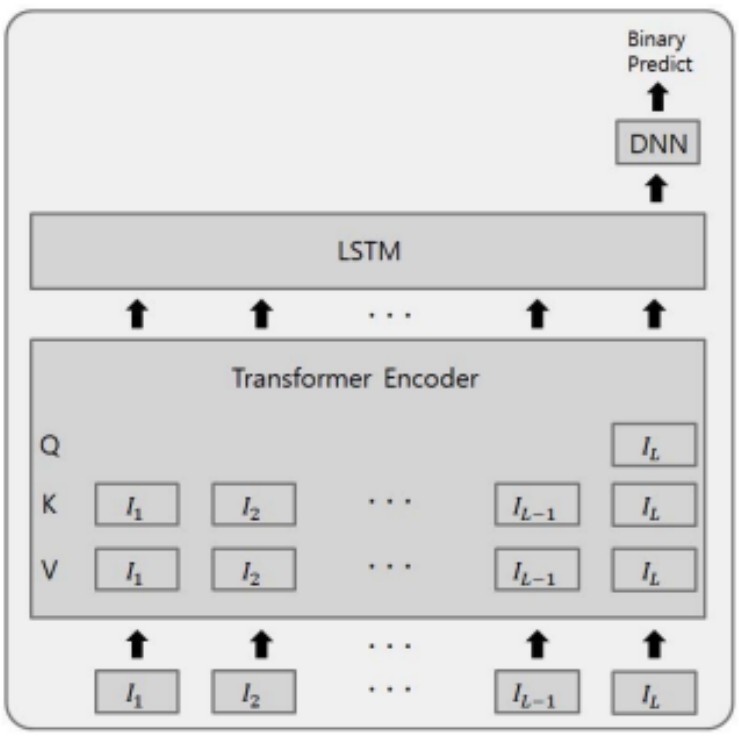


Figure 1. Model Structure

모델 구조 (논문 참고: https://arxiv.org/abs/2102.05038)

Exp1. LastQueryTrmLSTM max_seq_len 증강

유저가 과거에 푼 문제 정보를 많이 볼수록 예측을 더 잘할 것으로 판단하여 입력 시퀀스의 최대 길이(max_seq_len)를 바꿔보는 실험을 진행함

public score가 떨어져서 max_seq_len이 dkt 데이터에서는 의미가 없다고 생각했는데, private 결과를 보니 의미가 있었음. 모델이 한 번에 볼 수 있는 데이터의 양이 늘어나서 성능이 좋아진걸로 해석됨

max_seq_len	valid AUC	public AUC	final AUC
20	0.7376	**0.7424**	0.7451
512	0.7391	0.7366	**0.7519**

Exp2. 문제를 푼 시간 feature 추가 (Elapsed time)

riid! 대회에서 의미있게 작용한 **문제 풀이 소요 시간** feature를 추가함(3시간 이상 걸린 문제는 3시간으로). 문제 풀이 소요 시간으로 문제 난이도를 유추할 수 있다고 판단해 유의미한 feature가 될 것으로 기대함. public score가 얼마 오르지 않아서 의미가 없다고 생각했는데, private에서 0.0036점 상승함. 문제 풀이 시간으로 난이도 정보를 뽑아냈고, 3시간 기준과 결측값을 더 신경 썼다면 더 좋은 성능을 낼 수 있었을 것 같음

Elapsed time	valid AUC	public AUC	final AUC
x	0.7376	0.7424	0.7451
o	0.7394	0.7429	**0.7487**

Exp3. Vanilla LSTM vs. LastQueryTrmLSTM

동일한 피처 셋(22개)으로 두 모델의 성능을 비교해봄. 두 모델은 LSTM 구조는 같지만, 트랜스포머 인코더 블록의 포함 여부에서 차이남. public AUC를 통해 LSTM에 비해 트랜스포머 인코더가 타겟 예측에 중요한 피처를 찾아낸다고 추론하였으나, final AUC를 통해 LSTM이 일반화 성능이 더 좋았음을 알게 됨

Model	valid AUC	public AUC	final AUC
LSTM	0.8237	0.7873	**0.8251**
LastQueryTrmLSTM	0.8316	**0.7957**	0.8238

3) CF method (SVD, Graph Model)

DKT 문제를 추천 방식으로 접근하여 학습자의 현재 성취도를 바탕으로 새로운 문제를 맞출 수 있을지, 없을지 예측함. **피처 사용하지 않음**

1. DKT를 Matrix Factorization (MF) 기반으로 접근함 그 중 SVD를 사용함
2. lightgcn을 사용해서 userID와 assessmentID간의 그래프를 모델링하고 collaborative signal을 학습함

Experiments

Exp1: SVD

배경 및 기대효과 DKT 문제를 전통적인 협업 필터링 방식을 활용하여 풀었을 때, 특징 정보를 활용하지 못하고 순서를 고려하지 못해 시퀀셜 모델링에 버금가는 성능에 달하지 못할 것으로 예상함

