Deep Knowledge Tracing Wrap-up Report

RecSys-2조(R_AI_SE)

김수진, 김창영, 박승아, 전민서, 한대희, 한예본

1 프로젝트 Wrap Up

1.1 프로젝트 요약

- 주제: 유저의 학습 상태에 대해 새로운 문제를 풀 가능성을 예측
- 배경
  - 시험 성적은 우리가 얼마만큼 아는지 평가하는 한 방법이지만, 학생 개개인의 이해도를 기리키는 '지식 상태'는 알 수 없다.
  - 이를 해결하고자 딥러닝 방법론인 'Deep Knowledge Tracing'이 등장하였고 맞춤화 교육을 위해 아주 중요한 역할을 한다.
  - 대회에서는 지식 상태를 예측하기보다는, 주어진 문제를 맞출지 틀릴지 예측한다.
- **Task**: test_data 마지막 문제의 정답여부가 주어지지 않으며, 이를 예측해야 한다.

1.2 프로젝트 일정 및 협업 방식

1.2.1 프로젝트 일정

1) 프로젝트 개발환경 구축(Server, WandB, Github 등)

2) EDA를 통한 데이터 구조 파악 및 Feature Engineering

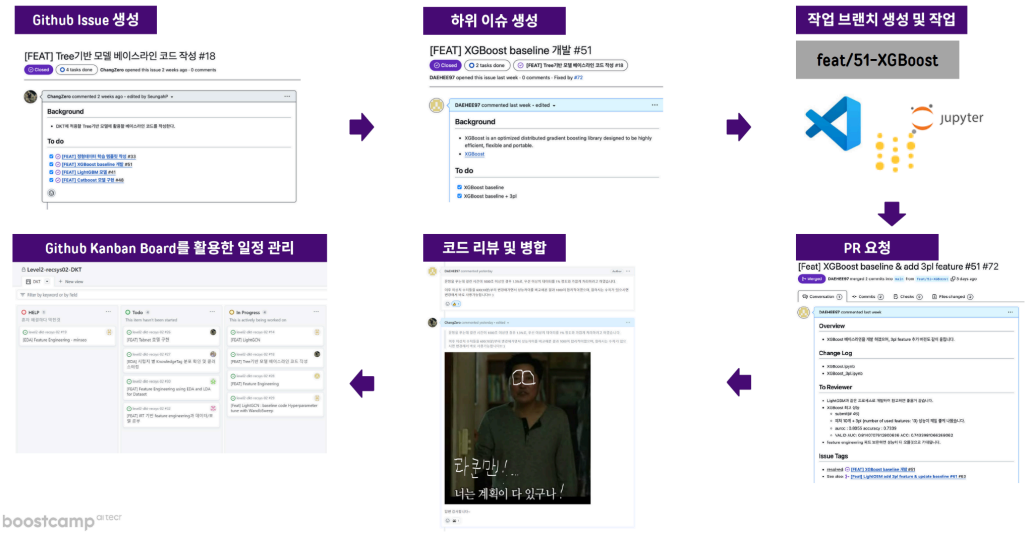
3) 가설 기반 모델링

4) Weighted Ensemble을 통한 최종 결과물 제작

1.2.2 협업 방식

Github 기반 작업

- 데일리 스크럼/피어세션 시간을 활용해 PR 및 코드 리뷰 후 병합 진행
- 이슈 기반 작업
- 깃 칸반 보드를 활용한 프로젝트 일정 관리



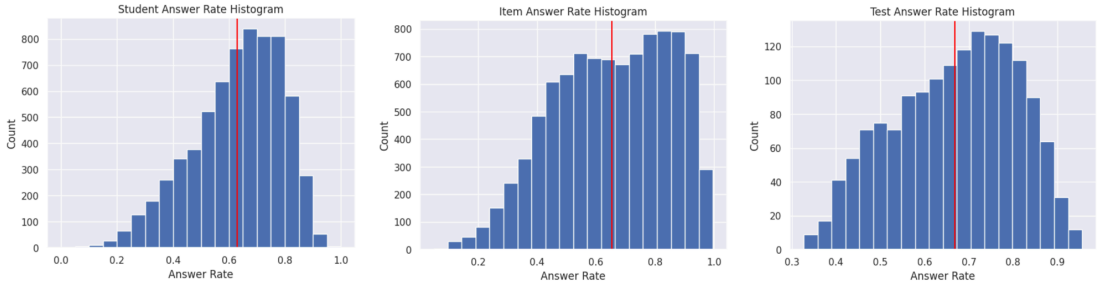
1.3 프로젝트 팀 구성 및 역할

전체	모델 실험, Wrap-up report 작성
김수진	Transformer, SASRec, BERT4Rec, SAINT(+), CustomModel 구현, 모델 성능 실험, DKT Baseline 반자동화
김창영	EDA, 정형데이터셋 베이스라인, TabNet, Tree모델 모듈화, Out Of Fold, WandB sweep HPO
박승아	데이터 EDA, Last Query Transformer RNN, SAINT(+) 모델, T-fixup 구현, 하이퍼파라미터 튜닝
전민서	Transformer, Graph, LSTM, SASREC, Bert4REC, ML, OOF, Wandb, LastQuery
한대희	EDA, Feature Engineering, XGBoost, LGBM, CatBoost, GridSearchCV, AutoGluon
한예본	IRT 기반 Feature Engineering, Transformer, Graph, LSTM, SASREC, ML

1.4 프로젝트 수행 결과

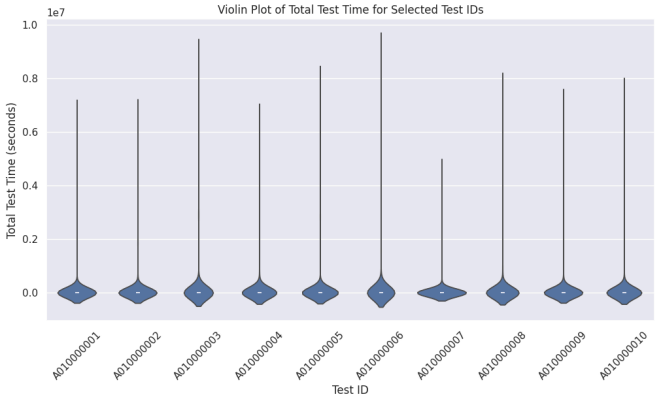
1.4.1 Data EDA

- 학생 별, 문항 별, 시험지 별 정답률 분석



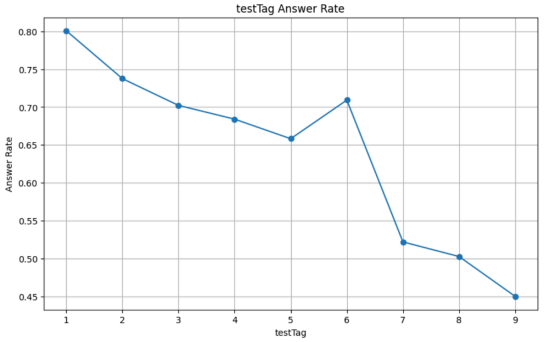
학생 별, 문항 별 ,시험지 별 정답률을 시각화하여 차이가 있음을 확인했다.

