Wrap-Up Report

1. Team Wrap-up Report

▼ 1-1. 프로젝트 개요

프로젝트 주제

- **Deep Knowledge Tracing**
- **"지식 상태"를 추적하는 딥러닝 방법론**.
- 평가 지표 : AUROC(Area Under ROC Curve)

활용 장비 및 재료

- 개발 환경
    - 팀 구성 : 5인 1팀, 인당 V100 서버를 VS Code와 SSH로 연결하여 사용
    - 협업 환경 : Notion, GitHub, Wandb
    - 의사소통 : Slack, Zoom
    - 기타 : pyenv, poetry

사용 데이터셋의 구조

- Train Data : 2,526,700행, 6열
- Test Data : 260,114행, 6열

변수명	변수 설명
`userID`	사용자의 고유 번호 (총 7,442명의 고유한 사용자)
`assessmentItemID`	문항의 고유 번호 (총 9,454개의 고유한 문항)
`testId`	시험지의 고유 번호 (총 1,537개의 고유한 시험지)
`Timestamp`	사용자가 해당 문항을 풀기 시작한 시점의 데이터
`KnowledgeTag`	문항 중분류 태그 (총 912개의 고유한 태그)
`answerCode` **(Predict Label)**	사용자가 해당 문항을 맞췄는지 여부에 대한 이진 데이터 (0 → 틀림 / 1 → 맞춤) Train Data : 65.45%가 정답을 맞춤 (Test Data → 각 유저의 마지막 시퀀스 데이터 : -1)

▼ 1-2. 프로젝트 팀 구성 및 역할

이름	역할
노관옥	Python Modularization, NGCF Modeling, Environment Setting, Github Issue Managing
박경원	Tranformer, LightGCN modeling, initial setting shell script
이석규	Project Environment Settings, Apply LightGCN to Transformer Model
장성준	wandb settings, EDA, Feature Engineering, GRU · GRUAttention Modeling
이진원	EDA, Feature Engineering, XGBoost · CatBoost · SAINT+ Modeling

