Level 2 DKT Wrap-Up Report

RecSys_04 파이팅해야조

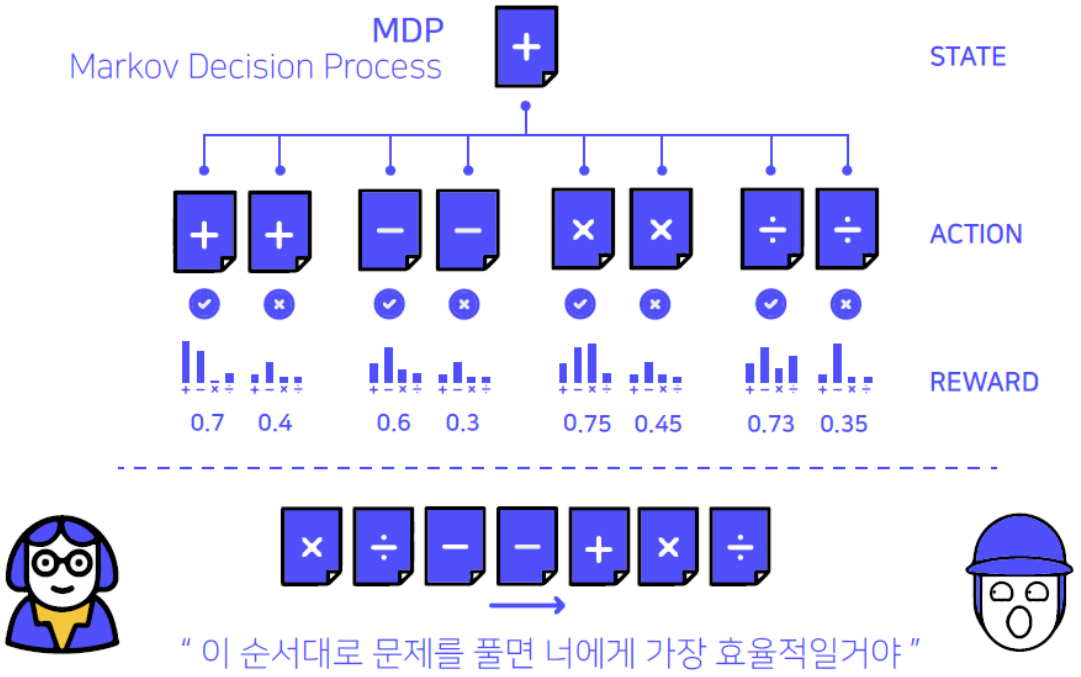
김세훈 김시윤 문찬우 배건우 이승준

▼ 1. 프로젝트 요약

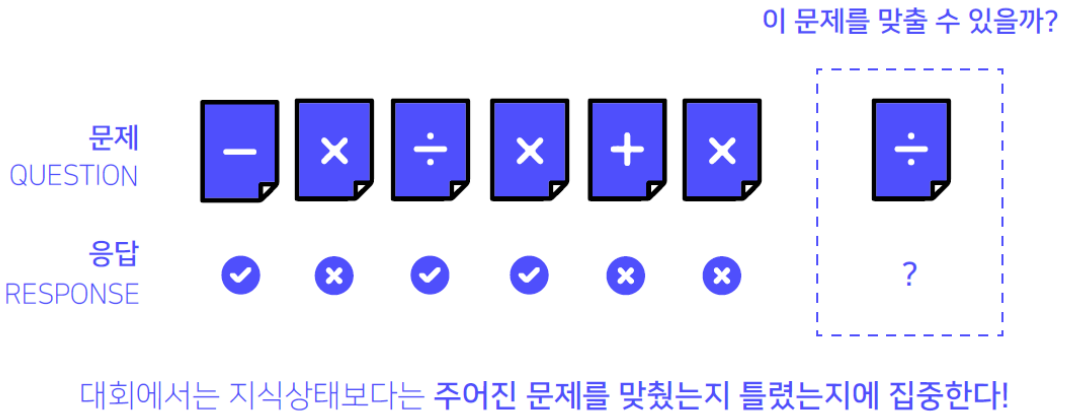
본 프로젝트는 사용자의 지식상태를 추적하기 위해 시험지의 메타 데이터를 활용하여 사용자의 마지막 문제에 대한 정답 여부를 확인하는 것에 목적이 있다. 데이터의 특성을 반영하기 위해 Boosting 계열 모델인 LightGBM, Catboost에대한 실험을 진행하였다. 추가적으로 주어진 데이터의 시간적 특성을 반영하기 위해 RNN, LSTM-ATTN,GRU-ATTN,Bert,Last-Query,GRU+Saint,T-Fixup과 같은 모델을 사용하였다. 마지막으로 사용자와 시험지의 상호작용을 확인하기 위해 lightgcn 모델을 사용하였다. 이후 이 모델들의 장점을 취합하기 위해 앙상블하여 최종적인 결과를 도출하였다. 본 프로젝트의 결과 (LightGBM1 : GRU+Saint : Last-Query : LSTM-ATTN : GRU-ATTN : LightGBM2를 0.65 : 0.075 : 0.075 : 0.0625 : 0.0625 : 0.075)의 비율로 앙상블한 모델의 auroc는 0.8109로 가장 성능이 높았으며 최종적으로 이를 제출하였다.

▼ 2. 프로젝트 개요

교육과정에서 시험의 한계를 극복하고자 하는 새로운 접근 방식인 Deep Knowledge Tracing (DKT)에 중점을 두고 있습니다. DKT는 학생들의 지식 상태를 추적하고 이해도를 측정하는 딥러닝 기반의 방법론으로, 개별 학생들에게 맞춤형 피드백을 제공하고 향후 문제 해결 능력을 예측하는 데 사용됩니다. 이 프로젝트의 목적은 각 학생의 수학 이해도와 약한 부분을 파악하여 개인별로 적합한 문제를 추천함으로써, 보다 효율적인 학습 방향을 제시하는 것입니다.



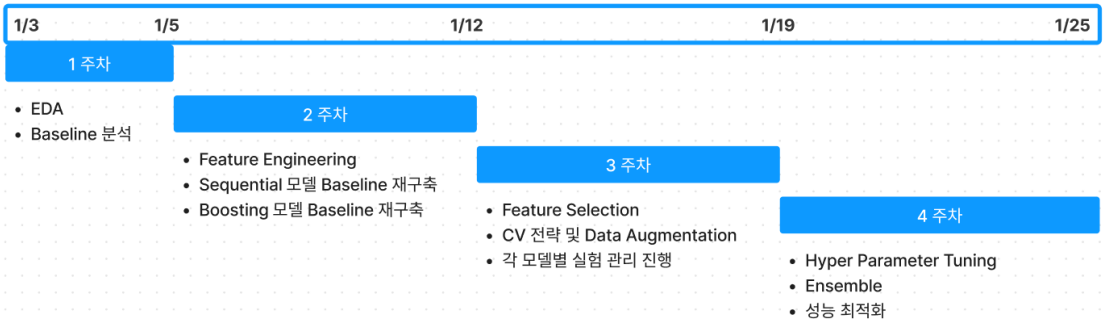
본 대회에서는 'Iscream' 데이터셋을 사용하여 DKT 모델을 구축하고, 이를 통해 주어진 문제를 학생들이 맞출지 틀릴지를 예측하는 것입니다.



약 7000명의 사용자들의 문제 풀이 결과 중에서 최종적으로 테스트 데이터 사용자들의 마지막 문제에 대한 정답 여부를 예측하는 것이 목표입니다. 이를 통해 개인화된 교육의 미래를 탐색하는 것이 이 프로젝트의 핵심입니다.

▼ 3. 프로젝트 수행 절차 및 방법

▼ 3-1. 프로젝트 일정 및 타임라인

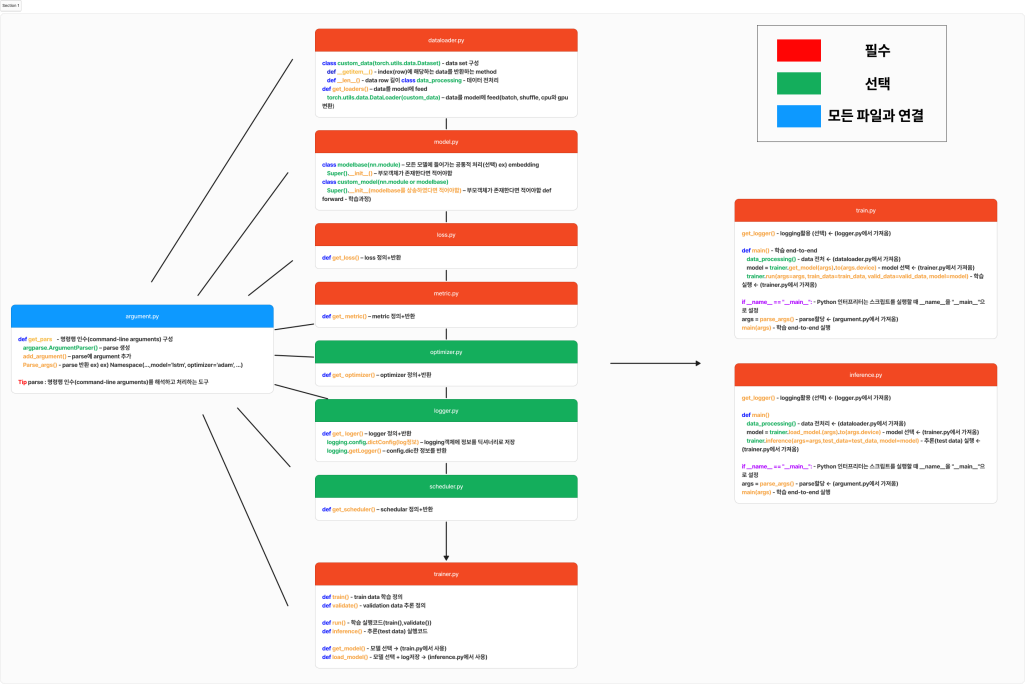


▼ 3-2. 서버 구성 및 환경

개발 환경

- 버전 정보 : Python 3.10.13
- 패키지 정보

```
pandas==20.3
scikit-learn==1.3.2
tqdm==4.51.0
wandb==0.16.2
transformers==4.36.2
pytorch==1.12.1
torchvision==0.13.1
torchaudio==0.12.1
cudatoolkit==11.3
```



▼ 3-3. 프로젝트 팀 구성 및 역할

공통	**EDA, 데이터 전처리, Feature Engineering, Hyper Parameter Tuning**
김세훈	Boosting모델 베이스라인 구축, T-Fixup모델 구현, K-Fold 적용, Data Augmentation 적용
김시윤	LGBM 베이스라인 구축 및 최적화, lightgcn 모델링, Ensemble 진행
문찬우	Lastquery 모델링, Rnn, Gru, Tcn, 등 Sequence 모델링
배건우	Base environment 구축, Base pipeline 구축, Sweep 구현, Stacking ensemble 구현
이승준	Saint, Saint + GRU, GRUATTN 모델링

▼ 3-4. 협업 전략

- **notion**
  - 프로젝트 일정과 역할 분담 정리 및 ideation을 기록하여 공유
- **github**
  - 발생한 이슈나 생각난 아이디어를 github 이슈를 통해 공유하고 개인 branch를 통해 이를 해결한 내용을 pull request를 활용하여 공유함
  - github flow를 활용하여 코드 공유 및 관리를 효율적으로 도입하고자 함
- **slack**
  - 회의 중에 실험해보거나 참고해볼 reference 등을 공유

▼ 3-5. 프로젝트 파이프라인

▼ (1) EDA

- 사용자가 한 문항을 풀었을 때 그 문항을 맞췄는지에 대한 정보가 담겨져 있습니다. 데이터는 모두 Timestamp 기준으로 정렬되어 있습니다.

iScream_Data	
userID	integer
assesmentItemID	string
testId	string
answerCode	integer
Timestamp	timestamp
KnowledgeTag	integer

userID	assessmentItemID	testId	answerCode	Timestamp	KnowledgeTag
0	A060001001	A060000001	1	2020-03-24 00:17:11	7224
0	A060001002	A060000001	1	2020-03-24 00:17:14	7225
0	A060001003	A060000001	1	2020-03-24 00:17:22	7225
1	A040013001	A040000013	1	2020-01-06 08:40:43	2048
1	A040013002	A040000013	1	2020-01-06 08:43:46	2048
1	A040013003	A040000013	1	2020-01-06 08:44:29	2047
2	A030050001	A030000050	1	2020-01-10 11:02:53	407
2	A030050002	A030000050	1	2020-01-10 11:03:45	407
2	A030050003	A030000050	1	2020-01-10 11:04:20	407

`userID` : 사용자 별 고유번사로 총 7,422명의 사용자 데이터가 존재합니다.