- 시험지별 시험 시간 분석



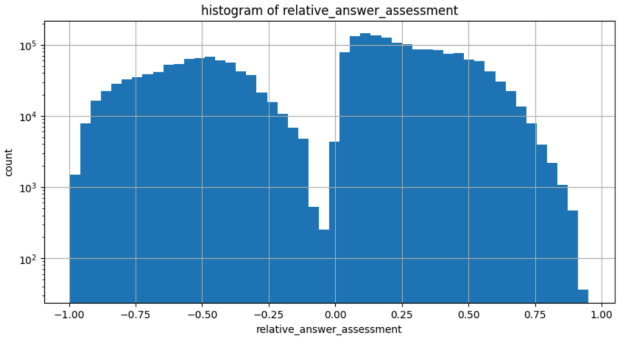
시험지별 유저의 시험시간을 시각화 및 4IQR분석을 진행했을때 상한을 넘는 유저가 존재함을 확인했다.

- 시험지 대분류 분석



시험지를 파싱하여 가운데 수를 대분류로 보고 시각화진행, 시험지 대분류에 따라 정답률 차이 확인했다.

- 문제의 정답여부와 평균 정답률간 차이 분석



유저 단위로 총 평균이 높을수록, 다른 학생들에 비하여, 어려운 문제에 대한 정답 비율이 높음을 확인했다.

1.4.2 Feature Engineering

IRT

- 문항마다 보편적인 고유한 특성을 가지는 문항함수를 이용해 모델리앟여 잠재적 구인에 대한 보편적 측정을 시도하는 이론으로, 특정 집단에 한정되지 않는 보편적인 문항 고유의 특성을 도출하여 피험자에 대한 표준화된 분석을 하는 접근이다.
- R의 ltm 패키지를 이용하여 문항별 3모수(난이도, 변별도, 추측도)와 사용자의 능력 수준을 feature로 생성하였다.

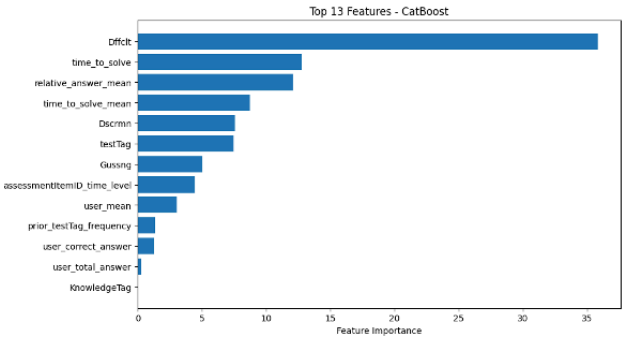
$$P_i(\theta) = P(X_i = 1 \mid \theta) = c_i + (1 - c_i)\frac{e^{a_i(\theta - b_i)}}{1 + e^{a_i(\theta - b_i)}}$$

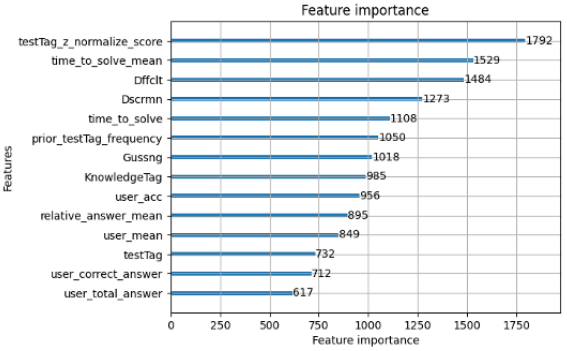
어떤 능력 θ를 가진 가진 사람이 문항 i를 맞출 확률을 로지스틱 모형으로 표현
이때 확률 분포에 적용되는 모수가 난이도(b), 변별도(a), 추측도(c)

Feature	dtype	Description
`dffclt`	float64	문항별 난이도
`dscrmn`	float64	문항별 변별도
`gussng`	float64	문항별 추측도
`user_level`	float64	시험지별 학습 능력 수준

FE

- 원본 데이터를 기반으로 도메인 지식, Task 등을 고려하여 문제 해결하는 데 도움이 되는 피처들을 생성하고, 결측치와 이상치 데이터를 처리하여 모델에 적합한 형식으로 변환하는 작업을 수행하였습니다. 유의미한 피처들을 선택하기 위해 feature importance 을 확인하고 이를 통해 최종적으로 사용할 피처들을 선정했습니다.



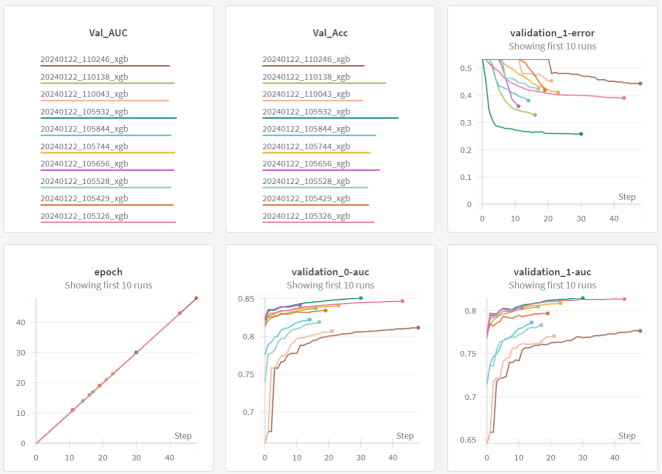


Feature	dtype	Description
`user_correct_answer`	float64	유저별 누적 정답 횟수
`user_total_answer`	int64	유저별 누적 제출 횟수
`user_acc`	float64	유저별 누적 정답률
`user_mean`	int64	유저별 평균 정답률
`prior_testTag_frequency`	int64	유저별 testID 가운데 수 별 이전에 몇번 풀었는지
`time_to_solve`	float64	유저별 문항을 해결하는데 걸리는 시간
`time_to_solve_mean`	float64	유저별 문항을 해결하는데 걸리는 시간의 평균
`relative_answer_mean_level`	float64	유저 학습 수준 1 (유저별 상대적 정답률의 평균)
`assessmentItemID_time_level`	float64	유저 학습 수준 2 (유저별 평균 문항 풀이 시간으로 처리)

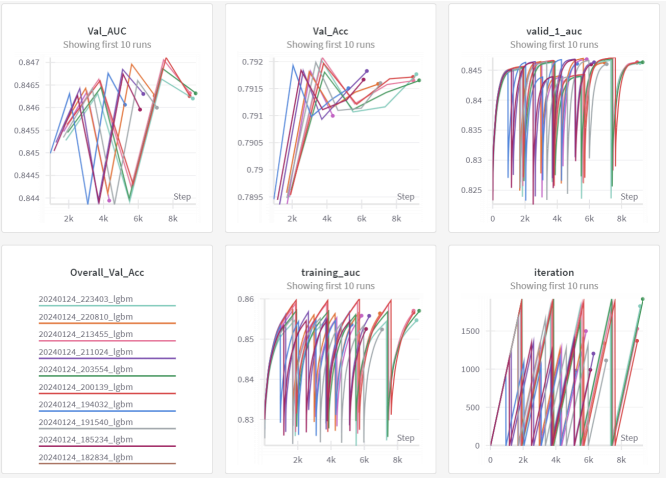
1.4.3 Decision Tree Model

시간적인 순서에 상관 없이 정답 여부만을 예측하는 '정형 데이터 분류 문제'에 대한 접근 방식으로, 다양한 부스팅 트리 모델을 주어진 데이터와 task에 맞게 개발하였다. 모델들을 효과적으로 테스트하고 성능을 최적화하기 위해 GridSearchCV 와 Wandb sweep을 통해 하이퍼파리마티 튜닝을 자동화 처리를 하였으며, 최종적으로 CatBoost 와 LigtGBM은 높은 성능을 달성하였다.