▼ 1-3. 프로젝트 수행 절차 및 방법

환경 세팅 및 협업

- 개발 환경 : VSCode, Jupyter Notebook
- 협업 : GitHub, Slack, Notion, Zoom, Wandb

▼ 서버 환경 관리 - Shell Script

Pyenv

- pyenv를 사용하여 가상 환경 분리를 통해 의존성 낮춤

Poetry

- conda 대신 poetry를 사용하여 패키지 간 의존성 관리

환경 설정 스크립트

- 서버가 재실행돼도 환경을 새롭게 구축하지 않도록 자동화

▼ 코드 버전 관리 - Github

Pull Request 전략

- main 브랜치에 직접 Push 하지 않고 PR을 통한 Merge

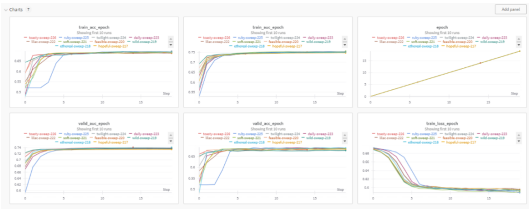
Code Review 문화

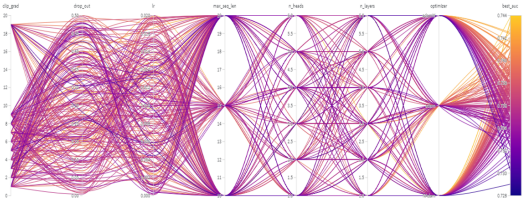
- 팀원이 올린 PR을 리뷰하며 코드의 품질 향상

Squash Merge

- Squash Merge를 활용해 main 브랜치의 복잡도를 낮춤

▼ 모델 실험 관리 - Weights & Biases



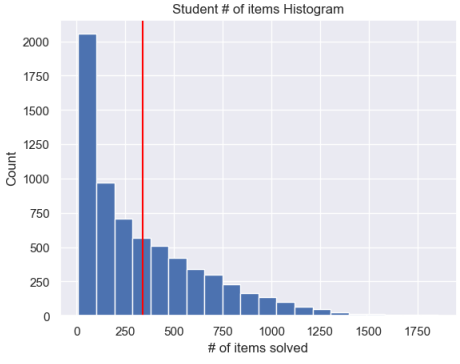


- Weights & Biases를 활용한 모델 실험 관리
- sweep 기능을 활용한 하이퍼파라미터 최화

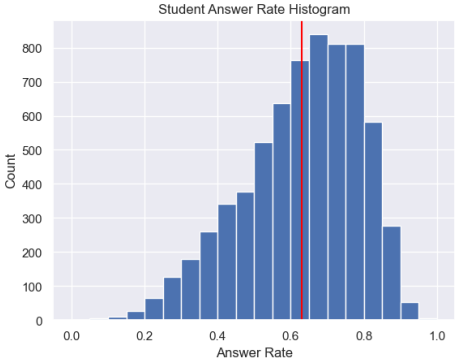
EDA

▼ 사용자 분석

- 사용자별 푼 문항 수

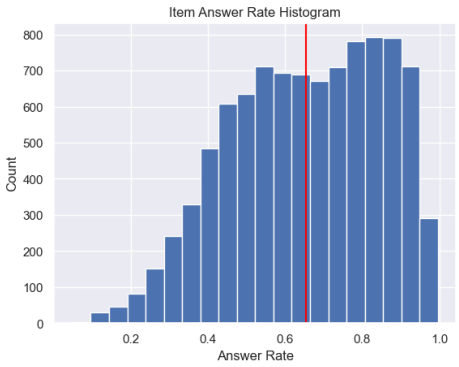


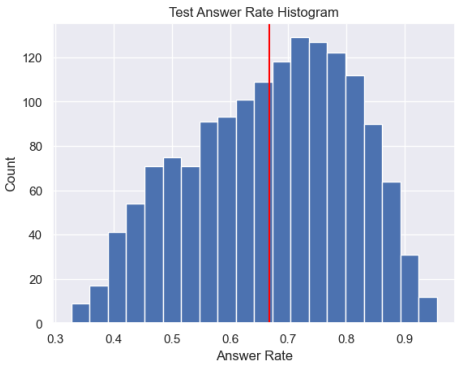
- 학생 별 정답률



▼ 문항 별 /시험지 별 정답률 분석

- 문항 별 정답률 분석
- 시험지 별 정답률 분석





▼ 특성 별 빈도 분석 종합

특성	고유값	평균 문항수	최소값	1분위수	2분위수 (중앙값)	3분위수	최대값
학생 userID	7,442명	339문항	9문항	78문항	232문항	519문항	1,860문항
문항 assessmentItemID	9,454개	267회	50회	250회	250회	300회	500회
시험지 testId	1,537개	1,643회	200회	1,500회	1,500회	1,800회	4,400회
태그 KnowledgeTag	912개	2770회	50회	1,137회	2,500회	4,500회	14,350회

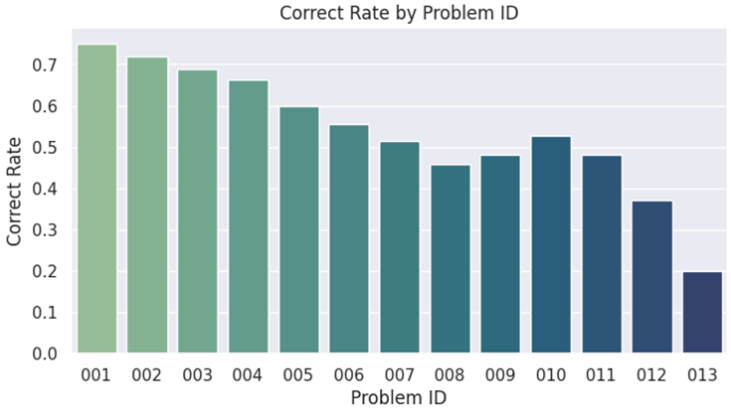
▼ 특성 별 정답률 분석 종합 (단위 : %)

특성	평균 정답률	최소값	1분위수	2분위수 (중앙값)	3분위수	최대값
학생 userID	62.8	0	52.7	65.1	75.0	100
문항 assessmentItemID	65.4	4	50.4	66.6	82	99.67
시험지 testId	66.8	33.0	56.2	68.1	77.8	95.6
태그 KnowledgeTag	61.6	14.7	49.9	60.1	74.1	97.2