`assessmentItemID` **:** 문항의 고유번호이며, 총 9,454개의 고유 문항이 있습니다.

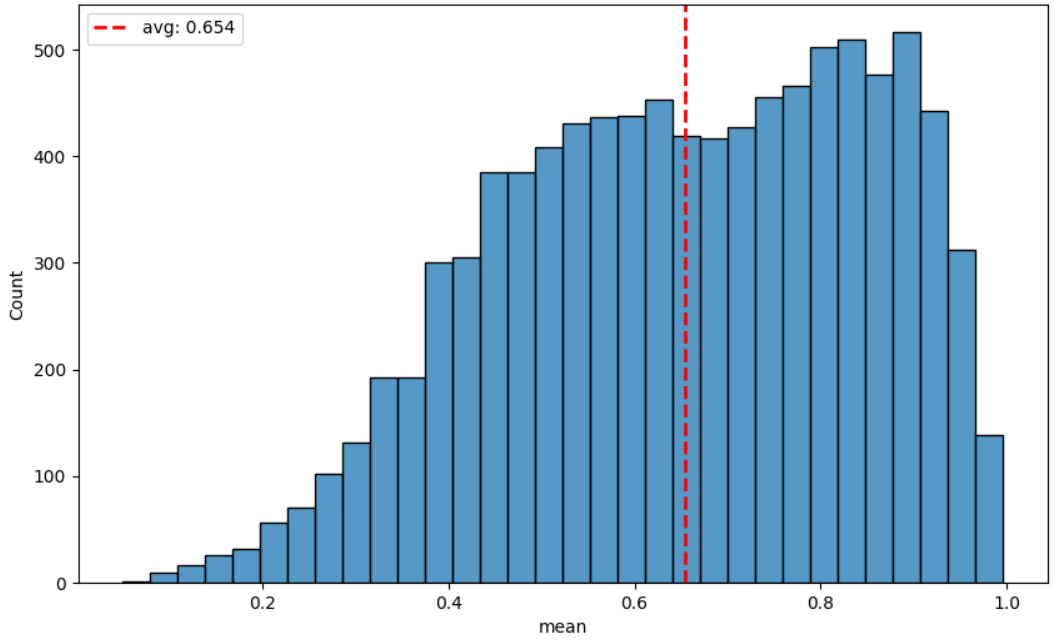
`testID` **:** 시험지의 고유번호이며, 총 1,537개의 고유한 시험지가 있습니다.

- 시험지 규칙 : 첫 자리는 A, 첫 자리 3개 + 마지막 3개 시험지 번호, 앞의 3자리에서 가운데만 1~9

`answerCode` **:** 사용자가 해당 문항을 맞췄는지 여부이며, 0은 틀릿 것, 1은 맞춘 것입니다. test 데이터의 경우 마지막 시퀀스의 answerCode가 -1로 예측해야 할 값입니다.

`Timestamp` **:** 사용자가 해당문항을 풀기 시작한 시점의 데이터입니다.

`KnowleadgeTag` **:** 문항 당 하나씩 배정되는 태그로, 일종의 중분류 역할을 합니다. 912개의 고유 태그가 존재합니다. 정답률은 약 65%로 나타났다.



- test의 경우 마지막 시퀀스의 answerCode가 -1로 구성

▼ (2) Feature engineering

수치 데이터

- 정답률과 관련하여 각 난이도를 수치로 표현할 수 있을 것이라 판단
  - **[Feature]** Feature별(userID, assessmentItemID, testID, KnowleadgeTag) 정답에 대한 평균/중간값/표준편차
  - **[Feature]** User ID 및 Knowledgetag, testId, assesmentID의 고유 개수를 확인

누적 데이터

- Sequential한 데이터를 가장 잘 표현할 수 있는 데이터일 것으로 판단
  - **[Feature]** 문제를 풀어나갈수록 Feature별(userID, assessmentItemID, testID, KnowleadgeTag) 정답률
  - **[Feature]** UserID별 testID 별 누적 푼 문제수, 누적 정답 수, 누적 정답률
  - **[Feature]** UserID별 Knowledgetag 별 누적 푼 문제수, 누적 정답 수, 누적 정답률
- 유저가 문제를 풀어감에 따라 학습효과로 인해 실력이 상승할 것으로 예상
  - **[Feature]** 유저별 정답여부에 따른 정답/오답 누적합

    - **[Feature]** 유저별 태그의 정답여부 누적합
- 사람들이 많이 푼 문제일수록 정답률이 높아질 것이라고 예상
    - **[Feature]** assessmentItemID과 유저 그룹에 따른 정답 및 정답률의 누적합
    - **[Feature]** testId와 유저 그룹에 따른 정답 및 정답률의 누적합
    - **[Feature]** assessmentItemID과 유저 그룹에 따른 문제 푼 시간
    - **[Feature]** testId와 유저 그룹에 따른 문제 푼 시간

상대적 데이터

- 사용자끼리 문제가 맞고 틀림이 상대적으로 점수를 매길 수 있으면 난이도 예측에 도움이 될 것이라 판단
    - **[Feature]** 정답률을 고려한 상대적 점수 계산

시간 관련 데이터

- Timestamp 컬럼을 활용하기 위해 문제 푸는데 걸린 시간을 수치화하여 Sequential한 데이터를 표현할 수 있을 것이라 판단
    - **[Feature]** 문제 푸는데 걸린 시간에 따른 초단위 및 누적 시간
    - **[Feature]** userID별 문제 푸는데 사용한 시간 정규분포화
    - **[Feature]** tag별 문제의 정답/오답에 따른 풀이 평균 시간 및 시험지 및 문제 번호 그룹 별 정답/오답에 대한 문제 풀이 평균 시간
    - **[Feature]** test_ID별, knowledgetag별 시간정보 및 문제 푸는데 사용한 시간 정규화
    - **[Feature]** 미래, 과거의 특정 시점에 유저별 정답 여부
    - **[Feature]** 문제를 푸는 시간대 별 특정 문제, 유저 정답률

▼ (3) 모델

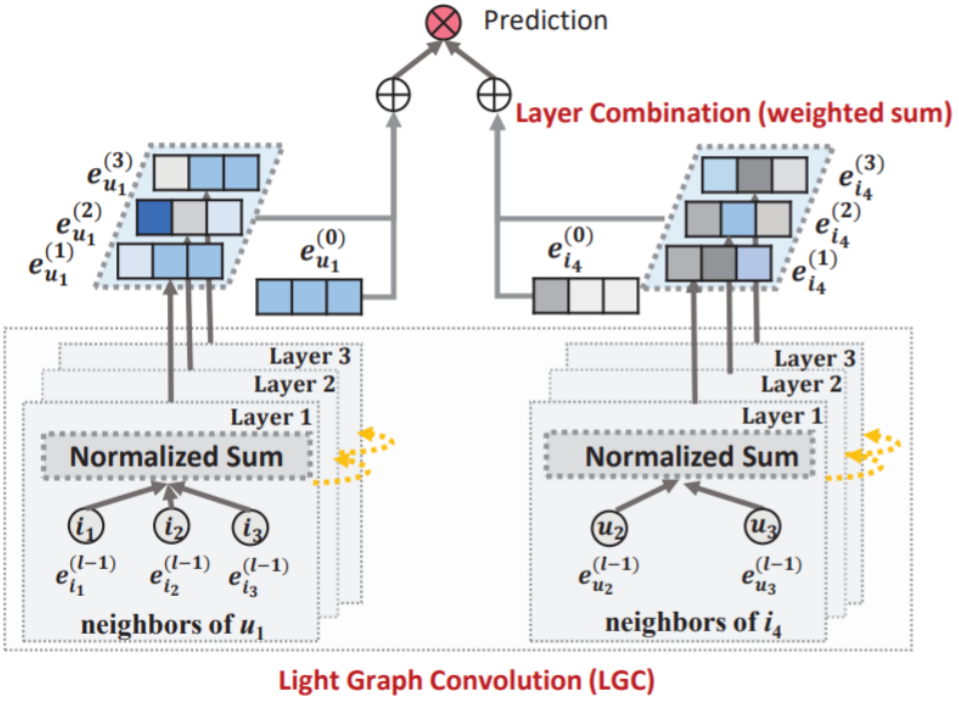
▼ 사용한 모델 요약

Model	description (AUC / ACC)
LightGCN	(0.6791 / 0.6237)
LightGBM	(0.7954 / 0.7312)
CatBoost	(0.7049 / 0.7043)
RNN	(0.741 / 0.6846) (Only Private)
LSTM	(0.7381 / 0.6720)

LSTMATTN	(0.7636 / 0.7222)
GRU	(0.7535 / 0.6828)
GRUATTN	(0.7422 / 0.6667)
Bert	(0.7099 / 0.6371)
LastQuery	(0.7554 / 0.7177)
Saint	(0.7601 / 0.6855)
GRU + Saint	(0.7274 / 0.6828)

▼ Graph

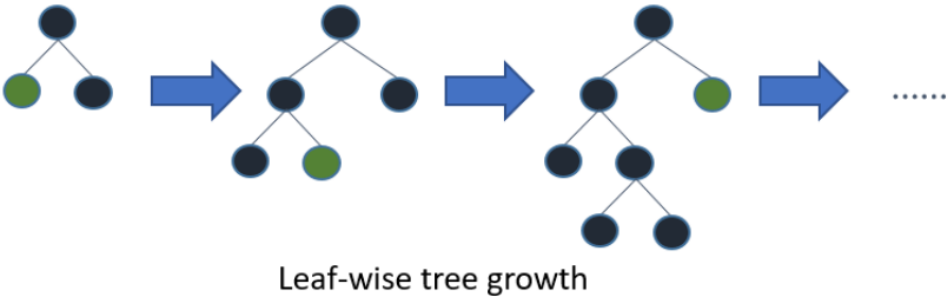
- **LightGCN**



- 사용자와 시험지간의 상호작용 관계를 예측하는 모델로 userID, assessmentiteID및 문제를 푼 시간인 elapsed, Tag에 따른 정답률인 KnowledgeTag_percent, 문제를 푼 횟수의 누적합인 cumulative, 시험지의 난이도를 파악하기 위한 문제지의 번호인 paper_number를 추가한 결과 baseline인 0.6791에서 0.5019로 auc가 감소하였다. lightgcn은 유저-아이템 쌍의 관계를 확인하기 위해 만들어진 모델로 concat하지 않은 다양한 feature의 상호작용이 모델의 복잡성을 증가시키고 개별 아이템 간의 상호작용이 제대로 이루어지지 않기 때문에 감소하는 것으로 해석 가능하다.