결과 및 해석 시퀀셜 모델링 대비 낮은 성능을 보임

model	valid AUC	public AUC	비고
SVD	0.7671	0.7424	32 차원

Exp2: LightGCN

배경 및 기대효과 이웃 노드 간의 정보를 현재 노드에 반영하는 방식으로 고차원 임베딩을 학습하여 SVD보다 더 높은 성능을 낼 것으로 기대함

결과 및 해석 SVD보다 좋은 성능을 보임

model	public AUC	final AUC	비고
LightGCN	0.7737	0.8136	adamW, cosine_annealing

4) Ensemble

앙상블 배경

No free lunch theorem 모든 데이터셋이나 태스크에서 항상 우월한 머신 러닝 모델은 없다는 이론. 모든 알고리즘은 특정 종류의 문제에 대해서는 효과적일 수 있지만, 모든 종류의 문제에 대해 똑같이 잘 작동하는 완벽한 알고리즘이 없다는 뜻. [참고1] [참고2]

따라서 DKT 문제에 대해 최고 성능을 내는 개별 모델이 있다고 하더라고, 비교적 높은 성능을 내는 모델을 적절히 결합하였을 때 이에 견줄 수 있는 성능을 낼 수 있다고 가정하고, 앙상블 방법 중 부스팅과 배깅 방식을 활용함

앙상블은 여러 개별 모델의 예측을 결합하여 하나의 강력하고 안정적인 예측을 만들어내는 효과적인 기법임. 이는 단일 모델을 사용할 때 발생할 수 있는 과적합 문제를 해결할 수 있고, 일반화 능력 또한 좋음. 즉, 여러 가지 모델을 결합하면 다양성을 확보하여 개별 모델의 약점을 서로 상쇄할 수 있음

Weighted Average

개별 모델들의 결과값을 이용하여, 최종값을 선택하는 앙상블 기법 중 Weight Average (가중 평균) 을 사용함. 해당 프로젝트에서는 각 모델의 성능을 나타내는 auc 값을 참고하여, auc가 더 높은, 즉, error 가 더 적은 모델에게 weight 을 더 많이 주고 이에 대한 평균을 최종 예측값으로 사용함

Exp: Catboost + LastQuery + LGBM + Sarec + LightGCN

배경 및 기대효과 각 모델들 중 sota 인 실험 결과를 앙상블 하면 더 좋은 성능을 낼 것으로 기대함. 또한 sota 가 아닌, 상위 n 개의 모델을 seed 앙상블 한 모델을 앙상블 하면 더 좋은 성능을 낼 것으로 기대함

결과 및 해석 5개의 모델을 앙상블했을 때 성능이 가장 좋았음. seed 앙상블은 sota 를 앙상블을 한 것보단 성능이 좋지 않았고, weight 값을 조절했을 때 성능이 더 좋아진 것으로 나타남

Model	Weight	final AUC	final ACC	public AUC	public ACC
Catboost_sota + LastQuery_sota + LGBM_sota + Sasrec_sota + LightGCN_sota	0.2, 0.2, 0.2, 0.2, 0.2	0.8451	0.7849	0.8119	0.7419
Catboost_seed + LastQuery_seed + LGBM_seed + Sasrec_seed+ LightGCN_seed	0.1, 0.125, 0.15, 0.125, 0.5	0.8449	0.7769	0.8111	0.7285
Catboost_seed + LastQuery_seed + LGBM_seed + Sasrec_seed+ LightGCN_seed	0.2, 0.2. 0.2. 0.2. 0.2. 0.2	0.8405	0.7742	0.8124	0.7392
LightGCN (5)	0.17, 0.17, 0.17, 0.17, 0.33	0.8433	0.7688	0.8124	0.7419

Boosting

부스팅은 high bias, low variance를 가진 약한 학습기를 순차적으로 조합하여, 각 학습기가 이전 학습기의 bias를 보완하도록 설계하여 높은 예측 성능을 내는 앙상블 기법임. 그 중 Gradient Boosting 방식을 발전시켜 등장한 LightGBM, XGBoost, CatBoost가 강력한 성능을 발휘하는 것으로 알려짐에 따라 활용함

본 대회의 데이터는 시퀀스 형태이지만, 부스팅은 general predictor로, 입력 데이터를 (배치, 시퀀스길이, 특징개수)가 아닌, (배치, 특징개수)의 shape으로 가공해야 했음. 우리 조의 경우, 시퀀스의 각 요소를 개별 데이터로 입/출력하는 **Full 방식**과, 마지막 문제를 맞추는 task에 맞게 마지막 문제만 입/출력으로 활용하는 **Last element 방식**을 활용함

Experiments

Exp1: XGBoost vs. LightGBM vs. CatBoost w/ 8 features

배경 및 기대효과 8개의 기본적인 피처만 활용하고 하이퍼파라미터를 기본 값으로 세팅하였을 때, CatBoost의 valid AUC가 가장 높았음.