▼ 시험지의 대분류별 정답률 분석

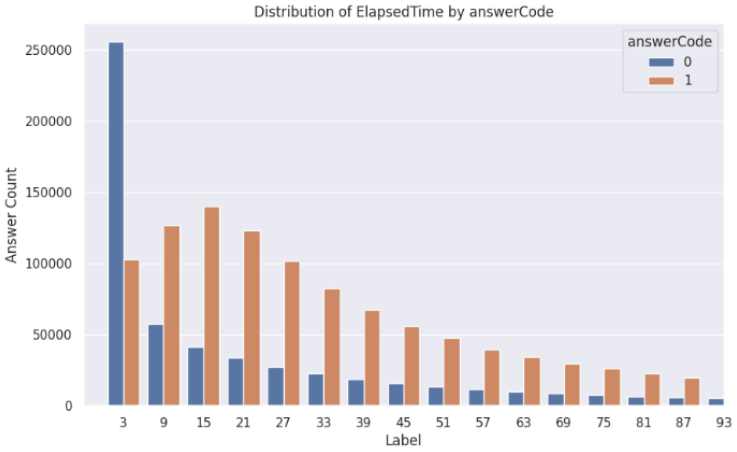
- testID의 3번째 자리는 1~9의 값을 가짐
- 특히, **번호가 커질수록 정답률이 낮아짐**
  - 이를, **시험지의 대분류(testCode)**라는 새로운 변수로 사용

▼ 시험지의 문항 번호별 정답률 분석



- ItemID의 8~10번째 자리는 1~13의 값을 가짐
- 특히, **번호가 커질수록 정답률이 낮아짐**
  - 이를, **시험지의 문항 번호(ProblemID)**라는 새로운 변수로 사용

▼ 문제 풀이 시간에 따른 정답률 분석



- 문제 풀이 시간이 매우 짧은 경우에, 오답의 수가 매우 높음
- 시간을 기준으로, **실제 문제를 풀었는지 여부**를 변수로 생성

Feature Engineering

▼ 시험지 관련 변수

변수명	변수 설명
`user_sum`	유저 별 정답을 맞힌 횟수
`user_cnt`	유저 별 문제를 푼 횟수
`user_acc`	유저 별 정답률
`user_itemID_sum`	유저의 `assessmentItemID` 별 정답을 맞힌 횟수
`user_itemID_cnt`	유저의 `assessmentItemID` 별 문제를 푼 횟수
`user_itemID_acc`	유저의 `assessmentItemID` 별 정답률
`user_testID_sum`	유저의 `testID` 별 정답을 맞힌 횟수
`user_testID_cnt`	유저의 `testID` 별 문제를 푼 횟수
`user_testID_acc`	유저의 `testID` 별 정답률
`user_testCode_sum`	유저의 `testCode` 별 정답을 맞힌 횟수
`user_testCode_cnt`	유저의 `testCode` 별 문제를 푼 횟수
`user_testCode_acc`	유저의 `testCode` 별 정답률
`user_testNum_sum`	유저의 `testNum` 별 정답을 맞힌 횟수
`user_testNum_cnt`	유저의 `testNum` 별 문제를 푼 횟수
`user_testNum_acc`	유저의 `testNum` 별 정답률
`user_problemID_sum`	유저의 `problemID` 별 정답을 맞힌 횟수

변수명	변수 설명
`user_problemID_cnt`	유저의 `problemID` 별 문제를 푼 횟수
`user_problemID_acc`	유저의 `problemID` 별 정답률
`user_tag_sum`	유저의 `KnowledgeTag` 별 정답을 맞힌 횟수
`user_tag_cnt`	유저의 `KnowledgeTag` 별 문제를 푼 횟수
`user_tag_acc`	유저의 `KnowledgeTag` 별 정답률