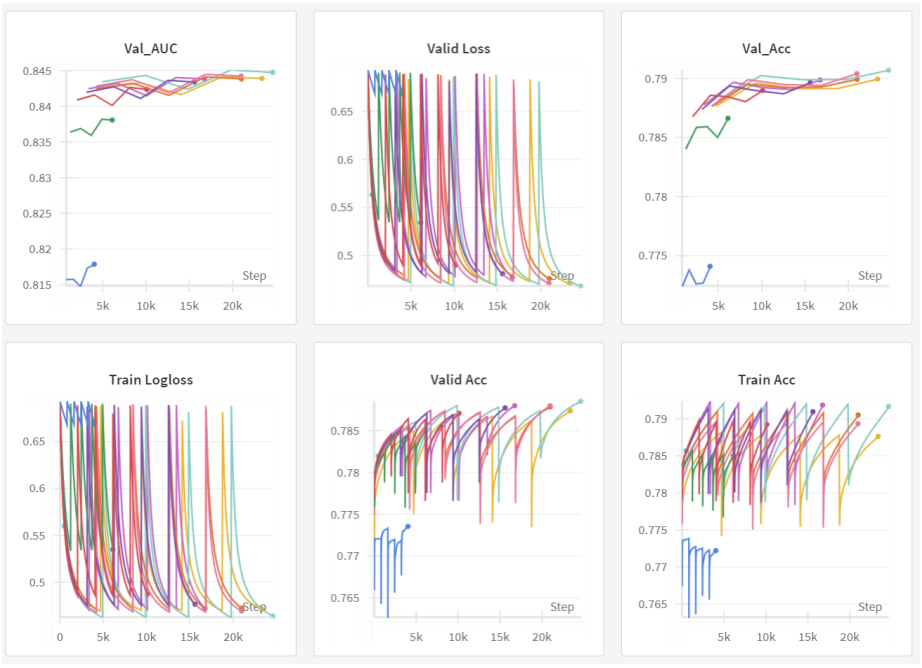
- XGBoost: 정규화된 손실 함수와 경사 부스팅 알고리즘을 결합한 트리 기반 앙상블 모델로, 병렬 처리와 효과적인 특성 중요도 추정을 통해 대용량 데이터에 적합하며, 높은 성능을 보이는 머신러닝 모델이다.
- XGBoost: A Scalable Tree Boosting System



- LightGBM: 리프 중심 히스토그램 기반의 분할 방식을 사용하여 XGBoost 보다 빠른 학습과 예측 속도를 보인다. 분산 학습과 카테고리 특성 최적화에 강점을 가지는 머신러닝 모델이다.
- LightGBM: A Highly Efficient Gradient Boosting Decision Tree



- CatBoost: 범수형 변수 자동 처리와 내부적인 규제 기능을 갖춘 모델로, 과적합 방지를 위한 최적화된 카테고리 처리를 지원한다. 즉, 범주형 데이터에 강점을 보이는 알고리즘이다.
- CatBoost: unbiased boosting with categorical features



1.4.4 AutoGluon

전처리 과정, 알고리즘 선택, 모델 학습 및 평가, 파라미터 튜닝 등 복잡한 단계를 단순화 하고, 자동화 하기 위하여 AutoGluon을 도입하였다.

AutoGluon의 전처리 과정 단순화, 알고리즘 선택 자동화, 모델 학습 및 평가 자동화, 하이퍼 파라미터 튜닝 자동화 와 같은 주요 기능들을 이용하여 높은 성능을 달성하였다.

```
*** Summary of fit() ***
Estimated performance of each model:
                     model  score_val eval_metric  pred_time_val     fit_time  pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  fit_order
0      WeightedEnsemble_L3   0.804950    accuracy     360.509053  2409.582476                0.394867          70.171329            3       True         10
1  RandomForestGini_BAG_L2   0.804488    accuracy     345.660675  1871.147307               10.147823         327.112043            2       True          8
2          LightGBM_BAG_L2   0.804145    accuracy     337.620583  1598.863739                2.107730          54.828475            2       True          7
3        LightGBMXT_BAG_L2   0.803547    accuracy     338.031329  1598.403353                2.518476          54.368090            2       True          6
4  RandomForestEntr_BAG_L2   0.803524    accuracy     345.340156  1903.102539                9.827304         359.067275            2       True          9
5      WeightedEnsemble_L2   0.802606    accuracy     335.908647  1580.847961                0.395794          36.812698            2       True          5
6          LightGBM_BAG_L1   0.800224    accuracy      95.088865   533.484463               95.088865         533.484463            1       True          4
7        LightGBMXT_BAG_L1   0.798751    accuracy     217.391481  1006.379364              217.391481        1006.379364            1       True          3
8    KNeighborsUnif_BAG_L1   0.741863    accuracy      10.774711     2.459974               10.774711           2.459974            1       True          1
9    KNeighborsDist_BAG_L1   0.733423    accuracy      12.257795     1.711462               12.257795           1.711462            1       True          2
Number of models trained: 10
Types of models trained:
{'StackerEnsembleModel_LGB', 'StackerEnsembleModel_RF', 'StackerEnsembleModel_KNN', 'WeightedEnsembleModel'}
Bagging used: True  (with 8 folds)
Multi-layer stack-ensembling used: True  (with 3 levels)
```

1.4.5 TabNet

TabNet은 정형 데이트에 맞게 설계되었으며, Tree기반 모델의 변수 선택 특징을 네트워크 구조에 반영한 모델이다. 딥러닝 모델이 해석하기 어려운 문제를 sequential attention mechanism을 이용하여 해결했다.

1.4.6 Matrix Factorization

사용자와 아이템간의 상호작용 행렬을 낮은 차원의 잠재 요인으로 분해하여 예측하는 모델이다.

1.4.7 LightGCN

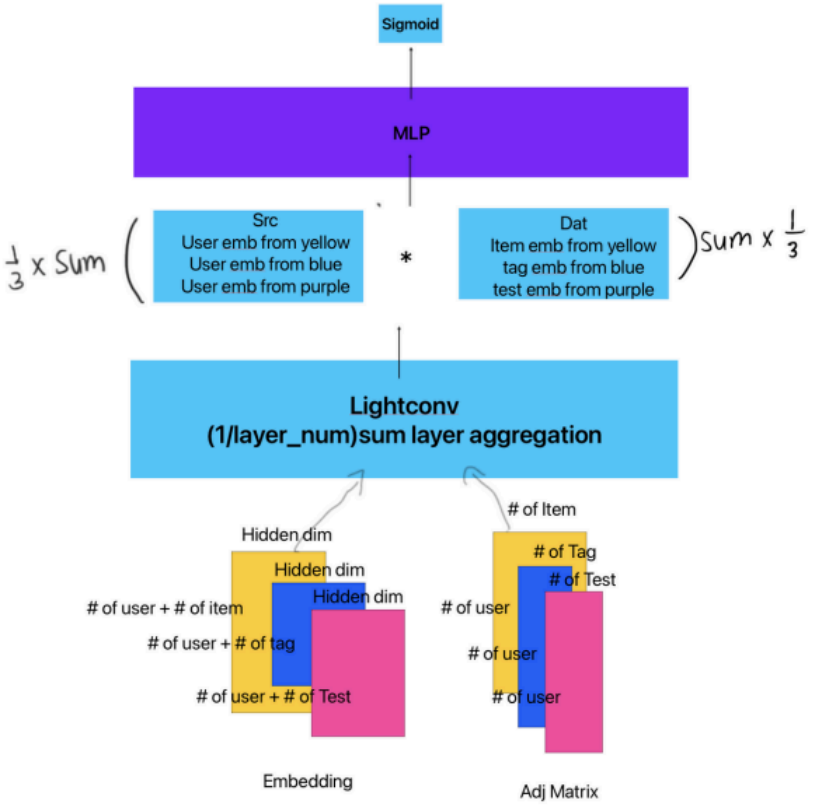
기존 NGCF의 Feature Transformation과 activation function 부분을 제거하여 성능과 속도를 전부 잡은 그래프 기반 추천 모델이다.