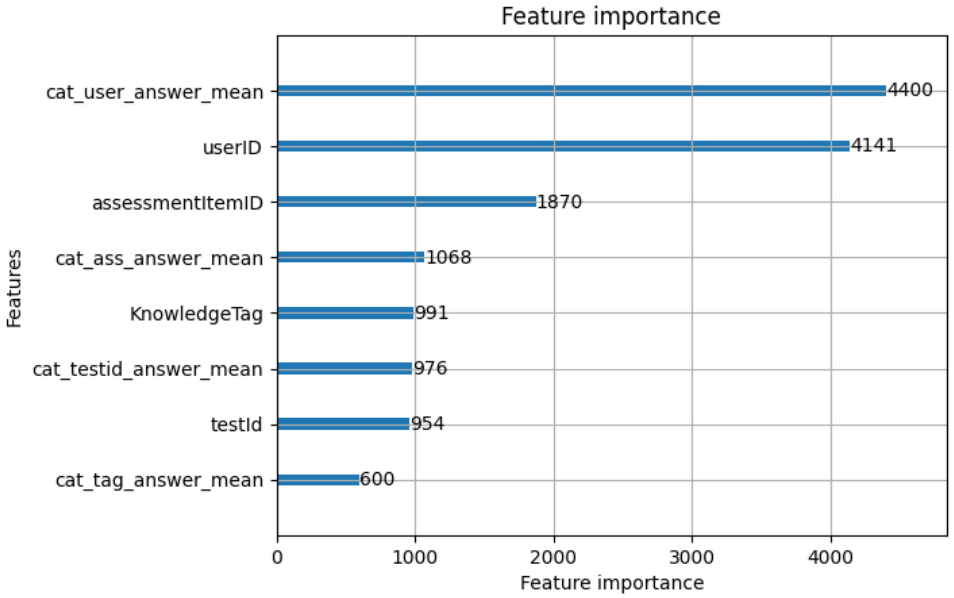
▼ Boosting

▼ LightGBM

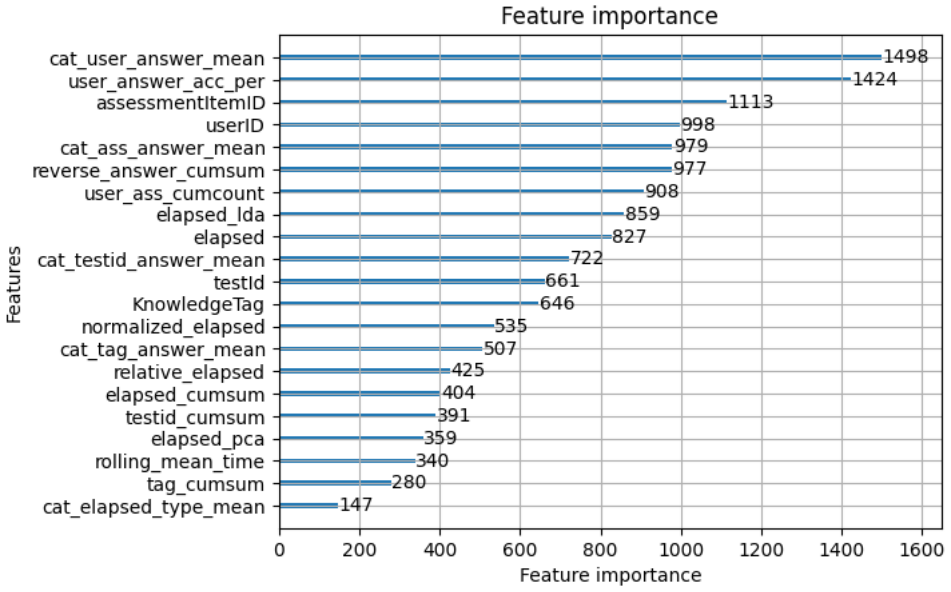


lightgbm1

- LightGBM은 Tree기반 부스팅 모델 중 leaf-wise model로 트리가 깊어지면서 소요되는 시간과 메모리를 절약하는 모델이다. 이러한 특성으로 인해 빠른속도로 학습을 하면서, 손실을 최소화 할 수 있기 때문에 본 모델을 사용하였다.
- Tree 기반 학습 알고리즘으로 예측 결과는 데이터의 종류와 특성에 따라 달라지기 때문에 Feature Engineering을 통해 유의미한 정보를 반영하면 모델이 더 잘 예측할 것임을 초점에 두고 실험을 진행하였다.
- userID, assessmentItemID, testId, KnowledgeTag를 이용하여 예측한 결과의 AUC 및 ACC는 각각 0.6568,0.6644 으로 나타났다. 추가적으로 사용자별, 문항별, 시험지별, 태그별 정답률 등 정답을 기준으로 하는 Feature를 추가하여 예측한 결과 AUC 및 ACC는 0.7893, 0.7396으로 향상됨을 보였다. Feature 중요도를 확인한 결과 사용자,문항,문제,태그 기준 순으로 나타났다.



- 또한 추가적인 Feature Engineering을 진행하여 사용자 및 문항이 LGBM에서 가장 유의미한 예측을 나타내는 Feature임을 확인할 수 있었다. 최종적인 AUC와 ACC는 각각 0.8396, 0.7855로 나타났다.



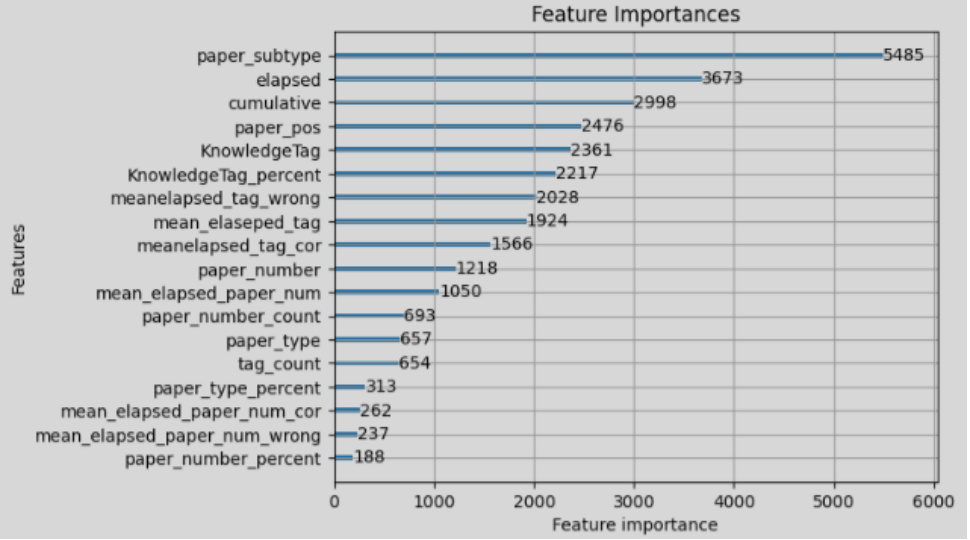
- Feature Engineering이 sequential하지 않은 모델에서 성능에 크게 영향을 미친다는 것을 확인할 수 있었으며 특히 본 데이터에서는 사용자와 문항에 대한 Feature의 중요도가 높음을 확인하였다.

lightgbm2

- lightgbm1에서 사용한 feature 이외에 다양한 시각에서 데이터를 바라보기 위해 추가적으로 Boosting을 진행하였다.

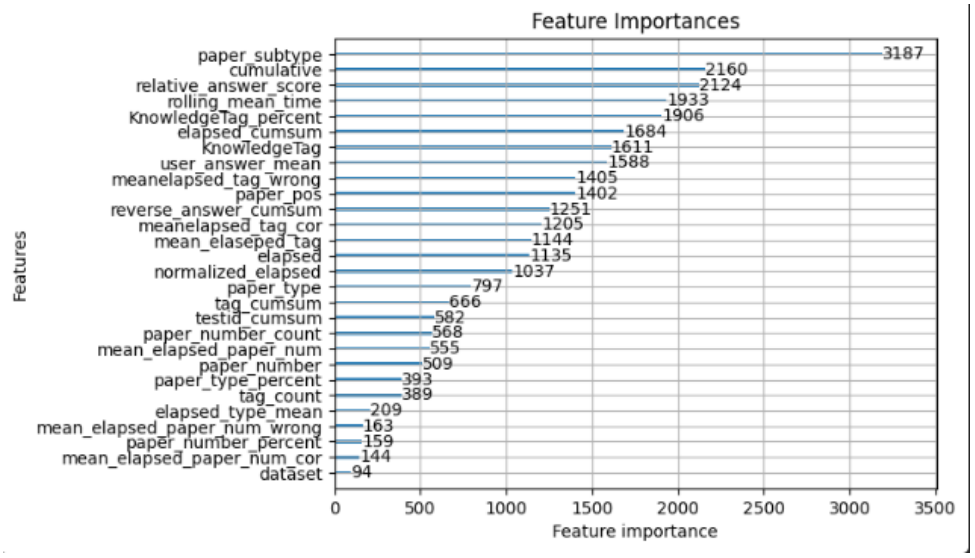
- EDA 결과 생성된 문제를 푼 시간인 elased와 그에 따른 파생변수, 시험지 번호를 나타내는 paper_number, KnowledgeTag에 따른 정답률 등을 feature로 사용하여 부스팅을 진행한 결과 private에서 auc와 acc는 각각 0.81/0.77 로 나타났다.

```
FEATS = [
       'KnowledgeTag', 'elapsed', 'cumulative', 'paper_number',
       'mean_elaseped_tag', 'meanelapsed_tag_cor', 'meanelapsed_tag_wrong',
       'mean_elapsed_paper_num', 'mean_elapsed_paper_num_cor',
       'mean_elapsed_paper_num_wrong', 'paper_type',
       'paper_subtype', 'paper_number_percent', 'paper_type_percent',
       'KnowledgeTag_percent', 'paper_number_count', 'tag_count', 'paper_pos']
```



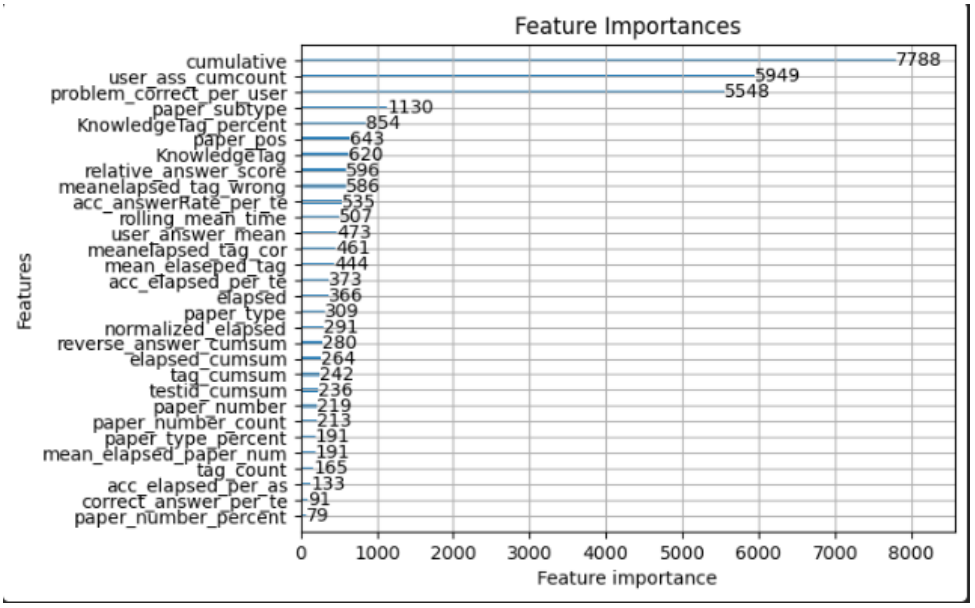
- 추가적인 eda 분석 결과를 바탕으로 elapsed의 파생변수와 누적합의 파생변수를 추가적으로 생성하였으며, 각 유저별 정답률에 대한 추가적인 변수를 생성하여 부스팅을 진행하였다. 그 결과, 유저의 누적 정답률를 나타내는 user_answer_acc_per에서 과적합이 발생함을 확인, 최종적으로 제거한 결과의 auc와 acc는 각각 0.83, 0.78로 나타났다. 사용된 변수와 feature importance는 다음과 같다.