▼ TimeStamp 관련 변수

변수명	변수 설명
`itemID_sum`	전체 유저의 `assessmentItemID` 별 정답을 맞힌 횟수
`itemID_cnt`	전체 유저의 `assessmentItemID` 별 문제를 푼 횟수
`itemID_acc`	전체 유저의 `assessmentItemID` 별 정답률
`testID_sum`	전체 유저의 `testID` 별 정답을 맞힌 횟수
`testID_cnt`	전체 유저의 `testID` 별 문제를 푼 횟수
`testID_acc`	전체 유저의 `testID` 별 정답률
`testCode_sum`	전체 유저의 `testCode` 별 정답을 맞힌 횟수
`testCode_cnt`	전체 유저의 `testCode` 별 문제를 푼 횟수
`testCode_acc`	전체 유저의 `testCode` 별 정답률
`testNum_sum`	전체 유저의 `testNum` 별 정답을 맞힌 횟수
`testNum_cnt`	전체 유저의 `testNum` 별 문제를 푼 횟수
`testNum_acc`	전체 유저의 `testNum` 별 정답률
`problemID_sum`	전체 유저의 `problemID` 별 정답을 맞힌 횟수
`problemID_cnt`	전체 유저의 `problemID` 별 문제를 푼 횟수
`problemID_acc`	전체 유저의 `problemID` 별 정답률
`tag_sum`	전체 유저의 `KnowledgeTag` 별 정답을 맞힌 횟수
`tag_cnt`	전체 유저의 `KnowledgeTag` 별 문제를 푼 횟수
`tag_acc`	전체 유저의 `KnowledgeTag` 별 정답률
`itemID_high_freq`	전체 유저에 대해 `assessmentItemID` 가 평균 이상으로 노출되었는지 여부
`testID_high_freq`	전체 유저에 대해 `testID` 가 평균 이상으로 노출되었는지 여부
`testCode_high_freq`	전체 유저에 대해 `testCode` 가 평균 이상으로 노출되었는지 여부
`testNum_high_freq`	전체 유저에 대해 `testNum` 가 평균 이상으로 노출되었는지 여부

변수명	변수 설명
`tag_high_freq`	전체 유저에 대해 `KnowledgeTag`가 평균 이상으로 노출되었는지 여부

▼ 유저 별로 변수 별 정답/문제 풀이 수, 정답률(시간 순으로 누적)

변수명	변수 설명
`user_past_solved`	유저가 해당 문제를 이전에 풀었던 횟수
`relative_correct_rate`	상대적인 정답률 (`answerCode` - `itemID_acc`)
`is_correct_before1`	1번째 이전 문제의 정답 여부
`correct_rate_before1`	1번째 이전 문제의 정답률
`relative_correct_rate_before1`	1번째 이전 문제의 상대적인 정답률
`is_correct_before2`	2번째 이전 문제의 정답 여부
`correct_rate_before2`	2번째 이전 문제의 정답률
`relative_correct_rate_before2`	2번째 이전 문제의 상대적인 정답률
`is_correct_before3`	3번째 이전 문제의 정답 여부
`correct_rate_before3`	3번째 이전 문제의 정답률
`relative_correct_rate_before3`	3번째 이전 문제의 상대적인 정답률
`is_correct_before4`	4번째 이전 문제의 정답 여부
`correct_rate_before4`	4번째 이전 문제의 정답률
`relative_correct_rate_before4`	4번째 이전 문제의 상대적인 정답률
`is_correct_before5`	5번째 이전 문제의 정답 여부
`correct_rate_before5`	5번째 이전 문제의 정답률
`relative_correct_rate_before5`	5번째 이전 문제의 상대적인 정답률

▼ 전체 유저에 대한 변수 별 정답 수 / 문제 풀이 수 / 정답률 계산

변수명	변수 설명
`itemID_sum`	전체 유저의 `assessmentItemID` 별 정답을 맞힌 횟수
`itemID_cnt`	전체 유저의 `assessmentItemID` 별 문제를 푼 횟수
`itemID_acc`	전체 유저의 `assessmentItemID` 별 정답률
`testID_sum`	전체 유저의 `testID` 별 정답을 맞힌 횟수
`testID_cnt`	전체 유저의 `testID` 별 문제를 푼 횟수
`testID_acc`	전체 유저의 `testID` 별 정답률
`testCode_sum`	전체 유저의 `testCode`별 정답을 맞힌 횟수

변수명	변수 설명
`testCode_cnt`	전체 유저의 `testCode` 별 문제를 푼 횟수
`testCode_acc`	전체 유저의 `testCode` 별 정답률
`testNum_sum`	전체 유저의 `testNum` 별 정답을 맞힌 횟수
`testNum_cnt`	전체 유저의 `testNum` 별 문제를 푼 횟수
`testNum_acc`	전체 유저의 `testNum` 별 정답률
`problemID_sum`	전체 유저의 `problemID` 별 정답을 맞힌 횟수
`problemID_cnt`	전체 유저의 `problemID` 별 문제를 푼 횟수
`problemID_acc`	전체 유저의 `problemID` 별 정답률
`tag_sum`	전체 유저의 `KnowledgeTag` 별 정답을 맞힌 횟수
`tag_cnt`	전체 유저의 `KnowledgeTag` 별 문제를 푼 횟수
`tag_acc`	전체 유저의 `KnowledgeTag` 별 정답률
`itemID_high_freq`	전체 유저에 대해 `assessmentItemID` 가 평균 이상으로 노출되었는지 여부
`testID_high_freq`	전체 유저에 대해 `testID` 가 평균 이상으로 노출되었는지 여부
`testCode_high_freq`	전체 유저에 대해 `testCode` 가 평균 이상으로 노출되었는지 여부
`testNum_high_freq`	전체 유저에 대해 `testNum` 가 평균 이상으로 노출되었는지 여부
`tag_high_freq`	전체 유저에 대해 `KnowledgeTag` 가 평균 이상으로 노출되었는지 여부