1.4.8 SGCN - [Link]

Edge의 부호를 고려하여 node feature를 학습하는 graph convolution network의 부호 고려 버전이다.

1.4.9 LightGcn - stacked(custom)

기존 light gcn 모델이 노드종류 2개, 엣지종류 1개를 이용하는 모델 이었다면, lightgcn을 노드 종류 4개, 엣지 종류 3개로 확장한 모델이다. 마지막에 해당 모델을 차용하였다.

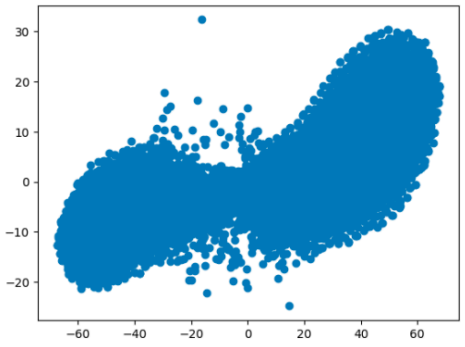


1.4.10. LightGCN + SGCN (custom)

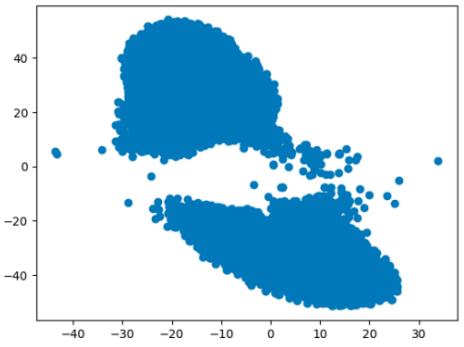
LightGCN의 가벼움과, SGCN의 부호를 고려한 부분을 섞은 모델로 SGCN의 Conv에서 Feature Transform을 제거 한 것의 layer를 activation function 없이 제거하여 sum한 모델이다.

1.4.11 SGCN + LSTM (custom)

SGCN으로 만들어진 노드 임베딩을 유저별 시계열 형태로 변환하여 사용하였다. 하지만 노드 임베딩이 아래와 같이 잘 그룹화 되지 않아서 사용하지는 못했다.

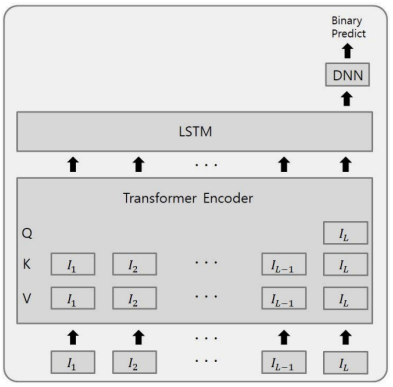


Item Node T-sne



User Node T-sne

1.4.12 Last Query Transformer RNN

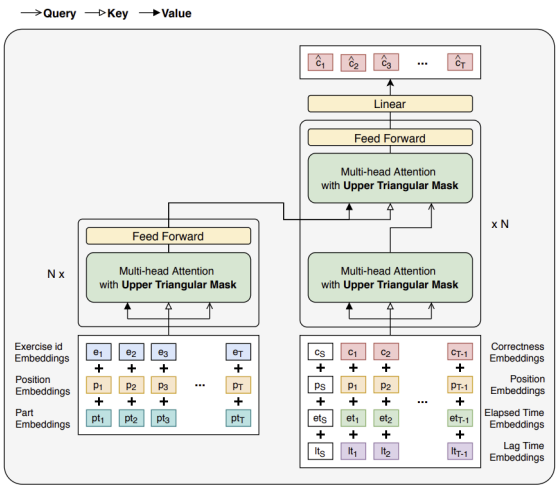


해당 프로젝트에서 구현한 ‘Last Query Transformer RNN for knowledge tracing(이하 LQTR)’은 Transformer의 encoder에서 마지막 입력만 쿼리로 사용하여 시간 복잡도를 줄이는 방식을 채택했다. 이로써 입력 시퀀스를 더 길게 만들었고 Kaggle의 'Riiid! Answer Correctness Prediction' 대회에서 1위를 차지했다..

LQTR의 경우, 이번 DKT 대회의 데이터와 Riiid 대회의 데이터가 유사한 점에서 착안해 이 프로젝트에서도 좋은 성능을 보일 것이라는 가설을 세워 구현하였다. 사용한 데이터는 기존에 제공된 데이터 외에 Feature Engineering을 통해 만든 13개의 feature가 추가로 활용되었다.

기본 parameter 상태에서 LQTR의 Public auroc는 0.7441, accuracy는 0.6855로 나타났다. 기대한 바와는 달리, 좋지 못한 성능을 보였는데, 대회 데이터의 크기가 작아 기존 Riiid 대회의 데이터와 비교했을 때 학습량이 적기 때문으로 분석된다. 이에 하이퍼파라미터 튜닝은 따로 진행하지 않았다.

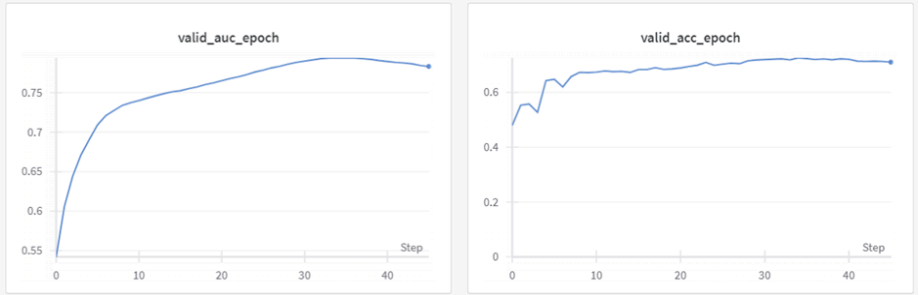
1.4.13 SAINT+



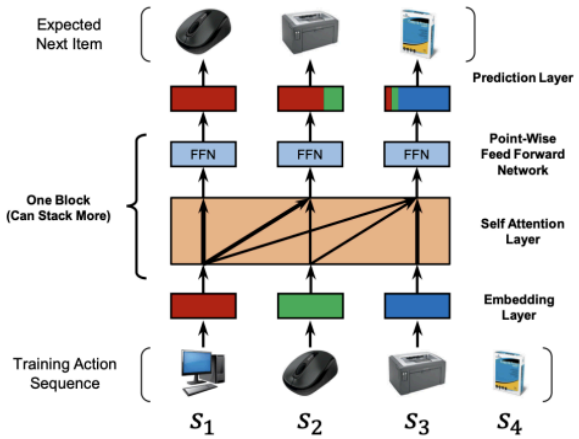
해당 프로젝트에서 구현한 'SAINT+' 모델은 Transformer 기반의 encoder-decoder 구조를 사용한 SAINT 모델로부터 파생된 모델이다. SAINT+는 학생의 지식 상태를 더 정확히 추적하기 위해 문제 푸는 데 걸린 시간과 이전 문제와의 시간 간격을 고려한다. 그러나 우리의 데이터에는 풀기 시작한 시간만 있어 현재 문제와 다음 문제의 풀기 시작한 시간 차를 나타내는 elapsed time feature("time_to_solve")만 사용했다.기본 parameter 상태에서 SAINT+의 Public auroc는 0.7772, accuracy는 0.7339로 나타났다.