결과 및 해석 CatBoost는 하이퍼파라미터 튜닝 없이도 높은 성능을 내는 것으로 알려진 모델이며, tag, test, item이 다범주형 변수라서 높은 것으로 판단. 다만, 이후 피처 엔지니어링을 통해 생성한 대부분의 특징은 연속형이었기에, LightGBM과 CatBoost에 대해 동일 피처셋으로 성능을 비교함

Model	train AUC	valid AUC	etc.
XGBoost	0.8318	0.7081	early stopping 40
LightGBM	0.9285	0.7143	early stopping 40
CatBoost	0.7773	**0.7507**	early stopping 40

Exp2: LightGBM vs. CatBoost w/ 22 features

배경 및 기대효과 SVD 알고리즘을 통해 32차원의 유저, 아이템 임베딩을 생성한 뒤 추가 피처로 활용한 뒤, FE를 통해 22개의 특징을 입력으로 두 알고리즘의 성능을 비교. 연속형 특징을 14개 + 32*2개(임베딩) 추가하였으므로, LightGBM에서 특징 간 상호작용을 잘 포착할 것으로 기대함

결과 및 해석 대회 중에는 public AUC와 valid AUC의 gap을 비교하여 LightGBM이 AUC는 낮지만 일반화 성능을 반영한다고 판단하였는데, final AUC와의 gap에서는 반대의 결과를 보임. public AUC와의 비교도 중요하지만, public이 전부가 아니라는 것을 유념해야겠다고 생각함

Model	valid AUC	public AUC	final AUC	etc.
CatBoost	0.8169	0.7917	0.8170	early stopping 40
LightGBM	0.7987	0.7871	0.7811	early stopping 40

Exp3: Last element LightGBM vs. full LightGBM

배경 및 기대효과 부스팅 배경에서 설명한 것처럼 last element 방식과 full 방식을 모두 적용하여 비교하였음. 데이터의 개수가 300배 차이나므로, full 방식이 더 높은 성능을 낼 거라고 예상했음

결과 및 해석 우리 조 실험 내에서는 Last Element LGBM이 Full 대비 높은 성능을 나타냄. 데이터의 절대적인 건수는 Full 방식이 압도적으로 많지만, Last element 방식에서는 집계를 통해 시퀀스의 정보를 집약적으로 제공하여 public AUC에서는 높은 성능을 냄. 그러나 Final AUC를 확인해보니, 일반화 성능에서는 Full LightGBM이 더 높았으므로, 절대적인 데이터의 양을 무시할 수 없다는 것을 깨달음

Model	valid AUC	public AUC	final AUC	etc.
Last Element LightGBM	0.7386	0.7565	0.7439	early stopping 40
Full LightGBM	0.6857	0.7107	0.7513	

5. 자체 평가 의견

1. 시행착오
    - 12시 전에 급하게 결과 내다가 prediction 컬럼 값을 리스트로 제출함 FAIL
    - LGBM 카테고리 데이터를 인식하게 학습할 수 없었음. ordinal encoding으로 해결
    - LastQuery 모델 상속 받고 안 받고 다 성능 다름 → weight 초기화시 layer의 순서대로 random 호출하는 거잖아. 근데 seed 가 같아도, 다른 순서로 초기화되면 달라질거 같음
    - 데이터가 부족하여 논문에 제시된 모델의 성능과 가깝게 복원하기 어려웠음
    - sequential 모델에서 userID 넣었을 때 왜 과적합되는지 아직도 모르겠음
    - LightGCN에서 train이랑 valid 왜 제대로 안 쪼개지는지 모르겠음
    - Feature로 현재까지 맞춘 문제의 개수 정보를 넣어줄때, 현재 정답을 안 빼줬을 때 valid auc 1.0 찍음. data leakage 문제로 보임
2. 잘한 점
    - Feature store 적용은 못했지만 시도해보면서 보완할 부분에 대한 고민을 해본 점
    - 한 달 동안 코어 타임외 시간에도 데이터나 모델 방향성에 대해 고민하고, 함께 논의하며 다양한 실험을 해본 점
    - 강의를 끝까지 듣고 논문도 참고하여 아이디어를 직접 구현해본 점
    - 다양한 모델을 시도해본 점
    - wandb 의 sweep 기능을 이용하여 hyper parameter tuning 을 한 것
3. 아쉬웠던 점
    - 의사결정, 사고의 흐름이 논리적이지 않음, 기록과 고민이 부족함
    - 프로젝트 초기에 목표를 명확히 정하지 못한 점
    - 강의에서 제공되는 아이디어를 완전히 소화하지 못했고, 모델링에 대한 지식과 이해가 부족함을 느낌
    - 개인적 성장을 팀 내 유기적인 활동을 통한 팀의 성장으로 연결시키지 못함. 둘 다 가져가기 위해서 다음 프로젝트 때 보완해야 할 듯
    - 베이스라인을 꼭 써야하는 건 아니지만, 팀 내 공통 코드 구조로 활용하기로 했는데 활용을 잘 못한 것 같아서 아쉬움
    - 깃허브 레포 사용을 잘 못함
    - 데이터에 대한 이해가 부족해서 모델링을 해도 원인을 파악하고 성능을 올리기 쉽지 않았다고 생각함
4. 배운점
    - 시퀀셜 데이터를 꼭 시퀀셜 모델이 아니라 task에 맞게 다양한 방식으로 모델링 할 수 있다는 것을 알게 됨
    - 코드 구현보다, 모델과 피처에 대한 구체적인 가설을 기록하고, 그 결과를 해석하는 것에 더 초점을 두는 것이 좋을 듯
    - 결과론적인 측면이 있으나, 시드 앙상블한 것보다, 시드 고정한 것이 더 효과적이었어서 굳이 하지 않아도 됐었을 듯
    - 가중치 초기화에 따라 모델의 성능이 달라질 수 있다는 것을 깨달음
    - 실험 관리의 중요성과 실험 관리 툴로서 wandb의 필요성을 깨달음