▼ 과거 정보 활용 변수

변수명	변수 설명
`user_past_solved`	유저가 해당 문제를 이전에 풀었던 횟수
`relative_correct_rate`	상대적인 정답률 (`answerCode` - `itemID_acc`)
`is_correct_before1`	1번째 이전 문제의 정답 여부
`correct_rate_before1`	1번째 이전 문제의 정답률
`relative_correct_rate_before1`	1번째 이전 문제의 상대적인 정답률
`is_correct_before2`	2번째 이전 문제의 정답 여부
`correct_rate_before2`	2번째 이전 문제의 정답률
`relative_correct_rate_before2`	2번째 이전 문제의 상대적인 정답률
`is_correct_before3`	3번째 이전 문제의 정답 여부
`correct_rate_before3`	3번째 이전 문제의 정답률
`relative_correct_rate_before3`	3번째 이전 문제의 상대적인 정답률

변수명	변수 설명
`is_correct_before4`	4번째 이전 문제의 정답 여부
`correct_rate_before4`	4번째 이전 문제의 정답률
`relative_correct_rate_before4`	4번째 이전 문제의 상대적인 정답률
`is_correct_before5`	5번째 이전 문제의 정답 여부
`correct_rate_before5`	5번째 이전 문제의 정답률
`relative_correct_rate_before5`	5번째 이전 문제의 상대적인 정답률

▼ **ELO Rating 변수**

변수명	변수 설명
`theta`	ELO Rating의 유저의 theta 추정 값 (`theta_estimate_json` 에 저장)
`beta`	ELO Rating의 문제의 beta 추정 값 (`beta_estimate_json` 에 저장)

Feature Selection

- 과적합 방지를 위해 생성한 변수에 **VarianceThreshold** 적용하여 분산 임계값을 충족하지 않는 변수를 제거

```
from sklearn.feature_selection import VarianceThreshold

selector = VarianceThreshold(0.8)
train_thresh = selector.fit(X_train)
select_feat = train_thresh.get_feature_names_out()
```

Modeling

시도한 모델

- LSTM
- GRU
- LSTMAttention
- GRUAttention

- LightGCN
- LGCN Transformer
- Last Query Transformer
- SAINT+
- XGBoost
- CatBoost

▼ 1-4. 프로젝트 수행 결과

- 시계열성을 분해하여 변수로 활용한 XGBoost, CatBoost 등 Tree 기반 모델이 가장 우수한 성능을 보여줌
- Last Query Transformer와 SAINT+ 모델 또한 리더보드 기준 AUROC가 0.8 이상으로 우수한 성능을 보여줌

최종 모델

모델	XGBoost	CatBoost	Last Query Transformer	SAINT+
AUROC (LB)	0.8302	0.8253	0.8092	0.8042
Accuracy (LB)	0.7661	0.7473	0.7366	0.7258

- 모델의 성능에 따라 가중치를 주어 **0.75 : 0.15 : 0.07 : 0.03**의 비율로 앙상블을 진행

AUROC (Public)	0.8316	2nd
AUROC (Private)	0.8529	2nd

▼ 1-5. 자체 평가 의견

잘한 점

- 저번 프로젝트가 끝난 후 도전해보고자 했던
  - pyenv와 poetry를 활용한 환경 세팅
  - Github를 활용한 이슈 관리, Pull Request, 코드 리뷰
  - Notion을 활용한 일정 관리
  - Wandb를 활용한 실험 관리

  를 모두 성공적으로 해서 뿌듯하다.
- 이번 대회에서도 상위권에 들어서 다른 캠퍼 앞에서 발표를 할 수 있는 기회를 얻었고, 마스터님의 피드백을 받을 수 있어서 좋았다.