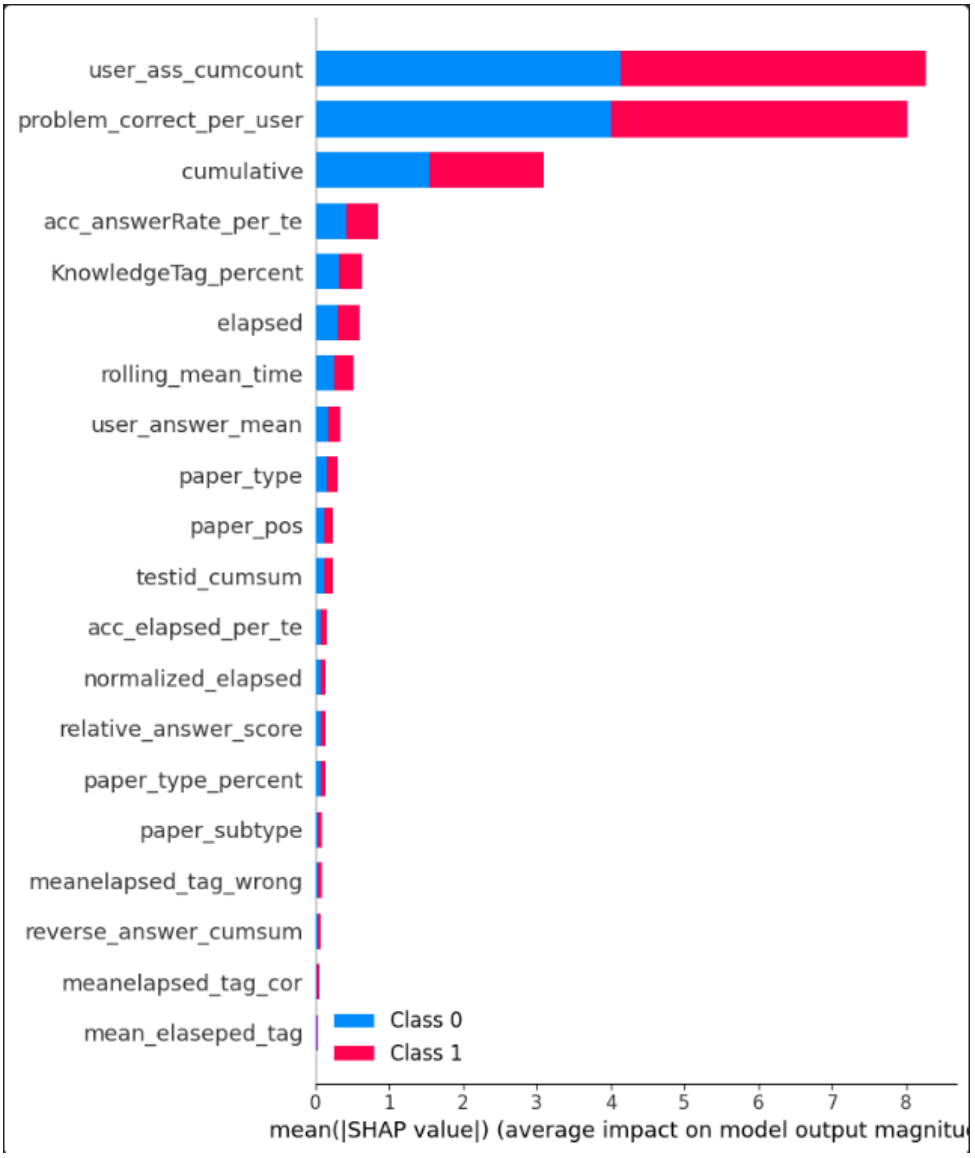
```
FEATS2 = ['KnowledgeTag', 'dataset', 'elapsed', 'cumulative', 'paper_number',
       'mean_elaseped_tag', 'meanelapsed_tag_cor', 'meanelapsed_tag_wrong',
       'mean_elapsed_paper_num', 'mean_elapsed_paper_num_cor',
       'mean_elapsed_paper_num_wrong', 'paper_type',
       'paper_subtype', 'paper_number_percent', 'paper_type_percent',
       'KnowledgeTag_percent', 'paper_number_count', 'tag_count', 'paper_pos',
    'user_answer_mean',
       'reverse_answer_cumsum', 'testid_cumsum', 'tag_cumsum',
       'relative_answer_score', 'elapsed_cumsum', 'normalized_elapsed',
       'elapsed_type_mean', 'rolling_mean_time']
```



- 최종적으로 유저-tag, 고유번호, test에 따른 누적합등을 포함하여 부스팅을 진행하였다. 첫 부스팅 결과 유저와 정답여부에 따른 정답률인 cum_answerRate_per_user로 인해 과적합이 발생함을 확인하여 제거하였다. 그 결과 private에서의 auc및 acc는 각각 0.87, 0.81로 나타났다. 사용된 변수와 feature importance, 그리고 shap value는 다음과 같이 나타났다.

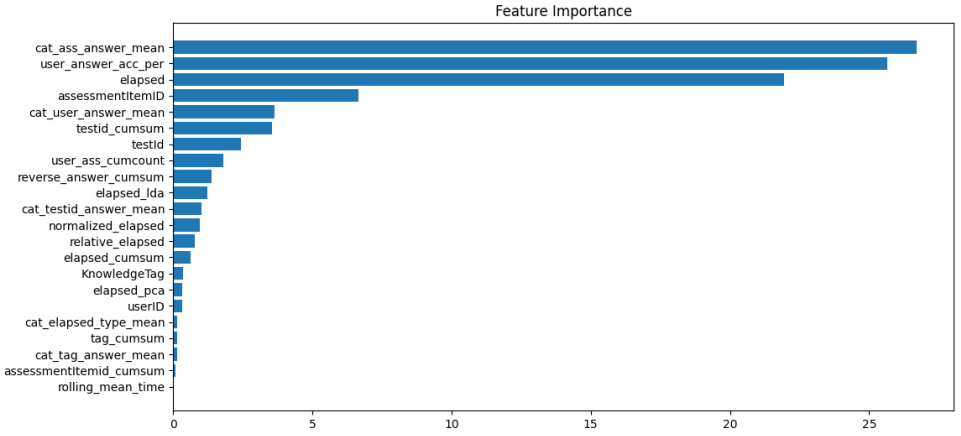
```
FEATS3 = ['KnowledgeTag', 'dataset', 'elapsed', 'cumulative', 'paper_number',
       'mean_elaseped_tag', 'meanelapsed_tag_cor', 'meanelapsed_tag_wrong',
       'mean_elapsed_paper_num', 'mean_elapsed_paper_num_cor',
       'mean_elapsed_paper_num_wrong', 'paper_type',
       'paper_subtype', 'paper_number_percent', 'paper_type_percent',
       'KnowledgeTag_percent', 'paper_number_count', 'tag_count', 'paper_pos',
        'user_answer_mean', 'user_ass_cumcount',
       'reverse_answer_cumsum', 'testid_cumsum', 'tag_cumsum',
       'relative_answer_score', 'elapsed_cumsum', 'normalized_elapsed',
       'elapsed_type_mean', 'rolling_mean_time', 'problem_correct_per_user',
       'problem_solved_per_user', 'cum_answerRate_per_user',
       'acc_tag_count_per_user', 'correct_answer_per_as',
       'correct_answer_per_te', 'acc_count_per_as', 'acc_count_per_te',
       'acc_answerRate_per_as', 'acc_answerRate_per_te', 'acc_elapsed_per_as',
       'acc_elapsed_per_te']
```


Feature Importances
cumulative
user_ass_cumcount
problem_correct_per_user
paper_subtype
KnowledgeTag_percent
paper_pos
KnowledgeTag
relative_answer_score
meanelapsed_tag_wrong
acc_answerRate_per_te
rolling_mean_time
user_answer_mean
meanelapsed_tag_cor
mean_elaseped_tag
acc_elapsed_per_te
elapsed
paper_type
normalized_elapsed
reverse_answer_cumsum
elapsed_cumsum
tag_cumsum
testid_cumsum
paper_number
paper_number_count
paper_type_percent
mean_elapsed_paper_num
tag_count
acc_elapsed_per_as
correct_answer_per_te
paper_number_percent
7788
5949
5548
1130
854
643
620
596
586
535
507
473
461
444
373
366
309
291
280
264
242
236
219
213
191
191
165
133
91
79
Features
0
1000
2000
3000
4000
5000
6000
7000
8000
Feature importance


user_ass_cumcount
problem_correct_per_user
cumulative
acc_answerRate_per_te
KnowledgeTag_percent
elapsed
rolling_mean_time
user_answer_mean
paper_type
paper_pos
testid_cumsum
acc_elapsed_per_te
normalized_elapsed
relative_answer_score
paper_type_percent
paper_subtype
meanelapsed_tag_wrong
reverse_answer_cumsum
meanelapsed_tag_cor
mean_elaseped_tag
Class 0
Class 1
0
1
2
3
4
5
6
7
8
mean(|SHAP value|) (average impact on model output magnitu

▼ CatBoost

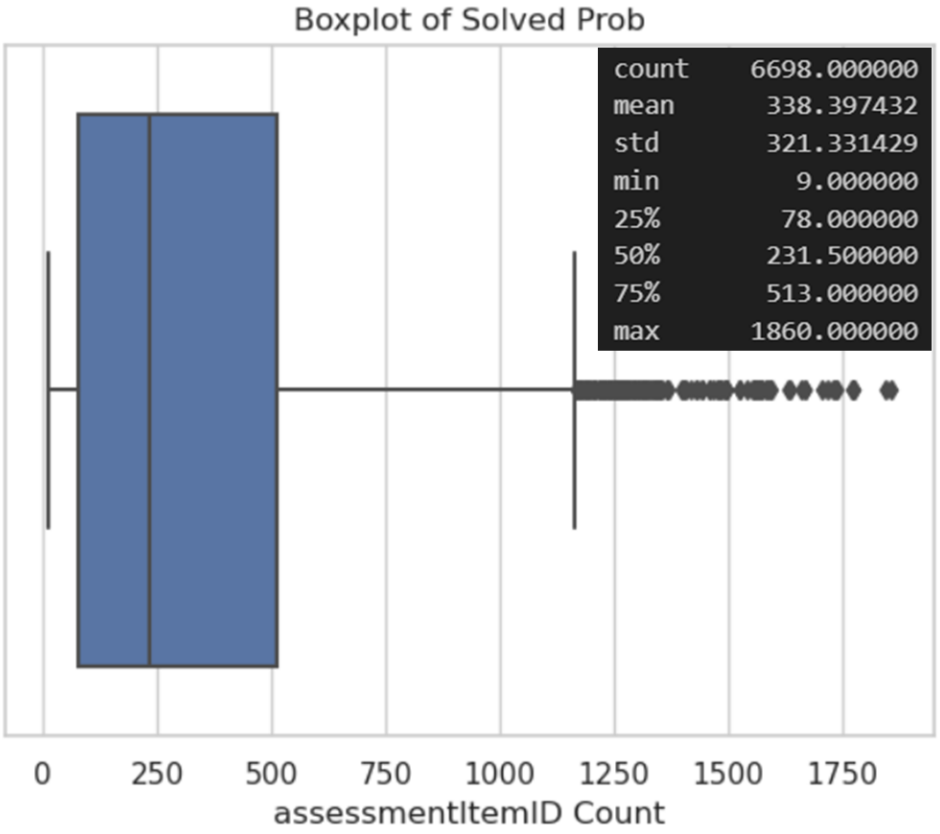
- 기본 카테고리 Feature인 문항, 시험지, 태그 기준으로 확인한 결과 AUC 및 ACC 는 0.6478, 0.6604으로 나타났으며 최종적으로 추가한 변수를 사용하여 학습한 결과 0.7049, 0.7043으로 성능이 향상되었다.



- 위 데이터의 특성을 고려하여 categorical 데이터를 바탕으로 결과를 예측함에 있어 categorical 문항의 정답의 평균, 유저의 정답률에 대한 평균, 문제를 푸는데 소요한 시간등이 중요한 요소임을 파악할 수 있었다.

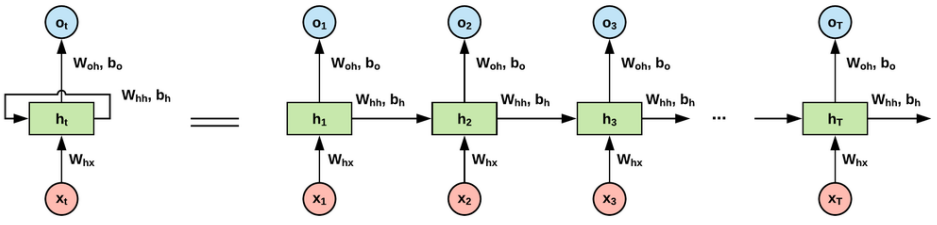
▼ Sequence

▼ About Sequence



- 시퀀스 데이터에서 사용하고자 하는 유저별 시퀀스의 길이의 평균값은 약 338로, 단순한 수치 만으로 시퀀스의 길이가 짧은 것인지 긴 것인지 판단이 불가능하다고 생각했다. 따라서 시계열 데이터를 처리하는 여러 모델들을 추가하여 DKT 데이터에 대한 모델의 성능을 먼저 파악하고자 하였다.