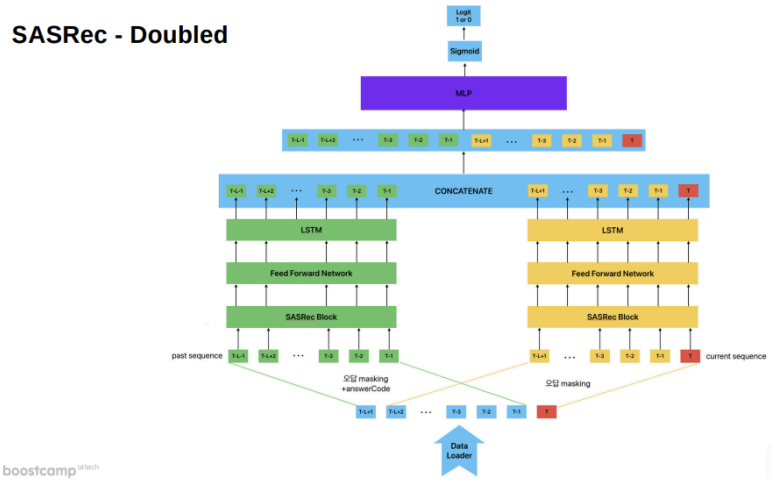
이후 Encoder-decoder 구조를 가진 모델에 적용할 수 있는 T-fixup 기법을 적용하여 모델의 학습을 안정화시키고, GeLU 활성화 함수를 사용하여 그래디언트 소실 문제에 대응했다.

학습을 Wandb를 통해 모니터링한 결과, 35 epoch까지 학습 후 AUC가 최고점을 찍고 다시 감소하는 과적합 현상을 관찰했고, 35 epoch까지 학습된 모델을 사용해 inference를 수행한 결과, Public auroc는 0.7792로 일정 부분 향상되었지만, accuracy는 0.7177로 하락했다. 이는 모델이 특정 클래스나 조건에는 잘 작동하나 다른 경우에는 부정확하게 작동한다는 것을 시사하며, 추가적인 실험이 필요하다고 생각된다.



1.4.14 SASRec (custom)





SASRec 모델을 변형하여 사용자의 정답 및 오답 패턴을 정확하게 파악하고자 했다. 기존 SASRec 모델을 기반으로 하여 self-attention block을 2개 사용하고 미래 정보에 대한 masking을 적용했다. 사용자의 과거 정답 맥락과 현재의 차이에 주목하기 위해 과거 정보와 현재 정보를 분리하여 각각 학습을 진행했다. 정답을 의미하는 correct Embedding의 값을 원래 임베딩에 추가하여 정답인 context에 주목하여 정답 패턴 인식의 정확도를 향상시키도록 했다. 오답인 경우 별도의 masking을 적용하여 모델이 정답과 오답을 구분하는 능력을 강화했다. self-attention 이후에 시간에 따른 정보의 중요도를 모델이 더 잘 파악하도록 LSTM을 적용하여 어떤 정보를 얼마나 기억할지에 대한 학습을 진행했다.

1.4.15 최종 모델 및 앙상블 결과

	Public		Private	
Model	AUROC	ACC	AUROC	ACC
SASRec_v3	0.8314	0.7554	0.8544	0.7849
SASRec_v1	0.8261	0.7688	0.8546	0.7796
SASRec_v2	0.8304	0.7554	0.8581	0.7796
Transformer	0.8056	0.7366	0.8436	0.7715
MF	0.7973	0.8432	0.7177	0.7715
SAINT(-)	0.7772	0.8160	0.7339	0.7312
SASRec + 오답패턴	0.8105	0.7339	0.8407	0.7473
TabNet	0.7720	0.7417	0.7016	0.6989
LSTM	0.7197	0.7406	0.6398	0.6774
Bert	0.6875	0.7506	0.6559	0.6747

	Public		Private	
Model	AUROC	ACC	AUROC	ACC
CatBoost	0.8114	0.7581	0.8017	0.7392
LightGBM	0.8111	0.7446	0.8134	0.7419
XGBoost	0.8055	0.7339	0.8065	0.7392
AutoGluon	0.8064	0.7446	0.8258	0.7500
LightHgcn	0.7861	0.7043	0.8113	0.7312
LightGcn	0.6817	0.5941	0.6250	0.5672
SGCN	0.7644	0.7508	0.6962	0.6935
LightSGCN	Nan	Nan	Nan	Nan

*SASRec_v1: feature selection1 cat emb : num emb = 1 : 1
*SASRec_v2: feature selection2 cat emb : num emb = 1 : 1
*SASRec_v3: feature seleciton2 cat emb: num emb = 1 : 2

boostcamp

단일 모델들에 대한 성능 결과는 위 표와 같다. Public score에서는 SASRec_v3가, Private score에서는 SASRec_v2가 가장 좋은 성능을 보였다. 그 밖의 모델에서는 Catboost, LightGBM, Transformer 등의 순서로 Public score에서 좋은 결과를 보였다.

	Public		Private	
앙상블	AUROC	ACC	AUROC	ACC
SASRec_v3(0.395) + SASRec_v1(0.2325) + Catboost(0.1775) + LGBM(0.065) + HLGCN(0.065) + MF(0.065)	0.8330	0.7581	0.8550	0.7957
SASRec_v3(0.4) + SASRec_v1(0.15) + CatBoost(0.15) + LGBM(0.1) + HLGCN(0.1) + MF(0.1)	0.8327	0.7608	0.8553	0.7984
SASRec_v3(0.5) + SASRec_v2(0.5)	0.8316	0.7473	0.8568	0.7823
SASRec_v3(0.5) + SASRec_v1(0.5)	0.8315	0.7634	0.8564	0.7876
SASRec_v2(0.3) +SASRec_v1(0.3) + CatBoost(0.2) + LGBM(0.2)	0.8307	0.7634	0.8523	0.7957
SASRec_v2(0.3) +SASRec_v1(0.3) + CatBoost(0.4)	0.8306	0.7715	0.8488	0.7849
SASRec_v3(0.3) + SASRec_v1(0.3) + CatBoost(0.25) + LGCN(0.15)	0.8304	0.7581	0.8554	0.7930

단일 모델 성능에 기반해 괄호 안의 비중으로 앙상블을 진행한 결과, ‘SASRec_v3(0.395) + SASRec_v1(0.2325) + Catboost(0.1775) + LGBM(0.065) + HLGCN(0.065) + MF(0.065)’에서 0.8330의 가장 좋은 Public AUROC를, ‘SASRec_v2(0.3) +SASRec_v1(0.3) + CatBoost(0.4)’에서 0.7715의 가장 좋은 Public ACC를 얻을 수 있었다. 이에 이 두 모델을 최종 제출하였고 전자의 모델에서 좋은 Private AUROC를 얻어 DKT 대회를 1등으로 마무리하였다.

