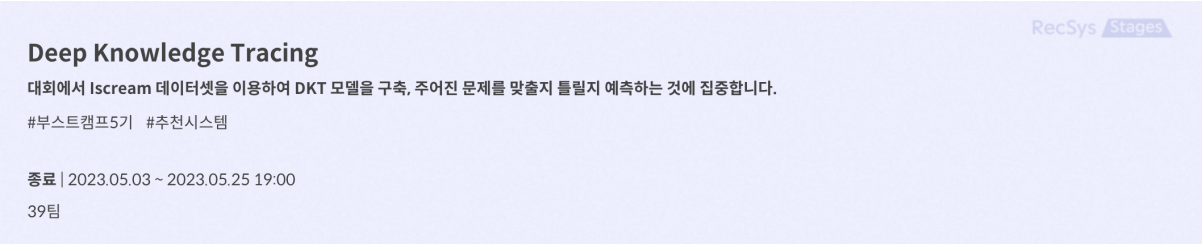
[Recsys 8조 EXIT] DKT Wrap Up Report

1.1 프로젝트 개요



프로젝트 주제

- 각 학생이 푼 문제와 정답 여부가 담긴 데이터를 바탕으로 최종 문제를 맞출지 틀릴지 예측하는 대회

  ※ DKT는 Deep Knowledge Tracing의 약자로 우리의 "지식 상태"를 추적하는 딥러닝 방법론

데이터 개요 및 구조

- 데이터 셋

train
userID: 사용자 고유번호 assessmentItemID: 문항의 고유번호 testId: 시험지 고유번호 answerCode: 해당 문제 정답 여부 Timestamp: 해당 문제를 풀기 시작한 시점 KnowledgeTag: 문항 중분류 태그

test
userID: 사용자 고유번호 assessmentItemID: 문항의 고유번호 testId: 시험지 고유번호 answerCode: 해당 문제 정답 여부 (각 유저의 마지막 시퀀스의 데이터 : -1) Timestamp: 해당 문제를 풀기 시작한 시점 KnowledgeTag: 문항 중분류 태그

실험 환경 및 사용한 툴

- (팀 구성 및 컴퓨팅 환경) 5인 1팀, 인당 V100 서버를 VSCode와 SSH로 연결하여 사용
- (협업 환경) GitHub, Notion
- (의사 소통) Slack, Zoom

1.2 프로젝트 팀 구성 및 역할

김지우_T5063	Boosting 기반 모델 적용, EDA 및 FE
박수현_T5085	Transformer w/o sequence 기반 모델 제작, FE
석예림_T5110	Git 프로젝트 설정 및 관리, Boosting 모델 적용, FE
임소영_T5172	Transformer w/ sequence 기반 모델 제작, LGBM 적용, FE

전증원_T5185	FM 기반 모델들 적용, FE

1.3 프로젝트 수행 절차 및 방법

1.3.1 Boosting Model

Boosting Baseline Code 작성

다양한 실험을 위해 3가지 부스팅 모델 (xgboost, lightgbm, catboost)을 돌려볼 수 있는 베이스라인 코드 작성

Data Set 구성

기존의 User 단위로 Split 되어 있던 Train, Test Data 재구성

- Test Data : Test에만 있던 userID에 대해 마지막으로 푼 문제
- Valid Data : Test Data를 Split하고 난 후에 모든 userID에 대해 마지막으로 푼 문제
- Train Data : Test, Valid Data를 제외한 모든 Data

모델 선정 이유

LightGBM

- 정형 데이터에서 특히 높은 성능을 내는 GBDT(Gradient Boosting Decision Tree) 모델 사용을 고려
- leaf-wise 트리 분할은 최대 손실 값(max data loss)를 가지는 leaf node를 지속적으로 분할하면서 tree의 깊이가 깊어지고 비대칭적인 tree가 형성되지만 level wise 트리 분할보다 예측 오류 손실을 최소화
- 기울기 기반 단측 표본추출(GOSS)과 배타적 변수 묶음(EFB) 을 통해 다른 gradient boosting model들에 비해 Training 및 Inference 속도 및 메모리 사용량 개선

EDA 및 Feature Engineering

문제 관련

문제의 종류와 난이도를 잘 나타내는 Feature를 생성하는 것이 가장 중요하다고 판단

- 문제 대분류
- 문제 번호
- 문제 위치
- 문항별 / 시험지별 / Tag별 / 대분류별 / 문제 번호별 Mean Encoding
- 시험지별 문제 수 / 태그 수