▼ RNN



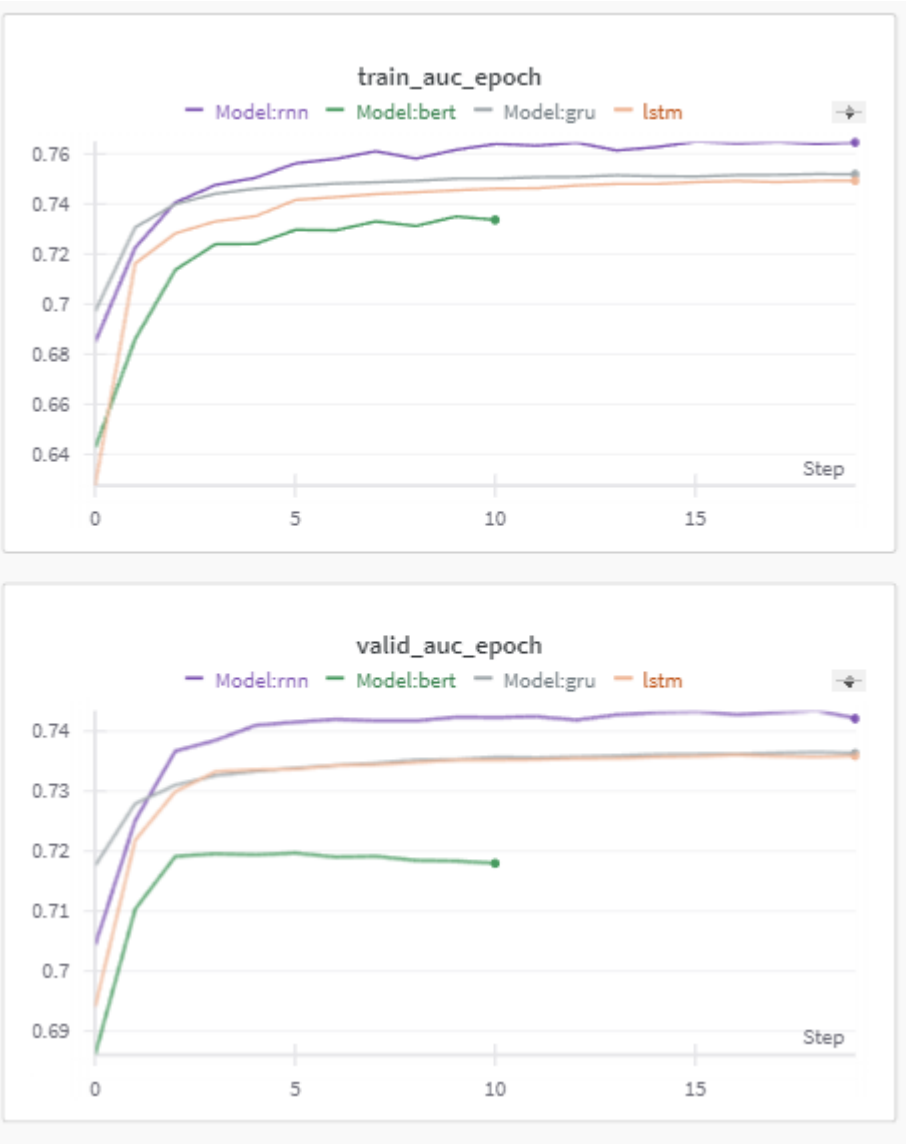
모델을 선택한 이유

- rnn은 시계열 데이터를 처리하는 가장 기본적인 모델로, DKT 모델이 단기적인 의존성 혹은 간단한 패턴을 갖고 있는 경우 성능이 높게 나올 수 있을 것이라 판단하여 이를 베이스라인 코드에 추가하여 성능을 확인하였다.

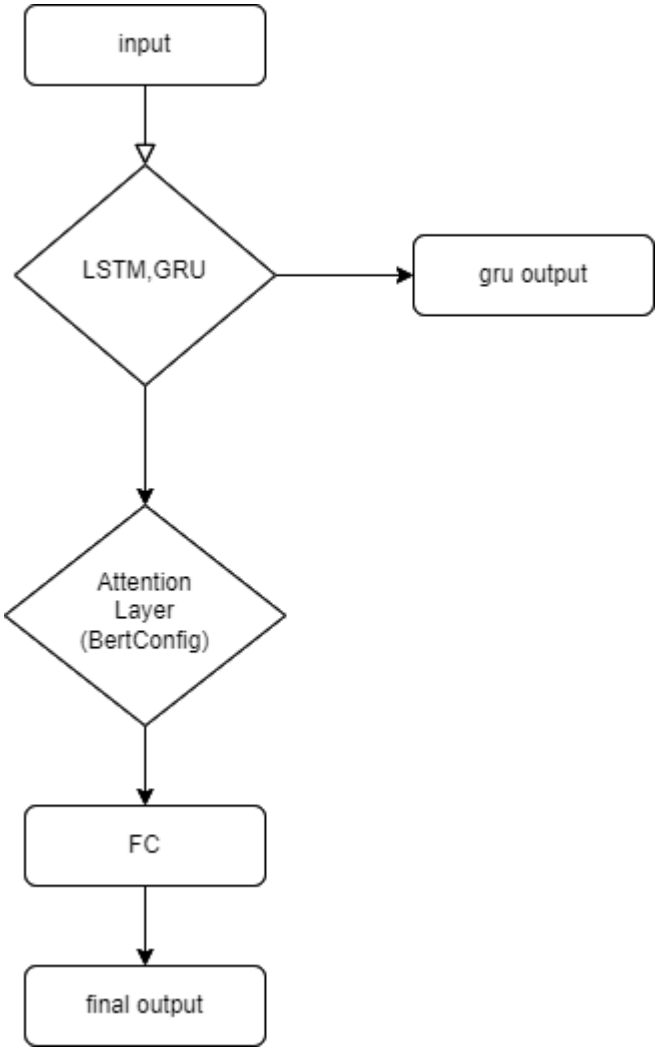
모델 실험을 위한 가설 설정 및 실험 방법

- 기준이 되는 모델인 만큼, 특별한 feature 추가 및 모델의 구조를 크게 변경하지 않고, lstm, bert, gru 등의 시퀀스 모델과의 성능 비교를 우선으로 하였다.

모델 실험 결과

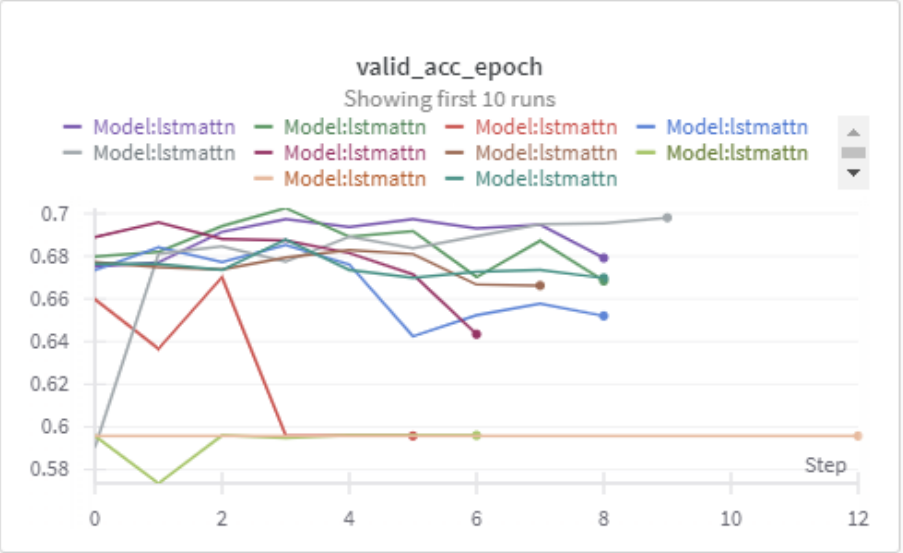


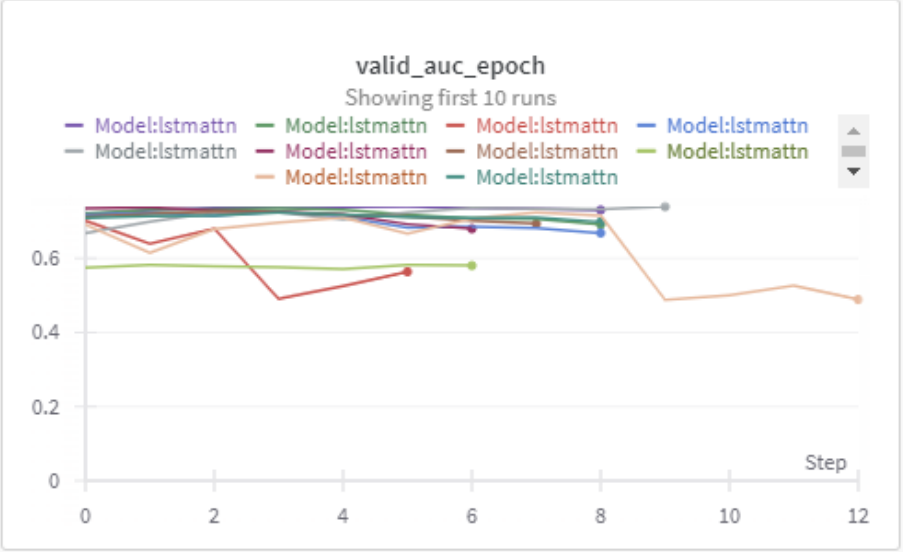
▼ 어텐션 기반 모델들



LSTMATTN

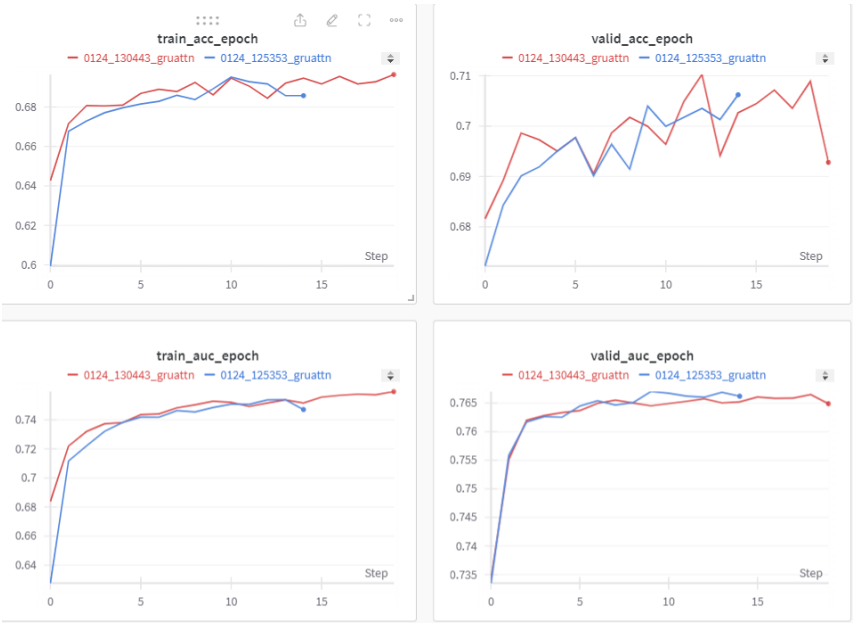
- Bert 모델은 양방향으로 sequence data를 확인하는 특징이 있는데, 이번 프로젝트의 마지막 문제에 대한 예측을 하기 위해서는 단방향 sequence에 대한 정보가 필요하다. 따라서 LSTM의 sequence data의 순차적 정보를 반영하는 특징을 갖는점을 활용하여 LSTM을 통해 단방향 sequence에 대한 정보를 반영한 data를 Bert 모델에 결합한 LSTMATTN 모델을 구현하였다.





GRUATTN

- gru는 LSTM의 단점인 누적에 의한 학습 저하 및 높은 계산 복잡성을 해결하기 위한 모델로 LSTMATTN의 모델의 성능 향상을 확인 후 추가적으로 실험을 진행했다.
- LGBM의 feature importance와 shap value를 고려하여 continuous feature를 추가한 결과 0.7662 → 0.7649로 오히려 성능이 낮아지는 것을 확인하였다.



최종적으로 LSTM, GRU 모두 Bert 모델과 결합했을 때 유의미한 성능 향상을 확인할 수 있었다.