    ▼ Feature Store 구현시 고려해야 할 것들

        1. Feature 공유
        2. Feature 정합성

3. Feature 버전 관리
    a. feature 생성에 관여한 데이터 소스 정보
    b. 생성하는 과정에서 적용된 변환 연산 정보
    c. 생성 주체
    d. 활용 주체
4. feature 모니터링
    a. 훈련데이터 저장소인 오프라인 저장소와 서빙 데이터 저장소인 온라인 저장소의 통계적 일치성 유지 위함

5. 다음 프로젝트에서 보완할 점
    - 일단 강의에서 제공되는 아이디어부터 적용해보는 연습을 프로젝트 초기에 하면 좋을 것 같음
    - 강의에서 아이디어를 얻고 그 과정을 소화하는데 70% 정도의 시간을 투자하고, 추가적인 의견과 가설을 검증하는데 30%를 사용하면 좋을 것 같음
    - 페어 프로그래밍을 해보니 재미도 있고, 같이 이야기하면서 진행하다 나오는 아이디어들이 대체로 재밌는 편이었던 것 같음
    - 깃허브 사용 룰을 우리 팀 전원이 공통으로 공유하여 잘 사용할 수 있음 좋겠고, 대신 룰을 칼각으로 지키기보다는, 실수하면 같이 보완하면서 오히려 더 성장할 기회로 삼으면 좋을 듯.
    - 프로젝트 초기에 모델과 파이프라인에 대해 서로 공통된 이해를 했는지 확인하고 시작하면 서로 이야기할 때 더 도움이 될 것 같음
    - 새로운 아이디어를 구현하기 전에 미리 어떤 모델을 왜 구현하는 것인지 공유하는 것도 좋을 듯

서동은

■ 개인 학습 측면

모델링 능력 향상, EDA부터 FE, 모델, 앙상블까지의 전반적인 지식 습득과 코드로 적용하는 연습, 하나의 모델에 대한 여러 사람의 github 소스 코드를 읽고 이해하고 적용해보기, 논문과 코드 동시에 이해하고 개선해보는 작업

■ 공동 학습 측면

1. 베이스라인 코드 리팩토링 참여 : wandb , pt 파일, submission 파일 이름 변경, 앙상블 코드 추가
2. 기존 베이스라인 코드에 주어진 모델 개선 : sequence 모델의 마스킹의 유무, positioinal encoding의 유무 등으로 시퀀스 계열 모델들의 성능 비교 실험
3. 기존 베이스라인 코드에 새로운 방안 도입 : 과거 대회와 논문을 참고하여 Saint+ 를 구현하고 도입, elo에 나오는 feature를 사용해봄, lightgcn의 valid data를 생성하는 방안에 대한 탐색, ightgcn convolution 함수 다른 함수로 변경하며 실험, 데이터 증강 실험

■ 사용한 지식과 기술

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

모델링에 필요한 기반 지식과 pytorch 사용 능력을 마련하는 것에 성공했다. 논문의 코드를 베이스라인으로 옮겨서 인풋을 알맞게 변경하여 실행시킬 수 있게 되었다. 기존 논문의 실험에서 사용했던 여러 대안들을 직접 대체하며 구현해보고 성능을 비교해볼 수 있게 되었다. 특히, 모델링에서 가장 중요하다는 트랜스포머 모델과 그래프 모델을 구현할 수 있게 되었다.