Time 관련

문제 풀이 시간이 정답에 기여하는 비중이 클 것이라고 판단하여 관련 Feature 를 다양하게 생성

- 문제 풀이 시간 ver1
  - 유저별로 diff 를 이용해 문제 풀이 시간을 초 단위로 계산
  - 만약 elapsed time 이 threshold 인 1000s 를 넘을 경우, test가 변경되어 새로운 문제가 시작되었음을 가정하여, 해당 시간을 0으로 초기화

- 문제 풀이 시간 ver2
  - 문제 풀이 시간 ver1 을 위로 shift 하여, 문제별 풀이 시간을 미세조정
  - test 내 마지막 문제의 경우 소요된 시간을 정확히 확인할 수 없기 때문에 같은 test 내 풀이 시간의 평균을 이용해 값을 할당
- Feature별 (KnoweldgeTag, assessmentItemID, 문제 번호) 모든 문제, 맞춘 문제, 틀린 문제별 풀이 시간의 평균
  - 각 Feature 별로 풀이 시간의 차이점과 영향력을 파악하기 위해 계산
- 유저 평균과의 시간 차이
  - 해당 문제 풀이 시간 - 해당 유저 문제 풀이 시간의 중위수
  - 해당 문제가 유저가 평소에 문제를 풀 때 걸리는 시간에 비해 얼마나 적게 걸리는지 혹은 많이 걸리는 지를 통해 문제의 난이도를 반영하고자 함
- 문제 정답 / 오답자들의 문제 풀이 시간 중위수
  - 문제 정답/오답자의 평균 풀이 시간을 유저가 해당 문제를 풀때 걸린 시간과 함께 모델에 넣어주고자 함
- week of year
  - 연도의 몇 번째 주차인지를 나타내는 변수로 시간적인 요소를 모델에 반영해주고자 함
- 이전의 정답 여부
  - 유저별로 n - step 이전의 정답을 맞췄는지를 통해 유저별로 최근 문제 정답 여부의 경향성을 반영해주고자 함

User 관련

유저의 학업 수준을 나타내는 변수를 생성하는 게 가장 중요하다고 판단

- 유저의 수준을 나타내기 위해 user의 정답률, 문제푼 횟수, 맞춘 문제수, 문제를 푸는데 걸린 시간 중위수, user별 푼 태그의 누적 개수를 변수로 추가
- 시간이 지남에 따라 변화하는 user의 수준(accumulative한 관점)을 나타내기 위해 user별 누적 정답률, 누적 정답 갯수, 누적 푼 문제수를 변수로 추가
- 대분류별로 정답률이 달라지는 것을 확인하여 user별로 대분류별 푼 문제 개수, 대분류별 맞춘 문제 개수, 대분류별 정답률을 변수로 생성

프로젝트 수행 결과

- Feature Engineering을 통해 다양한 Feature를 생성하여 LGBM 모델에 적용한 결과, 타 모델에 비해 우수한 성능을 내는 것을 확인
- 시간 관련 feature 가 주요 영향을 끼쳐 기존 auc 대비 큰 폭으로 상승하였음
- 특히 유의미했던 feature 였던 **이전의 정답 여부, 문제에 대한 정답자 / 오답자의 평균 문제 풀이 시간, 유저 평균과의 시간 차이, week of year** 추가 결과, Public auc 기준 0.8070에서 0.8225까지 성능을 올림

1.3.2 Transformer

프로젝트의 Performance Measure 선택

- CrossValidation을 고려하기 위해 Sklearn의 KFold를 활용
- CrossValidation이 적합한지 분석 중에 Random Sampling 기반 Hold-Out 기법도 충분하다고 판단

두가지 가설 기반 Transformer w/o sequence code 구현

- Positional Encoding 이 없는 Model
- Non-sequential masking
- 모델 스스로 Feature Engineering을 기대해볼 수 있도록 초기 레이어에 MLP 레이어를 추가

모델 선정 이유

- 과거 저의 수능을 준비했던 학창시절 경험을 통해 두가지 가설을 설정
  - Recent-Dependent
    - 현재 예측하고자 하는 문제에 유사도가 높은 문제는 최근에 풀었던 문제들에 있을 것
    - 인간의 기억력은 한계가 있기 때문에 최근 풀었던 문제들이 현재 예측에 영향을 많이 미칠 것
  - Permutation-Independent
    - 과거에 풀었던 문제들의 푸는 순서는 중요하지 않고, 현재 예측하고자 하는 문제에 유사도가 높은 문제를 최근에 풀었는지가 중요함
- Recent-Dependent를 잘 담아낼 수 있는 방법론이 Self-Attention이라고 생각했으며, Permutation Independent를 반영할 수 있도록 Transformer w/o sequence를 사용