시도했으나 잘 되지 않았던 것

- LightGCN과 Transformer를 합친 모델을 만들었지만, 성능이 좋지 못해서 실패하였다.
- Crossformer를 사용하지 못했다.
- Sequential 한 Feature의 특성을 잘 살리지 못한 것 같다.

아쉬웠던 점

- Feature Store가 Feature 관리에 용이한 솔루션이 아니라는 걸 늦게 알았다.
  - 대회에서는 DVC가 좋은 대안이라는 결론을 내렸다.
- LightGCN에 더해, UltraGCN을 사용하려 했지만 시간이 부족해서 하지 못 함

프로젝트를 통해 배운 점 또는 시사점

- Wandb Sweep를 사용해 HPO 시각화를 통해 인사이트를 얻을 수 있었다.
- 실험을 여러번 실행하면서 반복되는 작업을 더 직관적으로 자동화하는 방법에 대한 깊은 고민을 했다.
- 데이터 관리에 대한 다양한 시도를 했는데 실험을 반복적으로 수행하기 위해서는 데이터가 무조건 로컬 사이에 있어야 한다는 걸 알았다.
- poetry를 사용해 팀원들과 동일한 Package 버전을 관리했다.

2. 개인 회고

▼ 노관옥_T6050

- 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?
  - 모델 공부보다 운영을 시도해봤습니다. AI 컴피티션에서 MLOps를 어떻게 적용할 수 있을 지에 대한 고민을 많이 했습니다. 부스트캠프에 들어오기 전 가장 이루고 싶은 목적이었기에 정말 만족했습니다.
- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?
  - 모델 개발보다 데이터 엔지니어링, MLOps에 관심이 많다고 느껴졌습니다. 이러다 보니 대회 기간 모델에 기여한 부분은 거의 없고 모델에 대한 이해도 부족하다고 생각됩니다. 기본적인 스킬을 제대로 연마하는 걸 포기하고도 MLOps 엔지니어를 할

수 있는지도 잘 모르겠습니다. 하지만 함께 프로젝트를 진행하면서 다양한 시행착오를 겪으며 문제를 해결하다보니 어쩌면 제가 부스트캠프에 들어온 이유기도 하고 어디서도 할 수 없는 경험을 하고 있다는 생각도 듭니다. 지금의 팀원을 믿고 제가 자신있는 분야를 깊게 공부해서 Ops가 이루어진다면 이게 바로 제가 부스트캠프에서 얻어갈 수 있는 최고의 경험이라고 믿기로 했습니다.

- 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?
  - DVC 도입
  - 데이터 분석 플랫폼 구축
  - 데이터 파이프라인 구축
  - Train 오퍼레이터 구축

▼ 박경원_T6055

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

저는 이전 프로젝트에선 제 스스로가 팀에 기여도가 낮다고 생각했습니다. 그래서 이번 프로젝트에선 기여도를 높이고 싶었습니다.

이번 프로젝트에선 팀원들과 협업할 수 있는, 현업에서 많이 사용하는 툴을 사용하는 것도 목적 중 하나였습니다.

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

강의에서 트랜스포머 모델에 대해 매우 자세히 설명해 주시고 변형된 모델에 대해서도 소개해주셨습니다. 문득, 이전에 유튜브에서 사람의 의식을 설명하는 이론인 GWT를 가장 잘 표현하는 모델이 트랜스포머란 얘기가 생각났고, 이 생각에 꽂혀서 베이스라인을 기본 틀로 밑바닥부터 하나하나 만들었습니다. (베이스라인에도 트랜스포머 모델이 포함되어 있지만, 다 만들고 나서 뒤늦게 알았습니다..)

이전 트랜스포머 강의 자료로 제공된 코드를 사용해서 만들고, 추가로 마지막 쿼리만을 사용하는 모델을 만들기위해 multi-head attention코드를 추가하고, 트랜스포머 뒤에 LSTM을 추가해보고, LightGCN으로 user와 문제를 임베딩해서 트랜스포머에 넣어보고, 한 번에 다 하려면 못 할 것들을 하나하나 만드니 완성할 수 있었습니다. 결과적으로 처음 모델이 가장 좋은 성능을 냈고, LightGCN을 추가한 모델은 성능이 되려 떨어져서 실망했지만요.