1.5 자체 평가 의견

잘했던 점
• 깃허브를 이용한 이슈 기반 협업을 원활히 진행하였다. • 여러 모델들을 사용하였다. • decision tree model에 의존하지 않았다.
시도했으나 잘 되지 않았던 것들
• 그래프 오버피팅을 잡지 못해서, 그래프의 끝을 보지 못하였다.
아쉬웠던 점들
• 강의에 소개된 내용을 모두 적용해보지 못해 아쉬움이 있다. • 제출 시 Description 작성하는 것으로 실험들을 단순하게 정리 하였지만, 제출 수가 많아지고부터 구분하기가 쉽지 않아지면서, 이후 과정에는 노션
프로젝트를 통해 배운 점 또는 시사점
• 템플릿을 반드시 사용하여 실험관리를 조금 더 체계적으로 한다면 더 좋았을 것이다. • 의외로 간단하지만 성능이 좋은 모델들이 많다. • 백지장도 맞들면 낫다(앙상블 학습).

김수진_T6022

1. 학습 목표 달성 방법

- **팀과 개인의 학습 목표**: 프로젝트를 통해 모델 구현에 대한 전반적인 이해, 성능 향상을 위한 다양한 시도, 그리고 협업을 위한 소통 능력을 향상
- **개인 학습 측면의 접근**: 모델 하나만 구현하는 것이 아니라, 전반적인 코드의 연결과 구조를 분석하여 엔지니어링 경험, 원하는 모델을 스스로 구현할 수 있도록 엔지니어링 능력 향상
- **공동 학습의 중요성**: 효율적인 진행을 위해 팀원끼리 분업 및 지속적인 소통으로 협업의 중요성을 경험

2. 모델 개선 방법

- EDA, FE 등으로 데이터 전처리 후 주어진 task 에 맞는 모델 구상, transformer 의 decoder 를 활용하고자 함

- 사용자와 context 간의 상관관계를 보기 위해 decoder layer 까지 사용하는 saint 모델에 현재, 과거 정보를 나눠 각각 학습하도록 구상, encoder 는 SASRec 의 self-attention layer 를 활용하도록 시도함

- Intersample attention 을 구현함으로써 일반화 능력을 향상시켜 교육분야에서 학생들의 전반적인 학습 경향을 학습하도록 시도함
- 정답을 맞췄을 때의 context 를 확실하게 학습할 수 있도록 기존 SASRec 의 모델 구조를 변형함, self-attention layer 와 masking 은 동일하게 들어가나 각각의 위치 정보에 주목하는 것이 아니라 시험, 문제 상황의 context 자체에 주목하게 하기 위해 position embedding 을 제거하고 attention 을 진행한 다음 이후에 추가적으로 LSTM layer 를 넣어 과거 정보에서 얼마나 기억할지를 학습함

3. 달성한 결과와 깨달음

- 기존 Baseline 에서 구현되어 있던 최신정보로부터 max_len 만큼 자른 뒤 position embedding 을 넣고 학습을 했을 때보다 과거와 현재 정보를 나눠 학습을 했을 때 확실한 성능 개선이 있었음. 과거에 정답을 맞췄을 때의 context 와 현재 사용자의 context 가 비슷한 맥락이라면 정답을 맞출 것이고 그렇지 않다면 오답일 것이라는 예측을 수행했다고 분석함

- 다양한 것들을 적용했을 때 이론적으로는 더 정확한 예측을 수행할 수 있겠지만 parameter 수 등이 주어진 인프라에 맞춰 연산 가능한 모델인지 미리 확인하는 습관을 들여야 함(모델이 너무 무거워 OOM 발생으로 학습을 시도하지 못했음)

4.마주한 한계와 아쉬웠던 점

- 하이퍼 파라미터 변동으로는 아주 큰 성능 변화가 일어나지 않았고 모델 구조 자체가 성능에 큰 변화를 미침을 경험
- Data 를 load 하고 내가 원하는 모델에 맞춰 학습할 수 있도록 바꾸는 역량의 중요성, end to end 엔지니어링 역량이 필수임을 깨달음
- 학습이 가능한 효율적인 모델을 구상해야 함

5. 미래 프로젝트를 위한 새로운 시도

- 논문을 더 확실하게 읽어서 내가 해결하고자 하는 task 에 맞춰 논리적인 모델을 구상하고 싶음

- Competition 에 맞춰 접근하는 법을 체화하고 싶음

김창영 T6042

학습 목표

- EDA 및 Feature Engineering 진행 및 깃허브를 활용한 공유
- End-to-End ML pipeline 설계

프로젝트에서 시도한 것들

1. 깃허브를 활용한 EDA 및 FE 공유 시스템 구축, 깃허브 칸반보드를 활용한 일정관리

지난번 프로젝트에서는 EDA 및 전처리, FE 는 개별적으로 이루어졌으며, 별도의 공유 템플릿이 없었다. 이번 프로젝트에서는 깃허브의 Issue 와 PR 템플릿에 EDA, FE 를 추가적으로 만들어서 해당 부분에 대한 원활한 소통할 수 있게 의견을 개진했다.

2. EDA 및 Feature Engineering

시험지별 유저의 시험시간을 바이올린 플롯으로 시각화하여 특이하게 오래 푼 유저가 존재함을 확인했다. 이후 4 분위수를 계산하여 이상치 판단 여부 및 대체에 대해 논의했다. KnowledgeTag 에 대한 분산분석을 진행하여 태그별 평균 정답률의 유의미한 차이를 확인했다. 더 나아가 차원을 줄이기 위해 binning 하여 분산분석 후 평균값으로 대체하여 파생변수를 생성하였다. 유의미한 결과로 이어지지않았다. 원래 분포와 binning 후 분포가 달라지면서 데이터가 왜곡되는 문제가 있었다.

3. ML pipeline 설계 및 모델 학습

지난 프로젝트때 베이스라인이 기본적으로 주어지기 때문에 과연 내가 end-to-end 로 파이프라인을 직접 설계할 수 있을까? 의문이 들었다. 따라서 정형데이터(시계열 포함 x)에 대한 학습 파이프라인을 end-to-end 로 직접 설계해보았다. Tree 기반 모델과 TabNet 을 클래스로 모듈화하였으며, 커스텀 콜백을 구현해 WandB 에 logging 되게끔 했다. 일반화 성능 개선을 위해 Out Of Fold 를 구현으며, 해당 방법을 통해 이전대비 성능 개선을 이루어냈다. 또한 WandB sweep 을 통해 하이퍼파라미터 튜닝을 진행했다.

느낀점

지난번 프로젝트에서 아쉬웠던 부분들을 개선해서 의미있었다. 다만 각 종 1 등 솔루션에서 TabNet 에 대해 언급해서 Tree 기반 모델과 앙상블하면 좋은 성능이 나올거라 생각했다. 하지만 파라미터도 바꾸도 피쳐도 바꿔서 돌렸는데 성능이 기대한 만큼 나오지 않았다.(~~💩밟았다..~~) 해당 부분이 아쉽다

한계/교훈을 바탕으로 다음 프로젝트에서 시도해 볼 내용

EDA 시도는 좋았으며 유의미한 성능개선까지는 이루어지지 않았다. 다음 프로젝트에서는 EDA->FE or 전처리->모델링->성능개선까지 도달해보고 싶다.