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

강의에서 제공하는 미션을 수행하면서 대회 문제에 대한 아이디어를 얻을 수 있고 적용해보면서 실력이 느는 것을 느꼈다. 가장 중요한 것은 대회 콘텐츠가 흥미로워졌다는 것이다. 이전에는 대회가 너무 낯설어서 나랑 안 맞는 길이라고 생각했었는데 이번에 다시 흥미를 되찾아 재밌게 공부한 것 그 자체로 의미가 있었다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

트랜스포머 성능이 안 좋은 것의 원인을 데이터 부족에 있다고 생각해서 데이터 증강을 하는 데 시간을 많이 쓴 것 같다. 결국 오버피팅의 벽을 넘지 못 했기 때문이다. 모델의 성능 자체를 올리기 위해 노력했으면 더 시간을 효율적으로 사용할 수 있지 않았을까, 그리고 더 많은 것을 공부할 수 있지 않았을까하고 아쉽게 생각한다. 하지만, 지금의 수준에서는 결과와 상관없이 어떠한 노력이든 경험이 된다고 생각한다. 모델링에만 집중하여 EDA, 피처 엔지니어링의 실력이 미숙하다는 점, 그리고 deep learning model에만 집중하여 maching learning을 사용해보지 못한 점도 아쉽게 생각한다. 그리고, 다양한 cross validaton 전략이나 앙상블 기법을 스스로 도입해보지 못한 것도 아쉽다. 이 모든 것을 반영하여 베이스라인 코드에서 벗어나 나만의 코드로 재구성하고 싶다.

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

문제 정의부터 해결법까지 강의에 의존하지 않고 스스로 설계해보고 싶다. 이전까지 소홀히 했던 EDA와 피처 엔지니어링에 초점을 두고 데이터를 분석하는 능력을 배양하는데 집중할 것이다. 문제를 해결하는 창의적인 해결책을 고안해내고 스스로 모델링하여 최종적으로 좋은 성능을 가진 모델을 만들고 싶다. 모든 행동이 문제를 해결하는 근거와 연결되어 모델이 설명 가능한 상태가 되는 것을 목표로 한다.

신상우

학습 목표

- 시퀀스 모델링에 집중하기(Transformer)
- Gradient Boosting Model 사용해보기, Feature Engineering
- 협업을 위해 코드 잘 작성하기

시퀀스 모델링

lstm부터 시작해 베이스라인 모델들을 실험하던 중 lstmattn이 lstm보다 public 스코어 기준 성능이 떨어진 다는 것을 발견함. attn 연산이 lstm 연산 결과들을 섞어서 학습을 방해 한다고 생각했고, attn을 먼저 수행해서 중요한 피처 정보를 뽑아내게 한 후에 lstm연산을 수행하도록 attnlstm을 구현함. public기준 lstmattn보다는 성능이 좋았지만 여전히 lstm보다 성능이 좋지 않았음.

attnlstm 구조를 더 발전시켜 트랜스포머 인코더와 lstm을 사용하기 위해 dkt와 비슷한 Ruiid 대회에서 1등 솔루션인 lastquery 모델을 도입함. QK 행렬곱 연산의 복잡도를 O(L^2) 에서 O(L)로 개선한 모델로 입력 시퀀스의 최대 길이를 늘려서 한 번에 많은 시퀀스를 학습할 수 있기 때문에 확장성이 높다고 판단함. 학습 속도 자체도 빠르기 때문에 다양한 실험을 빠르게 할 수 있다고 생각함.

유저가 과거에 푼 문제 정보를 많이 볼수록 예측을 더 잘할 것으로 판단하여 입력 시퀀스의 최대 길이를 바꿔보는 실험을 진행함. public score가 떨어져서 최대 길이가 dkt 데이터에서는 의미가 없다고 생각했는데, private 결과는 의미가 있었음. 모델이 한 번에 볼 수 있는 데이터의 양이 늘어나서 성능이 좋아진걸로 해석됨.

riid! 대회에서 의미있게 작용한 문제 풀이 소요 시간 feature를 추가함(3시간 이상 걸린 문제는 3시간으로 처리함). 문제 풀이 소요 시간으로 문제 난이도를 유추할 수 있다고 판단해 유의미한 feature가 될 것으로 기대함. public score가 얼마 오르지 않아서 의미가 없다고 생각했는데, private에서 더 큰 폭으로 상승함. 문제의 난이도 정보가 예측에 도움을 준 것으로 해석됨. 시간 기준과 결측값을 더 신경 썼다면 더 좋았을 것 같음

한계 및 개선점

- CV전략, public 데이터의 비중이 적은 경우 너무 public 스코어에 의존하지 말자
- 그래프 모델을 이용해 생성한 임베딩을 이용한 시퀀스 모델링을 시도해보고 싶음
- 완전히 새로운 구조의 나만의 모델을 논리적으로 설계하고 실험해보고 싶음

LightGBM과 Feature Engineering

시퀀스 데이터를 집계해서 LightGBM으로 문제를 해결할 수 있음. 이때 시퀀스 데이터를 모두 사용해 각각의 정답 여부를 예측하는 방식으로 모델을 설계함. 데이터 증강의 효과가 있어서 데이터를 모두 사용하는게 더 유리할 것으로 판단함. 미션에서 나온 Feature Engineering을 LightGBM에 적용해보며 실험했고, 과거에 유저가 푼 문제에 기반해서 나온 유저의 실력과 관련된 feature들이 의미가 있었음. 또한 ordinal 인코딩을 적용 유무가 성능에 큰 차이를 냈음