Data Preprocessing & Engineering

- 문항별/테스트별/태그별 Mean Encoding(answerCode)를 추가해 성능 향상
- Sliding Window를 통해 Data Augmentation을 통해 성능 향상
- Test set을 통해 Data Augmentation을 통해 약간의 성능 향상
- 각 문제에 대해 풀이 시간을 제공하는 consumedTime Feature 추가
- 문항별/테스트별/태그별 Mean Encoding(consumedTime)를 추가해 성능 향상
- 유저별 Accumulated Mean Encoding(answerCode)를 추가해봤지만 성능변화 없음

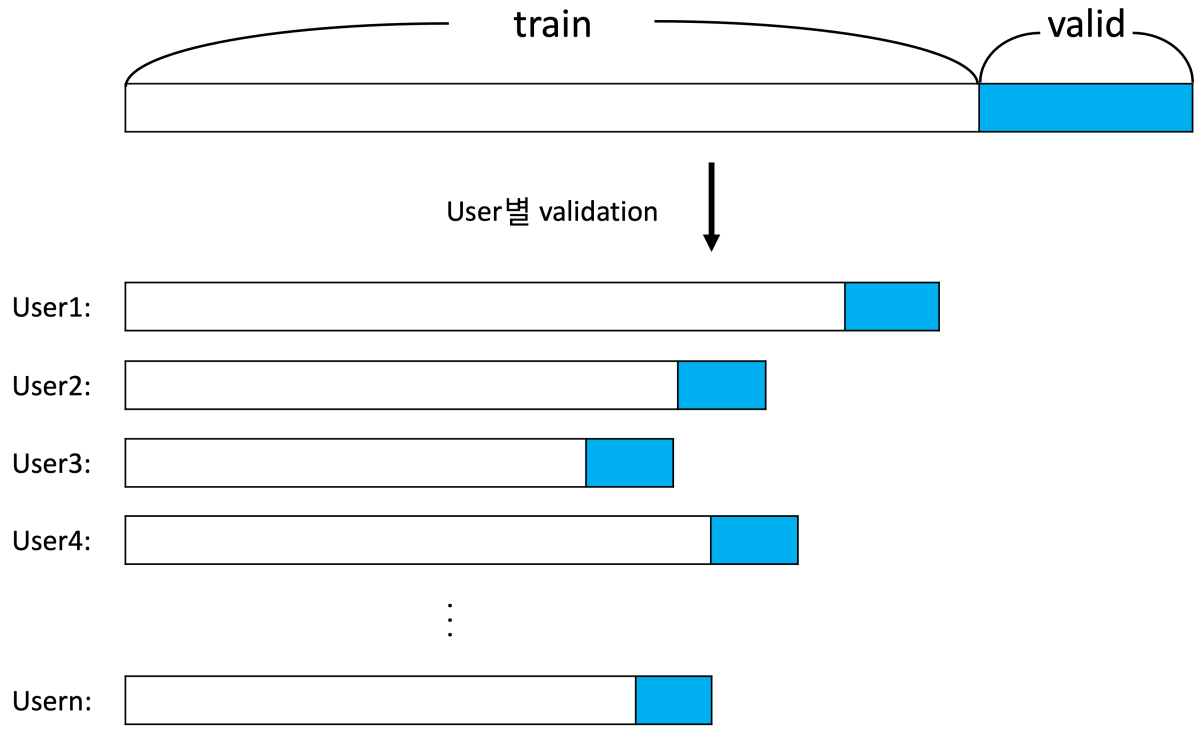
프로젝트 수행결과

Baseline Tranformer w/o sequence	0.7157
문항별/테스트별/태그별 Mean Encoding(answerCode) 추가	0.7157 → 0.7595
Sliding Window를 활용한 Data Augmentation	0.7610 → 0.7880
Test set를 활용한 Data Augmentation	0.7880 → 0.7903
문항별/테스트별/태그별 Mean Encoding(consumedTime) 추가	0.7903 → 0.8020
Model 초기 레이어에 MLP Layer 추가	0.8020 → 0.8040

1.3.3 FM Models

validation set 구성

- user별로 마지막의 interaction만을 validation set으로 구성
  - 이유: 데이터를 그냥 slice로 자르게 되면 validation set과 train set의 인원 구성이 달라지기 때문에 특정 유저에 맞춰져서 학습이 될 것을 우려



dataset 구성

- train과 test의 문제 종류, 유저의 종류가 달랐기 때문에 test에 있는 -1이 아닌 데이터를 모두 train에 붙여서 사용

```
drop_index_test = test[test['answerCode'] == -1].index
test2train = test.drop(drop_index_test)
train = pd.concat([train, test2train])
```