결론은 처음으로 직접 모델의 밑바닥부터 끝까지, 그리고 튜닝까지 해서 팀에 일부분 기여했고, 이전부터 부족하다고 생각됐던 지식을 채우기 위해 저번 강의들을 찾아보고, 하루종일

모델에 대해 생각했었습니다. 되돌아보니 제 자신에세 너무 자랑스럽습니다. 다음 프로젝트에선 어떤 모델을 만들어야 해도 두렵지 않을 것 같습니다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

트랜스포머에 LSTM을 섞는것 까지만 성능이 좋고 LGCN을 섞는건 성능이 좋지 못해 아쉬웠습니다.

혹여나 만든 모델을 잘못 만들었나싶어 모델 전체를 세세하게 보려했지만, 프로젝트가 얼마 남지않아 다른 분들이 하고있는 마무리 작업을 도와주는게 낫다고 생각했습니다. 시간이 많다면 더 많은 모델을 만들어보고 싶었습니다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

프로젝트와는 별개로 sequence 데이터를 처리하는 저만의 모델이 생각났었습니다. NLP 과정에 sequence모델 강의가 좋다고 들어서 지금 강의도 들어보고 있습니다. 이 모델을 만들어서 다음 프로젝트에 혹은 최종프로젝트에 도움이 되었으면 좋겠습니다.

▼ 이석규_T6124

학습 목표 설정

DKT라는 새로운 방법론에 지식 추적 대신 주어진 문제를 풀지 못풀지 예측하는 과정을 완벽히 이해하고자 다음과 같은 목표를 설정했습니다.

- 강의 기반 모델을 전부 구현해보기
- 구현한 모델을 실험하고 최적의 모델 찾기
- 팀원과 협업하는 힘 기르기(Github, notion 활용하기)
- 새로운 스킬 쌓기(wandb)

저는 강의 기반 내용 중 EDA와 Feature Engineering을 강의 보다 더 깊게 진행해보았고, 강의에서 놓치고 간 분산이 쏠려 long tail을 띄는 feature를 log를 씌워 처리하는 등 저만의 방식도 녹여내보았습니다.

이 후 모델을 실험하는 과정에서 LightGCN 모델의 학습을 통해 생긴 embedding layer를 transformer 기반에 사용해보고자 노력했으나 모델의 점수적 성과가 좋지 않아 폐기했습니다.

비록 제가 만든 모델이 좋은 실험적 성과는 없었으나 하나씩 제 스스로 모델을 설계하면서 기본이 된 모델들에 대한 이해도를 높일 수 있었습니다.

또한, 이번 대회에선 github을 적극적으로 사용했습니다. 팀원들과 Issue, 그리고 그에대한 feedback을 통해 개개인의 학습에도 도움이 되고자 노력했습니다.

도전과 개선

모델을 구현할 때, 체계적인 방법을 도입해야함을 실험적으로 깨달았습니다. 각 기능을 그 때 그 때 만들다보니 모델의 규모가 커지면 수정이 어려워졌습니다. 그래서 저는 다음 모델을 만들 땐 Jupyter notebook으로 전체 흐름을 잡고 각 기능을 module화 하여 구현하는 방법을 사용해볼 것 입니다.

또, 짧은 시간 내 학습한 과정을 소화하는게 쉽지 않았습니다. 기본적인 부분을 다시 찾아보는 경험이 많았는데, 이해했다고 생각한 부분은 기록을 남기고자 노력했습니다. 이런 기록을 남기는 게 익숙하지 않아 처음엔 어려웠지만 기록을 하는 행위 자체에서 이해도가 많이 높아졌고, 다시 이 기록들을 살펴보며 동기부여도 확실히 잘 되었습니다. 이번 대회를 진행하며 'Entropy, Cross-Entropy', 'Metrics(Accuracy, AUROC)' 등에 대하여 정리해보았습니다. 기록하는 과정 외에도 주말과 저녁시간대를 사용하여 '밑바닥부터 시작하는 딥러닝'책을 활용한 추가 학습도 진행했습니다.

이번 대회에서 저의 학습과정은 팀에서 1인분을 하기위해 개인 역량을 키우기위해 많이 노력했습니다.

이를 바탕으로 다음 프로젝트에선 모델의 구현을 빠르게 진행하여 많은 모델을 미리 실험할 수 있게 팀에 기여하고 싶습니다.