Model	Best AUC / ACC
LSTM	(0.7381 / 0.6720)
LSTMATTN	(0.7636 / 0.7222)
GRU	(0.7535 / 0.6828)
GRUATTN	(0.7697 / 0.6968)

▼ Last Query

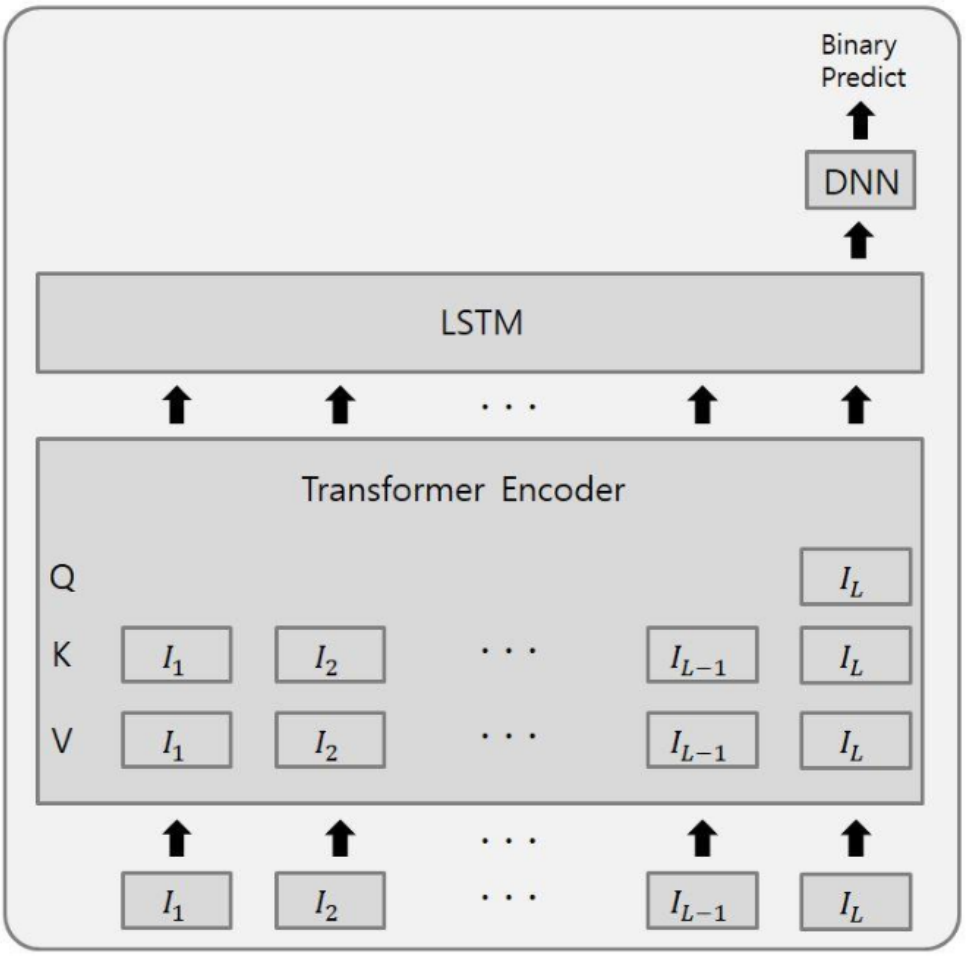
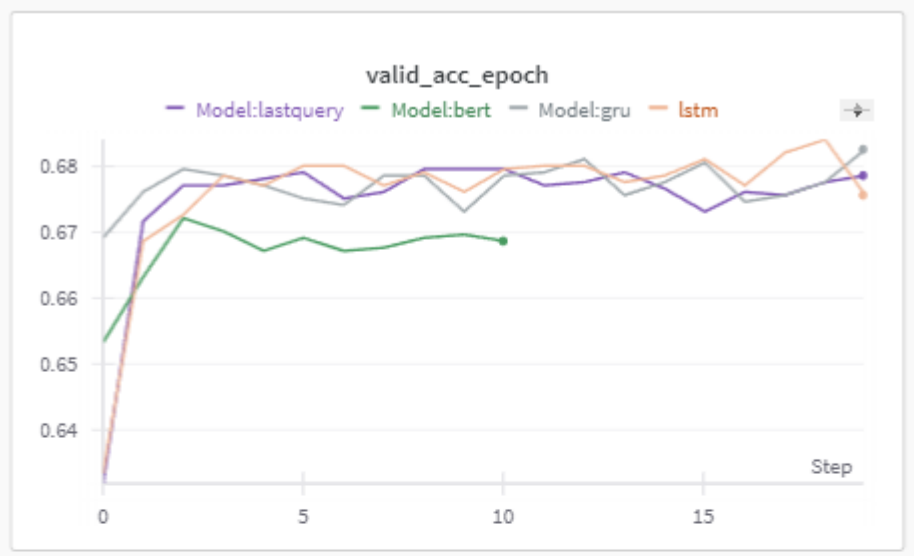
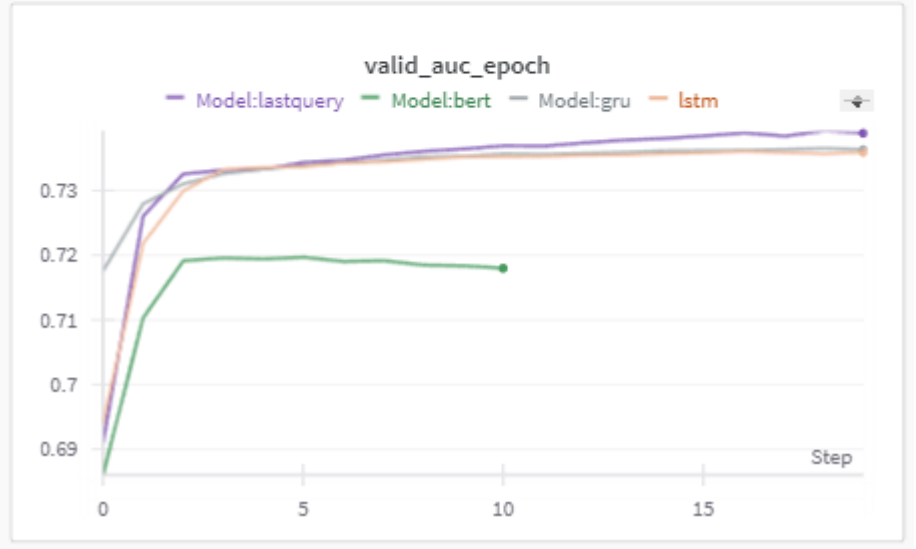


Figure 1. Model Structure

모델을 선택한 이유

- Last query 모델은 기존의 RNN이나 LSTM 모델보다 장기 의존성에 대한 문제가 적다고 판단 하였고, 학생이 마지막으로 접한 문제에 대한 지식 상태를 직접 반영 하기 때문에, Sequence에서 가장 최신을 학습시켰다.



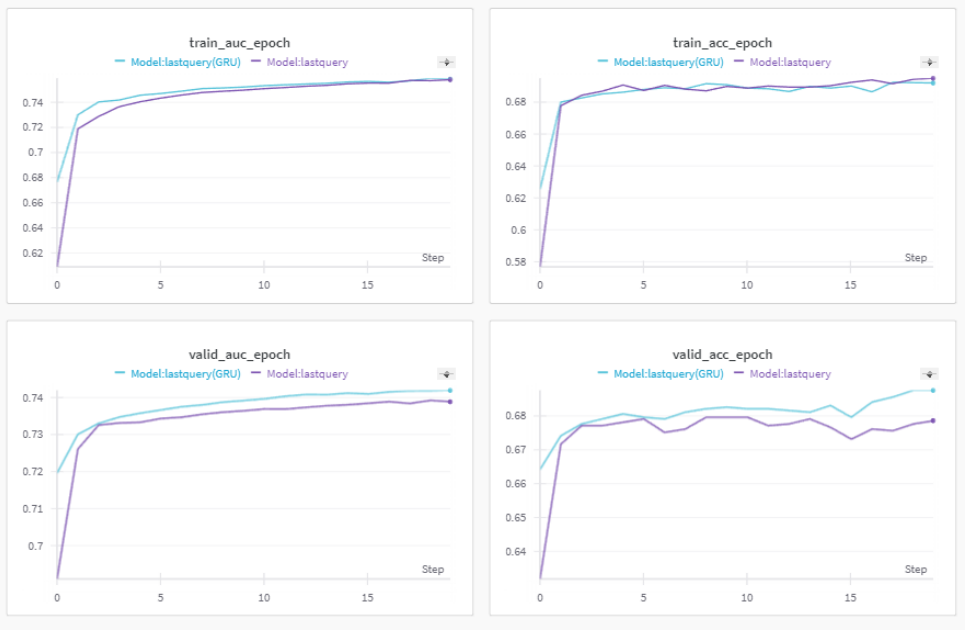


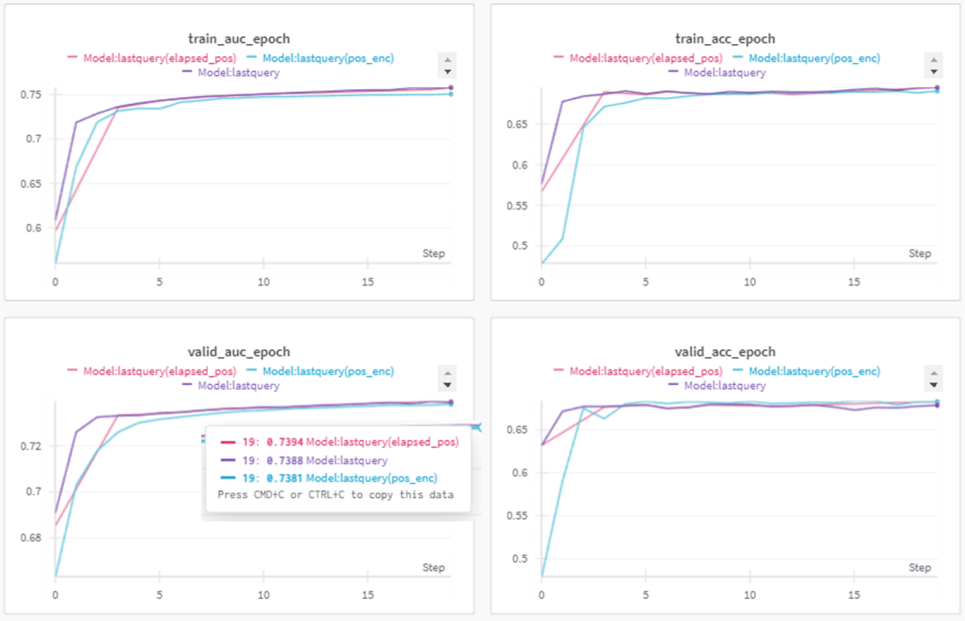
▼ Last Query(GRU + elapsed positional encoding)

모델을 선택한 이유

- Last query 모델은 기존의 RNN이나 LSTM 모델보다 장기 의존성에 대한 문제가 적다고 판단 하였고, 학생이 마지막으로 접한 문제에 대한 지식 상태를 직접 반영 하기 때문에, Sequence에서 가장 최신의 학습

모델 실험을 위한 가설 설정 및 실험 방법



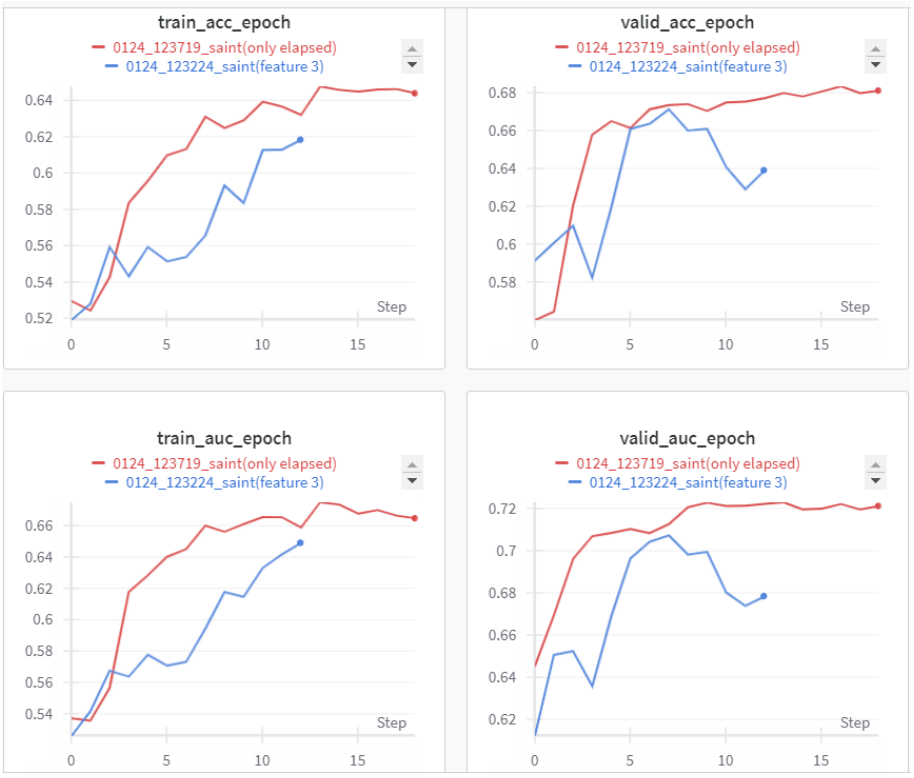


모델 실험 결과

▼ Saint

모델 실험 결과

- 논문에서 사용한 feature(assessmentItemID, testId, KnowledgeTag, elapsed)만 사용한 모델과 LGBM의 feature importance와 shap value를 고려하여 continuous feature를 3개 추가해봤을 때의 모델을 비교해봤을 때 0.7212/0.6811에서 0.6785/0.6391으로 feature를 추가한 모델이 오히려 성능이 낮아지는 것을 확인하였다.