한계 및 개선점

- 미션 외에 스스로 인사이트를 가지고 유의미한 feature를 많이 만들어보지 못한 점이 아쉬움
- LightGBM의 많은 하이퍼파라미터를 온전히 최적화하지 못해 아쉬움
- EDA와 팀원들과 아이디어 공유를 통해 다음 대회에서는 의미 있는 feature를 많이 만들고 싶음

협업

- lastquery와 lgbm 모델을 다른 팀원도 사용할 수 있도록 베이스라인에 추가함
- 페어 프로그래밍을 제안해서 협업 공유 문화를 조성함

이주연

학습 목표

개인 학습 목표 지난 대회를 회고하며 아쉬웠던 점을 추려내어 개인 목표를 설정함. 그 중 베이스라인 코드에 의존하여 제한적으로 실험을 진행했던 것이 가장 아쉬웠기에, **베이스라인과 독립적으로 end-to-end를 직접 경험하는 것**을 가장 큰 목표로 두었고, 이외에 체계적인 실험 관리와 교차 검증, LR 스케줄러 사용, 다양한 모델 디벨롭을 부차적인 목표로 삼음

팀 학습 목표 이번 프로젝트에서는 팀 공통의 학습 목표를 명시적으로 정하지 못했음. 개인의 성장이 있으면 팀 전체의 성장이 있을 것으로 기대함

목표 달성을 위한 노력

- **개인 학습 측면** 베이스라인에 의존하지 않고, end-to-end 파이프라인을 이해하고 직접 구현하는 것에 초점을 둠. 부스팅 모델, lastqueryLSTM, LSTM, ensemble 에 대하여 end-to-end 파이프라인을 직접 구현해봄
- **팀의 학습 측면** 팀 모집 이후, 별도의 공동 학습 기간 없이 바로 프로젝트를 시작하여 분위기를 만드는 것이 어려웠음. 공유하려는 문화를 만들기 위해 먼저 실험한 것에 대해 공유하고, 다른 팀원들도 공유할 수 있도록 지목하려고 노력함. 팀원이 공유할 때 귀담아 듣고, 이해한 것이 맞는지 확인하고, 새로운 아이디어에 대해 공유하려고 노력함

모델 개선 방법

- **시스템 측면** 구성한 모델에 대해 lr scheduler를 활용하여 하이퍼파라미터 튜닝 없이도 안정적인 성능을 낼 수 있도록 하였으며, CV를 모듈 내에 구현하여 LB와 비교할 수 있게 함
- **시퀀셜 모델링** Riid 대회의 태스크와 데이터 구조가 i-Scream 대회와 유사하여 1등 솔루션을 활용해보고 싶었으며, 해당 모델이 비교적 높은 성능을 내는 것인지 비교하기 위해 LSTM, RNN을 통해 베이스라인 모델의 성능을 우선적으로 파악함
- Riid 대회 1등 솔루션(LastQueryTrmLSTM)은 트랜스포머 인코더와 LSTM을 연결한 구조에서, 트랜스포머의 셀프어텐션 연산 시 전체 쿼리가 아닌 마지막 쿼리에 대해서만 연산하도록 변형한 것. 성능에서 큰 차이가 없으나, 연산 비용이 줄어드는 것을 의도하였다고 하였는데, 두 모델을 논문을 읽고 직접 구현하여, 성능 측면에서는 0.005의 저하가 있으나, 총 학습 시간이 1/3 줄어드는 것을 확인함
- 특징의 수가 적을 때와 많을 때 vanilla LSTM과 LastQueryTrmLSTM의 성능 차이도 실험해봄. 특징이 적을 때(5개) 0.015 이상의 차이가 발생하여, 트랜스포머 인코더는 타겟 예측에 중요한 특징을 찾는 데 분명히 도움이 된다는 것을 알게 됨. 다만 특징의 수가 충분히 많은(22개) 경우에는 두 모델의 성능 차이가 거의 없었고, 일반화 측면에서는 vanilla LSTM이 더 높아, 트랜스포머 구조는 데이터의 양이 매우 많고, 특징의 수가 적은 상황에서 더 유리할 수 있겠다고 생각함
- **CF** 데이터 형태와 태스크는 시퀀스에 기반하여 마지막 문제에 대한 점수를 맞추는 것이지만, 시퀀스에서 특징을 찾아내는 것만큼 다른 유저의 행동 패턴을 모델링하여도 잘 맞출 거라는 생각이 듬. **SVD** 에서 학습 데이터에 포함되지 않은 유저에 대해서는 아이템을 사용한 데이터가 있더라도 예측이 불가능하다고 생각했는데, MF나 NCF 같은 딥러닝 모델이 아닌 행렬 연산을 활용하였기 때문에, 역행렬을 활용하면 예측이 가능하다는 것을 알게 됨 **LightGCN** 의 경우 transformer geometric을 활용하여 구성하였는데, 인접 행렬, 차수 행렬을 직접 구현하지 않아도 그래프 구조의 전달만으로 예측이 가능했음. 다만, 그 동작 원리를 코드와 직접 매칭할 수 없어, 다음 프로젝트에서는 소스 코드를 분석하여 행렬 연산에 대해 더 깊이 이해해야겠다고 생각함
- **앙상블 측면** 시퀀셜 모델 대비 그래프 모델이 성능이 잘 안나올 수 있다는 얘기는 들었지만, 앙상블에서 다양성을 확보하여 bias를 줄이기 위하여 머신 러닝 모델과 그래프 모델을 모두 활용함. 최종 AUC 또한 다양한 모델을 모두 활용하여 평균 내었을 때가 가장 높은 성능을 내었으며, 이에 따라 평균을 낼 때에는 다양성을 충분히 확보해야 성능 향상을 기대할 수 있다고 판단함