모델 선정 이유

모델	선정 이유
Factorization Machine	tag, grade, time 데이터 등 implicit feedback 데이터를 사용할 수 있기 때문 유저와 문제간의 관계를 표현한 one hot 벡터가 희소하기 때문
FFM	feature간의 연관성이 다 다르므로 이를 표현해주기 위해 FFM 사용
DeepFM	낮은 수준의 상호작용은 FM으로 표현하고, 높은 수준의 상호작용은 DNN으로 표현하기 위해 DeepFM 사용

data processing

feature	상세내용
time divide	시간대별로 정답률의 차이가 있을 수 있다고 예측하고, 시간대를 오전, 오후, 밤, 새벽으로 4개의 시간대로 나눔
grade	시험지 태그의 2번째 index별로 정답률의 차이가 났기 때문에 시험지 태그의 2번째 index를 grade

	로 나눔
assessment item number	문제의 번호가 증가할수록 문제의 난이도가 높아질것이라 예측하고 문제의 번호를 새로운 feature로 생성
tag count	사용자가 저번에 봤던 tag이면 정답률이 높아질것이라 생각하고 이번이 몇 번째로 본 동일한 태그인지 새로운 feature로 생성
knowledge tag	knowledge tag별로 정답률의 차이가 있을것이라 생각하여 knowledge tag별로 나눔

프로젝트 수행 결과

	public→ private(auroc)
FFM	0.7380 → 0.6811
FFM + data processing	0.7518 → 0.7343
DeepFM + data processing	0.7645 → 0.7686

1.3.4 성능 고도화

1. 학습된 모델간에 Correlation을 고려해서 Ensemble 시도
2. OOF(Out-of-Fold) + Optuna with HyperParameter Tuning
3. Soft Voting의 Ratio 결정

LGBM	Transformer w/o sequence	DeepFM
0.9	0.05	0.05

1.4 프로젝트 수행 결과

public 2등

- auroc (0.8243) / accuracy (0.7634)

순위	팀 이름	팀 멤버	auroc	accuracy	제출 횟수	최종 제출
2 (-)	RecSys_08조	수현 증원	0.8243	0.7634	55	3d
1	Recsys_05조	박 은비 A	0.8273	0.7446	128	3d
2	RecSys_08조	수현 증원	0.8243	0.7634	55	3d
3	RecSys_01조	상우 미노	0.8231	0.7500	106	3d
4	Recsys_03조		0.8227	0.7473	84	4d
5	RecSys_04조	지수 도현	0.8225	0.7446	64	3d

private 1등

- auroc (0.8592) / accuracy (0.7823)

순위	팀 이름	팀 멤버	auroc	accuracy	제출 횟수	최종 제출
1 (1 ▲)	RecSys_08조		0.8592	0.7823	55	3d
1	RecSys_08조		0.8592	0.7823	55	3d
2	RecSys_07조		0.8579	0.7903	66	3d
3	Recsys_03조		0.8577	0.7742	84	4d
4	RecSys_11조		0.8549	0.7715	141	3d
5	RecSys_01조		0.8464	0.7796	106	4d

1.5 자체 평가 의견

잘했던 점

- 각자 써보고 싶은 모델에 대해 많은 실험을 진행해서 좋았음
- 성적보다 대회를 통한 성장에 집중했음
- 특이사항이 생길 때마다 신속하게 공유받아서 좋았음

아쉬웠던 점

- 스케줄링에 실패해서 길었던 대회 기간을 온전히 활용하지 못해 아쉬웠음
- 부족한 시간으로 인해 Git(issue, pull request)을 잘 활용하지 못했음
- 이번 프로젝트와 유사했던 사례를 충분히 검색해보지 못했음

배운점 & 시사점

- 학습된 모델간에 Correlation의 차이가 있는 경우 적극적으로 Ensemble을 시도해볼 필요가 있다는 것을 알았음
- OOF(Out-of-Fold) + Optuna 조합이 항상 성능 향상을 이끌어내는 것이 아니라는 것을 깨달았음

2.1 개인회고

김지우_T5063

- **나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?**
  - 제대로 된 협업을 경험해보고 싶었습니다. 또한, 주어진 베이스라인 코드를 돌려보면서 실험하는 것이 아니라 직접 데이터셋, 데이터 로더부터 시작해 모델까지 코드를 작성해보고 싶었습니다.
  - 목표를 달성하기 위해, 팀만의 Git 활용 Rule을 정하고 규칙에 맞게 깃을 활용해 협업하기 위해 노력했습니다. 데이터 셋을 전처리하고 Feature Engineering한 후 모델링하는 코드를 직접 작성해보았습니다.