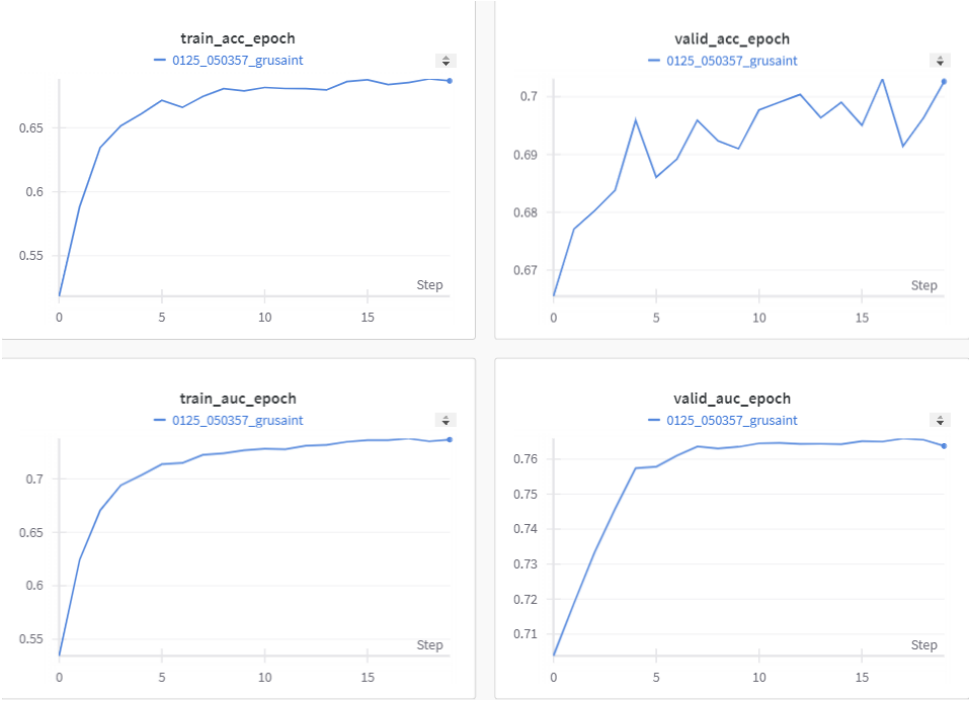
결과 해석

- Saint 구조가 기본적으로 transformer이기 때문에 양방향 시퀀스를 고려하는 모델이지만, 해당 문제는 마지막 문제를 예측하는 문제이므로 단방향 시퀀스를 고려할 수 밖에 없다. 따라서 순차적인 정보를 반영하는 RNN계열의 모델보다 성능이 낮은 원인으로 해석할 수 있다.

▼ GRU+Saint

모델 실험 결과

- Saint 기본 모델의 auc / acc와 비교했을 때 기존 Saint 0.7212 / 0.6811 수준에서 0.7638 / 0.7026의 성능 향상을 보이는 것을 확인할 수 있었다.



결과 해석

- GRUATTN과 비교했을 때 transformer의 encoder 뿐 아니라 decoder에도 sequence적인 특성을 반영했을 때 마지막 문제에 대해 잘 예측하는 것으로 보임.
- Saint와 비교했을 때 GRU를 통해 sequence 정보를 반영한 데이터를 학습시켰을 때 잘 예측하는 것으로 보임.

▼ (4) 하이퍼파라미터 튜닝

WandB의 sweep를 이용하여 하이퍼파라미터 튜닝을 진행했으며 각 모델 별 튜닝에 사용한 하이퍼파라미터는 아래와 같다.

Model	Hyper Parameter
Boosting	n_estimators, min_data_in_leaf, feature_fraction, _lambda
Sequence	batch_size, lr, n_epochs, max_seq_len, hidden_dim, drop_out, n_layers, n_heads
Graph	-

▼ (5) 앙상블

Weighted Ensemble Rate

Model	Weight

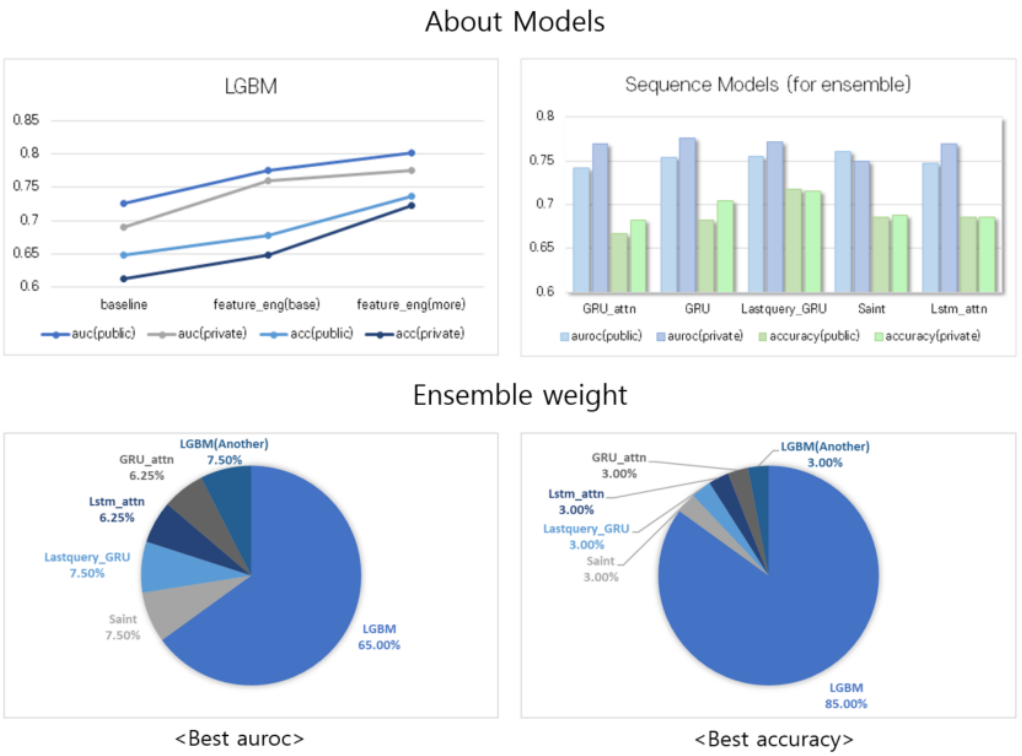
Ensemble Result

public AUC	public ACC

LGBM-v1	0.67
Saint	0.084
Last-Query + GRU	0.064
LSTMATTN	0.064
GRUATTN	0.059
LGBM-v2	0.059

0.8156	0.7527

Model 별 성능 요약 및 Best Ensemble Rate



<Best auroc> <Best accuracy>

▼ 4. 자체 평가 의견

- **잘했던 점**
  - git을 활용한 협업 진행
  - 학습을 위한 Baseline 코드를 수정하여 모델에 맞는 Baseline 재구축 하였다.
- **시도했으나 잘 되지 않았던 것들**
  - git을 활용하여 커밋 컨벤션 및 브랜치 전략은 잘 구성하였으나, Main 브랜치에 병합하여 통합 관리 부분에서 어려움이 있었다.

    - 다양한 모델들로 실험을 했지만 본질적으로 데이터의 특성을 파악하지 못하여 모델의 성능을 끌어올리는 데 한계가 있었다.
- **아쉬웠던 점들**
    - 모델을 구현하는 것에만 집중되어 실험관리를 제대로 하지 못했다.
    - Baseline 코드를 재구축 하는 것에 시간이 많이 소요됐다.
- **프로젝트를 통해 배운 점 또는 시사점**
    - Baseline을 재구축하며 코드 리딩 능력과 이해도를 높일 수 있었다.
    - 모델을 개발하기 전 데이터에 대한 이해가 선행되는 것이 중요하다는 것을 알게 되었다.
    - Wandb 플랫폼을 이용하여 실험 환경을 구성함으로 효율적인 실험 환경의 중요성을 배울 수 있었다.
    - 각 모델들의 결과를 가지고 앙상블을 했을 때 더 좋은 예측 결과가 나온다는 것을 배울 수 있었다.

▼ 5. 개인 회고

▼ 김시윤

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

- lightgcn을 다른 모델에 결합함으로써 train뿐이지만 유의한 학습효과가 존재하는 것을 알아냈다. 이에 대해 모델에 대한 이해를 바탕으로 여러 모델의 특성을 고려한 concatenation이 유의할 수 있다는 것을 알게되었다. 또한 모델과 데이터의 특성을 동시에 고려하는 것이 중요하다는 것을 알게 되었다.

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

- 전에는 하나의 단일모델을 구현하는 것에 만족하였지만, 이번에는 두 개의 모델의 다른 특성을 이해하고, 이를 결합하고자 하였으며, 완벽하지는 않지만 학습효과를 보이는 것을 확인하였다. 이전에는 단순히 모델을 어떻게 사용해야할 것인가에 대해 고민했다면, 이번에는 여러 모델을 이해하고자 하였으며 이를 연결시키기 위해 어떤 방식을 고려해야 하는가에 대해 고민하고 시도했던 것 같다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 구현력의 부족함을 느낀 것 같다. 데이터에 맞는 여러 feature를 생성하고자 하였으나 생각한 만큼 많이 시도하지 못했으며, 모델을 재구성하는 과정에서 일어날 여러문제를 고려하지 않아 디버깅하는데 너무 많은 시간을 들여 효율적인 활동을 하지 못했던 것 같다.

- 주어진 task에 대해 너무 단계적으로 생각한 것 같다. 데이터를 완벽히 파악하고 모델로 넘어가야 한다고 생각했지만, 이 과정은 동시에 진행되는 것이 더 효율적이라고 생각한다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 이론과 논리가 있어도 결국 그것을 원하는 결과로 나타내지 못하면 의미가 없다고 생각한다. 다음 프로젝트에서는 구현하는데 생기는 여러 문제를 빠르게 처리하고자 모델의 구조에 대한 이해를 최우선으로 하고자 한다. 또한 데이터에 대해 빠른 이해력이 필요하기 때문에 데이터를 분석하는데 있어 이번 대회를 바탕으로 더 큰 시야를 가지고 여러 분류로 나눈 후 체계적으로 분석할 예정이다.
- 다음에는 나의 지식을 공유하고, 팀원들의 지식을 받아 폭넓은 사고를 할 수 있도록 하고자 한다.