잘한 점

- 개인의 목표를 이루기 위해 충분히 노력하였고, 많은 부분을 이뤄낸 점. 특히 가장 큰 목표였던 **베이스라인과 독립적으로 end-to-end를 직접 경험하는 것**을 충분히 이뤄냈다고 생각함.

아쉬운 점과 보완할 점

- 지난 팀과의 프로젝트에서 당연하게 여겼던 부분들이, 이번 팀과 프로젝트에서는 당연하지 않았음. 당연한 것은 없으니 서로 의견을 많이 공유하는 것이 중요하다는 것을 깨달음
- 개인적인 성장에 집중하느라 팀 전체의 성장과 공유를 많이 놓쳤다고 생각함
- 팀과 개인의 목표를 모두 이룰 수 있도록 팀 회고도 추가적인 시간을 내어 진행했기 때문에, 다음 프로젝트에서는 함께 잘 이뤄나갈 수 있을 거라고 생각함

이현규

나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

- 팀의 목표: 공유를 통한 공동 성장
- 개인의 목표: 다양한 실험을 통해 경험을 쌓는 것

  다양한 실험을 하진 못했지만 공유 또한 제대로 하지 못했기 때문에 달성하지 못했다고 생각함.

나는 어떤 방식으로 모델을 개선했는가?

시계열 데이터로 접근하여 시간적 순서를 보다 잘 표현할 수 있는 Transformer와 LSTM을 사용하여 풀이 정보의 연관성을 표현하는 모델을 설계하려 함.

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

논리적으로 가능성이 있다고 생각한 부분에 집중하여 실험한 결과 유의미한 성적을 보였고 이 자체로 새로운 경험을 했다고 생각함.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

이전에는 갇힌 생각과 50개 모델 돌려서 가장 점수 높게 나오면 그게 이번 모델이 적합한 모델!

정도로 생각했는데 조금 더 열려 있다는 점이 더 재밌게 다가왔고 열의가 올라오는 효과가 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

1. 공유

   팀 목적인 공유를 잘하여 같은 일은 최소화하고 모르는 부분은 팀원의 도움을 받을 수 있도록 하는 것이 목적이었지만 생각보다 시간 비용도 많이 들고 다 같이 모르는 부분 그리고 집중이 깨지는 것. 여러 문제에 마주한 것이 오히려 독이 되어서 개인적으로 많이 지치는 부분이었다고 생각한다.

2. 체력

   feature에 신경을 못 쓴 것이 점수로 보였다. 시간은 넉넉하게 주어졌지만, 그만큼 스트레스도 넉넉하게 받은 탓인지 스스로에게 이 정도면 충분해. 쉽게 포기를 했다. 이 부분에서 팀에게 미안한 마음이 크다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

개인적으로는 체력 안배와 스트레스 관리를 위해 **시간 내 해결** 방법을 시도해 볼 예정이다.

팀적으로는 이번 대회에서 공유를 가장 중점적으로 바라봤던 이유는 다음 대회 그리고 파이널 프로젝트에서도 서로 힘이 되어야 하기 때문에 미리 연습하자는 취지였다. 처음이라 많이 삐걱거렸지만 다음 프로젝트에서는 그럴싸한 팀처럼 행동할 수 있지 않을까 생각함. 그렇기에 더욱 **다양한 시도**를 하고 **공유**하며 같이 고민하고 해결할 수 있게 **적극적**으로 나서보려 한다.

이현주

지난 4주 동안 DKT 에 대한 프로젝트를 진행했다.

잘했던 점

- wandb 의 sweep 기능을 사용하여 hyper parameter tuning 을 할 줄 알게 되었다.
- 각각 따로 사용하던 argparse 와 yaml 을 합쳐서 한 번에 실험할 수 있도록 만들었다.
- valid data 가 split 되지 않고 있었을 때, 디버깅을 통해 내가 원하는 방식대로 구현을 해냈다.
- Looker studio 를 이용하여 보고서를 작성할 줄 알게 되었다.
- 깃허브에 이슈를 남기고, 그 이슈에 대한 브랜치를 만들어 PR 을 남기는 습관이 생겼다.