박승아 T6059

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

저번 Book Rating Prediction 프로젝트와는 달리, 이번 DKT 프로젝트에서는 기존 베이스라인 모델의 간단한 변형을 넘어 새로운 모델을 구현해보고 우리의 코드에 맞게 도입하는 것을 목표로 설정하였다. 이에 Last Query Transformer RNN, SAINT 모델에 대해 학습한 후, 우리의 코드에 바로 적용할 수 있도록 코드를 디버깅하며 리팩토링하였다. 또한, 프로젝트에 더욱 집중하고자 관련 강의를 이전보다 빠르게 수강한 후, 더 오랜 기간 고민하며 T-fixup 등 새로 도입할 수 있는 내용들을 구현해보았다.

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

이전에는 팀원들과의 코드 충돌을 우려하여 local 에서 주로 작업을 하였다. 그러나 이번 프로젝트에서는 팀원들과 함께 깃허브 이슈 기반 개발을 새롭게 시도하였다. 원격 저장소를 이용하면서 VSCode 를 통해 Git 을 복제하고 브랜치를 내어 협업하는 것에 익숙해지면서 보다 적극적으로 실험할 수 있었다. 더불어, 깃허브 칸반보드 등을 통해 내가 수행할 내용을 명시적으로 잘 공유할 수 있었고 수정한 코드를 보다 수월히 피드백 받을 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

머신러닝, 딥러닝 모델에 관한 논문을 읽는 것에는 조금이나마 익숙해졌지만 이를 구현한 다른 이들의 코드를 함께 확인하고 이해하는 것은 아직 익숙하지 않다. 따라서 논문에 대한 이론적 지식은 알고 있어도 이를 구현하고자 할 때, 임베딩 차원 변형, 모듈화 등에서 어려움을 마주하였다. 이에 구현하는 데 마음이 급했고 이를 어떻게 변형하며 더 좋은 성능을 만들지에 대한 고민을 많이 못하였던 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

논문의 구조를 공부하면서 함께 공유되고 있는 코드를 함께 확인하는 연습을 하며 어느 부분에 어떤 코드가 들어갔는지 분석하는 시도를 해보고 싶다. 이를 바탕으로 나의 아이디어를 추가로 반영해 베이스라인 모델보다 보다 좋은 성능을 내는 모델을 구현해보고 싶다.

전민서 T6151

프로젝트 과정 속의 나의 학습 목표

- Transformer 기반의 모델을 공부하여 이를 직접 적용해보자.
- 그래프 모델을 이용해서 유의미한 성능 향상을 이끌어내자.
- 지난 프로젝트에서는 제대로 활용하지 못한 깃헙을 적극적으로 활용해보자.
- 데이터를 가공해 성능을 끌어올리는 것 보다, 모델 자체를 변경하는 방향으로 학습하자.

학습 목표를 이루기 위해 노력한 점들

- Baseline 코드를 처음부터 완벽하게 이해하기
- 이해한 베이스라인 코드를 바탕으로 새로운 코드를 처음부터 짜기
- gcn 모델을 이해하려고 노력함
- 트랜스포머 모델을 적용하기 위해 이해하는데 노력함

그 결과 내가 배운 것들 + 변화한 점

- 논문을 기반으로 사용 할 수 있는 내용인지 아닌지 판단 후 구현하는 능력
- 논문의 experiments 정보를 바탕으로, 유의미한 논문인지 파악하는 능력
- 프로젝트의 진행에 있어서 원활한 의사소통이 기반이 되어야 한다는 점을 알았다.
- 여러가지의 모델에서 주장하는 내용을 섞어 더 나은 모델을 만들 수 있다는 점을 알았다(graph+ transformer)
- 전혀 관계가 없어보이는 두가지 도메인의 아이디어를 뽑아 한가지로 섞어낼 수 있다.
- 모델을 만들며 부족한 부분을 다른 논문의 아이디어를 가져와 차용 할 수 있다.
- LightGcn -> SGCN -> LightSGCN -> Lightgcn double 순으로 약점을 보완하는 방향으로 모델을 수정한 점이 마음에 들었다.

마주친 한계

- 아직까지 원활하게 생각을 코드로 표현하는 것에 부족함이 있다.
- 논문을 읽는 능력이 부족한 듯 하여, 시간적인 압박이 있다.
- 내 의사를 한줄로 표현하여 의사소통 하는 것에 문제가 있다.

다음 프로젝트에서 시도해볼 것

- 존재하는 논문들을 많이 읽은 뒤, 아이디어를 뽑아 내 sota 모델을 만들어 보고 싶다.
- 전혀 다른 도메인의 내용을 가져와 융합해서 recsys에서 사용 해 보기
- 성능뿐만 아니라, 경량화도 잡기

Deep Knowledge Tracing Wrap Up Report
한대희_T6179

나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

- 주어진 데이터와 Task 를 시간 순서에 상관없이 마지막 유저의 정답 여부만을 예측하는 정형 데이터 분류 문제 접근 방식으로 처리해 보고자 하였습니다. 이를 위해 LightGBM 을 베이스로 하여 부스팅 트리 모델인 CatBoost 와 XGBoost 모델 베이스라인을 데이터와 task 에 맞게 개발하였습니다.

- AWS 에서 개발한 AutoML 오픈 소스 프레임 워크 AutoGluon 을 도입하여 복잡한 전처리 과정, 알고리즘 선택, 모델 학습 및 평가, 하이퍼 파라미터 튜닝 과정을 단순화하고 자동화했습니다. 이를 통해 모델 향상을 위한 시간을 절약할 수 있었습니다.

- 협업 능력을 향상하기 위해 GitHub 을 적극적으로 활용했습니다. GitHub Issue 생성, Issue 를 기반으로 브랜치 생성 및 작업, PR(Pull Request) 요청, 코드 리뷰 및 병합 과정을 통해 팀원들과 원활한 협업을 이끌어냈습니다. 이를 통해 효율적인 협업과 프로젝트의 성과를 높였습니다.

나는 어떤 방식으로 모델을 개선했는가?

- EDA 및 feature engineering 을 통해 팀 내에서 공통적으로 사용 가능한 유의미한 피처들을 생성하였습니다. 팀원들의 모델에도 바로 적용할 수 있도록 이를 모듈화하고 공유하여 팀 전체적인 성능을 매우 향상시켰습니다. 특히 도메인 task 에 적합한 상대적 유저 학습 수준 피처, 시계열 특성을 이용한 유저 학습 수준 피처들은 문제 해결에 큰 도움을 주었습니다.

- GridSearchCV 를 통해 교차 검증(Cross-Validation)과 하이퍼 파라미터 튜닝을 모두 자동화하여 모델의 성능을 신뢰성 있게 평가하면서 최적의 하이퍼 파라미터 조합을 찾아냈습니다. 모델에 적합한 하이퍼 파리미터들을 적용하여 모델의 성능을 향상시켰습니다.

- 모델 개발 및 최적화를 더욱 효율적으로 수행하기 위해 Wandb 를 적극적으로 활용했습니다. Wandb 를 이용하여 실험을 체계적으로 추적하고, 실험 결과를 시각화하여 모델의 성능 및 변화를 명확하게 관찰할 수 있었습니다.