▼ 김세훈

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

- **Boosting모델 Baseline 구축** : Sequence모델의 성능이 잘 나지 않아 Boosting 계열 모델인 CatBoost와 LGBM 모델의 성능을 측정할 수 있는 Baseline을 구축하였다. 기능별 각 함수를 만들어 모듈화 하는 과정에서 팀원들도 알아볼 수 있도록 최대한 구현하였으나 아직 부족한 면이 보이고, 특히 CatBoost와 LGBM의 실행 방법이 조금 상이하여 이 부분을 해결하려고 Class화 하여 구현했다.
- K**-Fold 적용** : K-Fold는 관점에 따라 2가지로 만들어볼 수 있었다. 첫 번째는 Fold를 기준으로 모두 학습한 후 최종 모델을 만드는 것이고, 두 번째는 각 Fold별로 한 번씩 돌아가며 학습하여 최종 모델을 만드는 것이다. 직접 실험해본 결과 두 번째 방법인 각 Fold별로 한 번씩 돌아가며 학습 시키는 것이 Overfitting 없이 가장 잘 학습할 수 있는 CV 전략이다.
- **Data Augmentation 적용** : Slidding Window 방식을 적용하여 Baseline에 구현하였으나, 실제 성능 측정 결과 DKT 데이터에서는 유의미한 결과를 얻을 수 없었다.
- **T-Fixup모델 구현** : T-Fixup모델을 구현하여 성능을 측정했으나 다른 Sequence모델에 비해 좋은 성능을 나타내지 못하였고, 특히 모델은 구현하였으나 모델에 대한 이해도가 부족하여 결과 분석을 제대로 하지 못하여 사용되지 못했다.
- **LGBM모델 결과 제출** : 기본 변수들이 모두 범주형 변수이기 때문에 Boosting 모델이 적합하다 생각했고, 실험을 한 결과 중 LGBM의 성능이 가장 잘나왔기에 LGBM에 대한 하이퍼 파라미터 튜닝과 K-Fold 등을 진행하여 유의미한 성능의 모델로 만들었다.

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

- git을 활용한 협업이 전과 비교해서 잘 되었다고 생각한다. 팀원들과 함께 Issue를 만들고 Pull Request하는 과정에 익숙해졌으며, merge 충돌 과정도 겪으면서 git을 활용하는 경험을 확실히 늘릴 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- Feature Engineering을 진행하고 무작정 추가해 보자는 식으로만 결과를 측정해서 아쉽다. 만들어진 Feature에 대한 데이터 분포도 확인해보고 팀원과 협의하여 모델에 적절한 Feature를 선택하지 못하였다.
- 실험 관리가 중요하다고 알고 있었지만 실천하지 못했다. 실험관리를 못한 이유는 Baseline 위주로만 모델의 성능을 측정하는데 급급해서 그런 것으로 느껴진다. Baseline만 코드를 실행하면 결과는 측정할 수 있으나 결과에 대한 Feature Importance나 결과값의 분포를 알아내기에는 적합하지 않다. Colab과 같은 jupyterNoteBook을 사용하여 데이터와 실험결과를 분석해보는 시도가 필요하다.
- 이번 대회의 취지는 “사용자가 이전에 풀었던 문제를 기반하여 다음 문제를 풀 수 있을 것인가?”에 대한 것이기에 Sequence 계열 모델을 사용하여 예측 해야 했지만, Boosting 계열의 모델이 성능이 더 좋다는 이유로 시간을 더 많이 써서 아쉽다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 팀원들과 실험 관리 방법을 정하고 절차에 맞게 시도해볼 것이다.
- 대회 취지에 맞는 모델을 사용하여 최대한 고민하여 모델을 구현해볼 것이다.
- 유의미한 Feature를 사용하기 위해 EDA 및 Feature Engineering에 대한 결과를 시각화 하여 만들어 볼 것이다.

▼ 문찬우

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

- Baseline 코드 외의 여러 시퀀스 모델을 추가해서 돌려볼 수 있었다. 기본적인 성능을 확인하기 위하여 베이스라인 코드외의 rnn과 gru를 추가하여 순환신경망 모델류에서 가장 성능이 잘나오는 모델을 확인하였고, 앙상블 성능 향상을 위해 TCN등 다양한 유형의 시퀀스 모델등을 추가로 실험해보았다. 기존 lstm에 비해 gru의 성능 향상이 있어 가장 높은 성능을 보였던 lastquery에 적용하여 성능을 높였고, TCN은 단일 성능은 높지 않았지만, 앙상블에 적용하면 성능 향상이 있을거란 기대가 있었지만 실험해보지는 못하였다. 다음 대회에서는 EDA를 통해 데이터에 대한 특성을 좀 더 구체적으로 파악하고, 논리적인 근거를 통해 데이터에 맞는 모델을 구현해 보는 연습을 해야할 것 같다.

- DKT 데이터의 경우엔 timestmp를 전처리 하여 유저가 푼 문제와 문제 사이의 시간 간격(elaspsed)을 구할 수 있었고, 이를 추가 적인 continous feature로 활용하였을 경우에 성능이 오르는 것을 확인하였다. 시퀀스의 시간적인 순서 에 대해서, 문제에 대한 시간적인 정보를 단순히 추가적인 feature로 활용하기 보다는 positional encoding 부분의 순서 간격을 유저 별로 전 처리한 문제의 시간 간격으로 바꾸는 작업을 시도해 보았는데, train/valid 단계에서 어느정도의 준수한 성능이 있었지만, test 성능이 오르지 않아서, 큰 역할을 하지 못한 것으로 판단했다. 또한, elapsed를 전처리 할때, 문제의 시간 간격이 10분 이상인 유저에 대해서 단순히 이상치로 판단하고 전처리 하였는데, 충분한 EDA를 통해 전처리 해볼 걸 하는 아쉬움이 있다.

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

- 저번 대회를 진행 할 때에는 단순히 모델의 내부적인 파이프 라인을 수정하고, 튜닝을 통해 성능을 올리는 일에만 집중을 하였었는데, 이번 대회에서는 모델에 추가적인 기능을 적용하기 위해 dataloader와 embedding 부분을 초기 단계 부터 바꾸는 작업을 하면서, 시퀀스 형 모델이 데이터를 어떤 식으로 처리하는지 더 깊게 이해하였다.

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

- 먼저, 코드에 대하여 일괄적인 정리를 체계적으로 하지 못한 것이 아쉬웠다. Git에 대한 숙련도가 부족하여 겁을 먹고 자주 사용하지 못했는데, 더 익숙해질 필요가 있다고 생각한다.
- 실험을 하기 위해 수정한 코드를 이후에 다른 실험 및 재현을 위해 처음부터 자동화 및 인자를 추가하는 작업을 진행해야겠다. 이번 대회에서는 단순히 실험을 해보는 것에만 급하여 오히려 재현 할때 더 많은 시간을 소모한 것 같다.
- EDA에 대해 많은것을 알아보지 못한것이 아쉽다. EDA를 끝내고 모델링을 진행한다고 생각하였는데, 생각보다 모델링을 하면서 새롭게 해야 할 EDA가 많다는 것을 느꼈고, 앞으로는 모델링을 하면서 EDA를 같이 할 수 있는 환경을 구축해 볼 생각이다.
- 모델링에 대해서 어떤 식으로 데이터에 대한 모델링을 하는 것 인지에 대한 막연함이 있었는데, 1등조의 발표를 통해 확실한 감을 얻은 것 같아 감사했다. 1등조의 경우 SAS-Rec을 사용하였는데, 기존의 baseline에서 bert의 성능이 다른 단방향 순환신경망 모델보다 성능이 나오지 않는것을 판단하고 이를 통해 단방향 Attention인 Sas-rec과 같은 모델을 사용했다면, 굳이 positional encdoing에 대한 고민을 하지 않고도 더 좋은 성능을 낼 수 있었을 것 같다.

▼ 배건우

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

- 팀원들과 버전들을 맞추기 위하여 baseline base코드를 자동화 시켰습니다. 그래서 코드만 복붙하면 환경이 구성되고 github에서 버전충돌도 없게 하였습니다.
- 베이스 python script에 코드들을 자동화 시켜서 팀원들이 쉽게 작업할 수 있도록 하였습니다. 특히 feature 추가하는 것과 embeding 부분이 너무 불편하게 되어있었는데 feature를 추가하는 함수 한 줄만 적으면 embedding까지 한 번에 이뤄지도록 end-to-end로 자동화 시켰습니다.
- hyper parameter를 최적화 시켜주는 도구인 sweep의 base 코드를 만들었습니다. 팀원들이 hyper parameter tunning하는 시간을 아껴 다른 작업에 시간을 쓸 수 있도록 하였습니다.우

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

- 전 대회에서는 구현한 모델을 베이스 python script에 녹이지 못하고 Jupyter Notebook를 사용한 것이 너무 아쉬웠습니다. 그래서 이번 대회에서는 베이스 python script에 구현한 코드를 녹이는 것이 목표였는데 성공하였습니다. python script에 익숙해져서 큰 프로젝트를 효율적으로 다뤄볼 수 있었습니다.
- 전 대회에서 hyper parameter tunning에 시간을 많이 할애하여 시간이 너무 아깝다고 느꼈습니다. 그래서 이번에 hyper parameter tunning을 최적화 하는 도구를 활용하고 싶었는데 적용에 성공하였습니다. 이 시간을 아껴 모델을 더 많이 공부할 수 있었습니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 제가 하나하나 python script를 짠 것이 아니고 베이스 python script를 수정하는 것에서 그친 것이 아쉬웠습니다.
- python script에 익숙하지 않다보니 코드 치는데 시간이 많이 흘렀습니다. 그래서 모델 성능을 높이는 것에는 많은 노력을 하지 못했습니다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 베이스 python script 제가 수정한 부분 말고도 비효율적인 부분이 많았습니다. 다음에는 새로 custom하게 코드를 구성하여 저희 입맛에 맞춰보려고 합니다.
- python script에 더욱 익숙해져 모델 성능을 높이는 것에도 노력을 하려고 합니다.

▼ 이승준

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

- Saint 모델을 개발했을 때 성능이 잘 나오지 않는 문제가 발생했는데, 이를 해결하기 위해 모델의 구조를 분석했다. transformer의 특징이 양방향 sequence를

본다는 것인데 마지막 문제를 맞춰야 하는 대회의 특성에 맞지 않는 모델이라고 생각했다. 따라서 다른 팀원이 실험한 LastQuery 모델의 성능이 뛰어나다는 것을 실험을 통해 확인했다.

- 그래서 Saint 모델의 input 데이터를 GRU를 통해 sequence 정보가 반영된 데이터로 모델을 합쳤고 그 결과 기존 saint 모델보다 성능 면에서 평균 약 5%가량의 성능 향상을 얻을 수 있었다.

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

- 모델을 하나씩 뜯어보며 내가 원하는 구조로 개선했던 프로젝트였다. 기존의 categorical data만 학습하는 구조에서 continuous data도 학습할 수 있는 구조로 모델을 개선하면서 모델에 대해 더 자세히 알 수 있었다. 또한 해당 실험을 해보며 saint 모델 및 transfomer 모델에서 continuous feature를 추가하는 것은 효용성이 없다는 사실도 알게 되었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- EDA를 하기 위해서 Pandas 및 matplotlib를 사용하는데 미숙하여 데이터에 대해 원하는 인사이트를 제대로 얻지 못한 점이 아쉽다.
- 모델 구현을 하는 부분에서 코딩 능력이 많이 부족하다는 것을 알게 되었고 결국 많은 시간을 모델 구현에 할애하여 다양한 실험들을 시도해보지 못한 것이 아쉽다.
- 대회 후반부에 들어서 모델의 성능을 끌어올리기 위한 실험들을 진행하고 모델의 구조를 바꾸는 실험을 진행했는데 조금 더 빨리 다양한 시도들을 했다면 더 좋은 성과가 나왔을 것이라고 생각한다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- EDA를 위한 pandas를 활용, 데이터 핸들링 및 matplotlib을 통한 시각화까지 더 발전된 데이터에 대한 인사이트 도출
- 빠른 모델 구현 및 실험으로 모델의 구조를 EDA를 통해 밝혀낸 데이터의 특성을 고려하여 다양하게 변경하며 실험해볼 것이다.
- 모델을 선택할 때 조금 더 깊게 이유를 생각해보고 모델을 개발해볼 것이다. 또한 모델 실험들을 자세하게 정리해놓고 이를 바탕으로 빠르게 모델이나 데이터의 개선으로 방향 전환을 할 것이다.