아쉬웠던 점 && 해결해낼 수 있는 방법

- 강의를 듣고 나서 추가적으로 정리를 못했다. 내 것으로 만드는 시간이 부족했다.
  → 강의를 들으면서 중요한 것만 메모하는 방식으로 정리를 해야겠다.
- wandb 에 찍히는 값만 보고 학습이 잘되고 있다고 판단했는데 사실 안되고 있었다.
  → 제대로 진행이 되는지는 리더보드만이 판단해줄 수 있다는 걸 명심하고, 매일 나에게 주어진 제출 횟수를 다 쓰며 판단을 내려야겠다.
- PR 을 큰 것만 보내서, 팀원들과 리뷰하는 시간을 얼마 못 가졌다.
  → 보다 좀 더 작은 기능으로 쪼개서 PR 을 보내고, 리뷰를 요청하는 습관을 들여야겠다.
- 건강이 진짜 안 좋아졌는데 시간 내기가 어려워서 그냥 버텼더니 상태가 악화됐다.
  → 당장 내일 병원을 가겠음…! 당분간은 물리 치료도 계속 받아야겠다…

숙제

- 그 날 공부한 거 딱 한 페이지로 정리해서 제대로 이해했는지 확인하기
- 제출 횟수 싹 다 쓰기
- 병원 가기

조성홍

1. 프로젝트 기간 집중한 부분

- Feature Engineering: 지난 레벨 1 프로젝트 당시 데이터를 좀 더 깊게 다뤄보지 못했던 부분이 아쉬움으로 남아, 이번 프로젝트에서는 데이터를 분석해보는 연습을 하는 것에 집중함.
- Feature Store 구축: 역시나 지난 대회 때 아쉬웠던 Feature 생성시 팀원간 중복되는 비효율을 줄이고 좀 더 효율적으로 재사용하기 위한 방법으로 feature store를 만들어 사용해보려고 함.

2. 잘했던 부분

- 지난 대회 때 아쉬웠던 부분을 보완하고자 시도해봤던 점이 좋았음.
- 타 팀원이 설계한 코드를 빠르게 분석해 처음 다뤄보는 라이브러리를 비교적 빠른 시간안에 적용해 원하는 실험 목적을 달성했음.

3. 아쉬웠던 부분

- Feature Engineering 을 하면서, EDA를 통한 통계적 특성이나, 관련 도메인 학술 연구 자료 등의 근거를 통해 추론하려는 노력이 부족했음.
- 또한, 생성한 Feature의 결측치 처리나, 의도한 대로 생성되었는지를 확인하는 과정이 없어, 초기에 생성한 feature의 오류를 프로젝트 후반에서야 수정하는 등 미흡한 부분이 많았음.
- Feature Store를 구축하는 과정상 부족한 사전 조사로, feature 조회 성능(feature 1개 추가시 조회시 30sec 추가됨), 모델별로 달라지는 후처리 방식(feature set의 최종 데이터 형태는 모델에 종속적인데, 모델에 상관없이 하나의 테이블로 설계한 feature 저장소 형태로는 사용하는 모델별로 별도 처리를 제거하지 못함) 등 설계상의 미흡한 부분이 많았음. 이 부분을 개선해 사용하기엔 남은 프로젝트 기간에 여유가 없어 개선해보지 못하고 종료한 부분이 아쉬움.
- 시도한 실험에 대한 결과를 정리하는 과정에서, 각 실험별로 정리한 항목이 일치하지 않아 취합해 분석하는 과정에서 어려움이 있었음.
- 강의에서 제공해주는 다양한 방법들이 있었는데, 이를 등한시하며 프로젝트를 진행한 것 같다. 실제로, 도출한 다양한 방법들 중 대부분이 이미 강의에서 제공되었던 내용들이라 좀 더 아쉬움이 남는다. 선입견이 생길 것 같아 의도적으로 배제하고 진행했는데, 지식이 아예 없는 상태에선 원하는 수준의 창의적인 아이디어가 나오진 않았다.
- 새로 결성된 팀으로 진행한 첫 프로젝트였는데, 각자 원하는 부분에 대한 공유나 서로에 대한 이해가 부족했음. 프로젝트 중반쯤에 이를 해결해보고자 논의하고 나름의 룰을 만들어 적용해보았지만, 원하는대로 잘 시행되진 않은점이 아쉬웠음.

4. 다음 프로젝트에 시도해 보고 싶은 부분

- Feature Set 혹은 데이터 소스를 팀 공통적으로 관리할 수 있는 환경을 프로젝트 초기에 팀 내에서 원활하게 사용 가능한 수준까지 빠르게 구현하는 것을 목표로 시도해볼 계획.
- Feature Engineering 시 가설을 뒷바침 할 수 있는 근거를 실제 구현에 앞서 찾아, 팀원들을 설득해보는 프로세스를 수행해볼 계획. 진행하며, 설득을 용이하게 할 수 있는 항목들을 정리해 프로젝트 종료 시 재사용가능한 템플릿으로 만들어내는 것이 최종목표.