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

- 최종 제출 기간 전까지 모델 성능을 향상시키기 위해 유의미한 피처를 생성하고 선택하는 과정을 반복적으로 수행하면서, 최적의 피처들을 도출해냈습니다. 이러한 노력 덕분에 최종 공개 점수(public score) 및 비공개 점수(private score)에서 모두 1 등을 차지할 수 있었습니다. 이 경험을 통해 피처 엔지니어링의 중요성과 영향력을 명확히 깨달았습니다.

- 프로젝트에서 제가 수행한 CatBoost 모델과 LightGBM 모델들을 앙상블하여 최종 1 등을 차지하는데 큰 기여하였습니다. 이러한 경험을 통해 제가 개발한 모델이 프로젝트 전체의 성공에 기여할 수 있다는 것을 깨달았으며, 프로젝트에서의 팀 협력과 모델 개발 능력이 중요하다는 것을 더욱 명확히 이해하게 되었습니다.

- AutoML 프레임워크 AutoGluon 을 활용하여 머신 러닝 프로젝트에서 모델 개발과 최적화를 효율적으로 수행하였습니다. 결과적으로 auroc 점수를 0.8 이상 달성하였습니다. 이 경험을 통해 앞으로 프로젝트를 진행할 때 시간과 리소스를 절약하면서도 더 나은 결과를 얻을 수 있는 방법을 습득하였습니다.

마주한 한계 혹은 아쉬웠던 점은 무엇이며, 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- RNN, LSTM, Transformer 와 같은 시계열 모델들을 사용해 예측하는 실험을 진행하면서 Sequential approach 에 대한 이해를 높일 수 있었으나, 아쉽게도 auroc 기준 0.8 이상의 점수를 달성하지 못한 것이 아쉬웠습니다. 앞으로 딥러닝 기본기와 모델링 기술에 대한 이해도를 더욱 향상시키고, 높은 성능을 달성할 수 있도록 지속적인 노력을 기울일 것입니다

- Feature engineering 을 통해 다양한 피처들을 생성 후, 실험적으로 피처 선택하는 과정에서 시간이 많이 소비되었습니다. 이를 극복하고 효율적인 피처 선택을 위해 VarianceThreshold 와 같은 피처 선택 기법들을 다음 프로젝트에서 도입하여 모델을 개발하고 최적화하는 데 소요되는 시간과 리소스를 최소화 해보겠습니다.

- 모델 평가를 위한 CSV 파일 제출 시 Description 작성만으로 실험들을 정리했더니, 제출 수가 늘어나면서 실험을 구분하기가 어려워졌습니다. 이를 극복하기 위해 다음 프로젝트부터는 Notion 템플릿과 같은 실험 관리 도구를 반드시 사용하여 실험을 더 체계적으로 관리하고 문서화할 것입니다.

한예본 T6181

[프로젝트 과정 속의 학습 목표]

- Transformer 기반의 모델을 공부하여 이를 직접 구현해보자.
- 그래프 모델을 공부해보자.
- 나의 전공인 수학교육과 관련되어 있는 주제이니, 이를 활용할 수 있는 방법을 모색하자.
- 지난 프로젝트에서는 제대로 활용하지 못했던 깃헙을 적극적으로 활용해보자.

[학습 목표를 이루기 위해 노력한 점들]

- SASRec 변형 모델 제작

주어진 데이터가 sequential 데이터라는 점에 주목하여 transformer와 lstm을 적극적으로 활용하고자 하였고, 이에 나온 아이디어가 SASRec 모델을 활용하는 것이었다. 이에 팀원들과 SASRec 논문 스터디를 진행한 후, 이를 바탕으로 SASRec block에 positional embedding을 제거하고 그 결과 나온 임베딩 벡터들을 lstm에 태우는 새로운 SASRec 구조를 고안하여 변형 모델을 제작하였다. 또한, '정답 패턴만을 학습하도록 설계된 새 모델에서, 오답 패턴 또한 학습하면 어떨까?'라는 아이디어와 같이 새롭게 만들어 낸 모델을 조금씩 수정하여 몇 가지 가설을 세워보고 그 결과를 실제로 확인해 보았다.

- 그래프 모델 공부

그래프 모델을 잘 알고 있는 팀원의 도움을 받아 그래프 모델의 구조를 공부하고, heterogenous LightGCN 모델을 만드는 데 도움을 줄 수 있었다. 기본 그래프 모델에 대해 공부한 뒤, '단순히 사용자와 아이템 간의 관계에 대해서만 그래프를 그리는 것이 아니라, 문항/시험지/태그와 같은 feature와 사용자간의 관계를 각각 그래프로 만들어서 모델로 활용할 수 없지 않을까?'라는 질문을 던졌을 때, 실제로 이것이 구현 가능하다는 팀원의 이야기와 이를 HLGCN으로 구현해볼 수 있어서 신기함을 느꼈다.

- IRT 기반 feature engineering

프로젝트의 데이터가 '수학 문제'에 대한 내용으로, 전공 지식을 살려보고 싶다는 생각이 들었고, 이에 교육평가에서 활용하는 IRT 이론을 가져와 피쳐 엔지니어링을 시도했다. 그 결과, IRT의 문항과 관련된 모수를 parameter로 tree 모델에 사용하였을 때, 눈에 띄는 성능 향상을 볼 수 있었다.

- 깃허브 이슈 기반 협업

팀원들과 깃허브 활용에 대한 규칙을 정하고, 이를 지키면서 깃 사용에 익숙해질 수 있었다. 특히, 이슈를 생성하고, 칸반 보드를 활용하여 자신이 진행 상황과 앞으로의 계획을 공유하면서 더욱 효율적으로 프로젝트를 진행할 수 있었다.

[프로젝트를 통해 배운 점과 변화한 점]

- 다양한 모델 공부

이번 프로젝트를 통해 Transformer, Graph, Tree 등의 다양한 모델을 공부하고 구현해 보면서 성장할 수 있었다. 특히, 구현 능력이 많이 좋아진 것을 느꼈다.

- 협업을 통해 일의 효율과 동시의 나의 능력을 키울 수 있다는 점

깃허브를 통한 협업을 통해 프로젝트 진행 속도를 높인 것뿐만 아니라, 팀원들과 함께 모델을 공부하고, 구현하면서 혼자서 공부할 때보다 훨씬 빠르게 인공지능 지식을 습득할 수 있었다. 특히, 인공지능 초심자인 내가 이번에 이렇게까지 프로젝트를 할 수 있었던 것은 팀원들의 도움이 컸던 것 같다.

[프로젝트 중 마주친 한계]

- 베이스라인 코드를 공부하고 이를 변형하는 것은 익숙해졌으나, 바닥부터 코딩하는 것을 어려워한다.
- EDA를 바탕으로 가설을 세우고 모델을 만들어 검정하는 체계적인 과정이 부족했던 것 같다.

[다음 프로젝트에서 시도해볼 것]

- 모델 하나를 잡고 데이터 분석부터 시작하여 가설을 기반으로 한 모델링, 구현 후 검정까지 시도해보고 싶다.
- 깃허브 브랜치를 더욱 잘 활용하여 내가 시도한 것들이 잘 정리 될 수 있도록 해야겠다.
- 매일의 실험일지를 잘 작성하여 나중의 회고에 더 도움이 될 수 있게 정리해두어야겠다.