▼ 장성준_T6147

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**
  - wandb settings
    - 이전 프로젝트 기간이 짧아 사용하지 못했는데 이번 프로젝트에서 wandb를 연동해서 모델 실험 관리를 진행하였고 wandb sweep을 통해 하이퍼파라미터 튜닝을 진행해 봤습니다.
  - GRU · GRUAttention Modeling
    - base code에 구현된 모델인 LSTM보다 가볍고 추론 속도가 빠른 모델을 구현해 봤습니다. 하지만 이 대회의 목적은 AUROC 성능이므로 모델의 추론 속도 대비 성능이 좋았다고 생각하지만 AUROC가 LSTM 모델보다 낮아 사용하지 않았습니다. 모델을 구현했다는 점에 의의를 두었습니다.
- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  - 프로그래밍 역량 부족으로 인해 대회를 시작했을 때 목표였던 Crossformer를 사용하지 못했습니다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해 보고 싶은 점은 무엇인가?**
    - 프로그래밍 역량을 기르기 위해 다음 프로젝트의 base code를 end to end로 리팩토링을 진행해 보고 싶습니다.
- **나는 어떤 방식으로 모델을 개선했는가?**
    - Feature Engineering을 통해 성능을 개선했습니다. 데이터의 주기성이 중요하다고 판단하여 분기와 각 시험지의 총 문항 개수 변수를 추가했습니다.
    - 생성한 칼럼이 총 95개로 전부 사용하면 과적합이 발생할 것 같아 변수를 선택해야 한다고 생각했습니다. 변수의 분산이 작으면 target data에 미치는 영향이 미미하다고 판단했습니다. 따라서 **Variance Threshold**라는 Feature Selection 기법을 사용해 분산 임계값을 지정하고 임계값보다 높은 변수만 선택해 최종적으로 66개의 변수를 사용해 과적합을 방지했습니다.
- **협업 과정에서 잘된 점/ 아쉬웠던 점은 어떤 점이 있는가?**
    - 발표 자료에 여태까지 시도했던 모델에 대한 설명을 넣지 않아서 아쉬웠습니다.
        - 발표가 끝난 이후 피드백을 받을 때 최종 프로젝트 이외에 여태까지 구현하고 실험한 모델에 대한 설명을 추가 했으면 좋겠다는 마스터 님의 질문을 받기 전까지 생각하지 못한 내용이었습니다. 다음 프로젝트에 발표할 일이 생긴다면 추가해 볼 예정입니다.
    - 협업을 위한 Github 사용에 익숙해졌습니다.
        - 기존에 개인 프로젝트 정리용으로 git을 사용하다가 처음으로 팀 프로젝트를 위해 git을 사용해 미숙한 부분이 많았습니다. 하지만 팀원 분들이 많이 도와줘서 이제는 git을 사용하는 것이 많이 익숙해졌고 conflict를 해결하고 merge를 수정할 수 있게 되었습니다.
    - 팀원 분들이 개발 환경 세팅을 해주셔서 EDA와 Feature Engineering 그리고 Modeling에 집중할 수 있어서 고마웠습니다.

▼ 이진원_T6199

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- EDA & Feature Engineering & Modeling & HPO를 모두 진행해보았다.

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

- 저번 대회에서는 DeepCoNN과 CNN_FM 모델 모두 ipynb 파일에 코드를 작성하여 모델링을 하였다. 이번 대회에서 SAINT+ 모델을 구현할 때 py 파일로 모듈화를 성공하였

고, yaml 파일에 파라미터들을 지정해주어서, 실험을 하는데 매우 용이하였다.

- 추가적으로, 저번 대회가 끝난 후 팀원들끼리 해보기로 약속한 pyenv와 poetry를 활용한 환경 세팅, Github 이슈 관리, PR 코드 리뷰, Wandb 연동을 모두 이루어내며 협업을 하는 데에 매우 도움이 되었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 딥러닝 기반 모델링을 많이 해보고 싶었지만, 아무래도 좀 더 익숙하고 많이 사용해본 Tree 기반 모델들에 비해 성능이 잘 나오지 않아서 아쉬웠던 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

- 이번 대회에서 2등을 해서 다시 발표를 할 수 있는 기회를 얻었다.
  저번 대회에서는 대부분 사용한 모델이 비슷하였지만, 이번 대회에서는 다양한 추천시스템 모델들에 대해서 많이 알게 되어서, 다음 프로젝트부터는 트리 기반 모델보다는 다양한 딥러닝 모델들을 꼭 사용해보고 싶다.