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

- 코드를 모듈화 시키고, 타입 힌트와 주석을 적절히 사용함으로써 원활한 협업을 가능하게 했습니다. 이를 통해 가독성이 높고 이해하기 쉬운 코드를 작성하는 것이 중요하다고 느꼈습니다.
- target 변수와 다른 Feature와의 관계를 분석하면서 유의미한 Feature를 추가하는 것이 모델의 성능 향상에 크게 기여할 수 있다는 것을 깨달았습니다.
- 성능차이가 많이 나더라도 correlation이 낮은 모델끼리 ensemble 하는 것이 효과적인 전략이라는 것을 알게 되었습니다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  - 코드를 짰는데, 타입 변환을 하는 함수를 잘못 작성해서 모델이 제대로 학습하지 못하는 문제가 발생했습니다. 이유를 찾는 과정에서 시간을 너무 많이 사용해서 성능을 올리기 위한 여러가지 시도를 해보지 못한 점이 아쉬웠습니다.
  - 일정 관리에 실패해서 각자 하나의 모델만 맡을 수 있었던 점이 아쉬웠습니다. 시간이 된다면, 공유를 통해 직접 맡지 않은 모델에 대해서도 자세하게 이해해보고 싶습니다.
  - 코드를 짤 때, 항상 데이터 전처리부터 모델 학습 및 추론까지 한번에 진행되도록 해놓으니 오히려 시간이 더 걸리는 문제점이 발생했었습니다. 데이터를 처리하는 부분을 따로 분리해서 중복되는 작업을 하지 않도록 처리하면 더 좋았을 것 같습니다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**
  - 코드 리뷰를 통해 서로의 코드를 한번씩 보면서 의도한 대로 코드가 잘 짜여졌는지, 어떻게 하면 더 깔끔하게 코드를 짤 수 있을지 이야기를 나누는 시간을 가져보고 싶습니다.
  - 효율적인 일정 관리를 통해 각자 하나 이상의 모델(ML,DL)을 맡아서 작업하고, 팀원들에게 각자 맡은 모델을 설명해주는 시간을 가지면 좋을 것 같습니다.
  - 데이터 버저닝을 시도하고 싶습니다. 버저닝을 사용해서 데이터의 변경 이력을 추적하고, 필요한 경우 이전 버전으로 되돌릴 수 있게 하여 효율적인 협업과 데이터의 일관성 유지에 신경쓰면 좋겠다고 생각했습니다.

박수현_T5085

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**
  - 데이터/근거 기반 의사결정을 내리도록 노력하기
  - EDA 및 Modeling을 열심히 수행하다보면 좋은 성적은 따라오는 것으로 굳게 믿고 Public/Private Score에 매몰되지 않기

- **나는 어떤 방식으로 모델을 개선했는가?**
  - Sliding Window를 활용한 Data Augmentation을 진행함
  - 예를 들어 max_seq_len를 20으로 설정할 경우, 문제를 1800개를 푼 유저에 경우 1780개가 Drop되는 문제가 있었음
  - 1800개를 20개씩 끊어 학습 데이터로 활용할 수 있도록 Sliding Window를 활용함
  - 모델이 문제별 난이도를 유추하여 활용할 수 있도록 Mean Encoding을 적극 활용함

- 문제마다 난이도가 상이하고, 이러한 난이도를 모델 학습에 활용할 수 있으면 성능이 향상할 것으로 기대했음
- 문제별 난이도를 유추할 수 있도록 문제마다 정답비율/풀이시간 정보를 추가했음

- **내가 한 행동의 결과로 어떤 지점을 달성하고 어떠한 깨달음을 얻었는가?**
  - 특정 행동을 통해 달성한 것
  - 문항별/테스트별/태그별 Mean Encoding(answerCode) 추가(0.7157 -> 0.7595)
  - Sliding Window를 활용한 Data Augmentation(0.7610 -> 0.7880)
  - 문항별/테스트별/태그별 Mean Encoding(consumedTime) 추가(0.7903 -> 0.8020)
  - Model 초기 레이어에 MLP Layer 추가(0.8020 -> 0.8040)
  - 달성한 점을 통해 깨달은 점
  - Model Structure를 커스터마이징 하는 것보다 Model에 Feeding 하는 데이터의 퀄리티를 높이는 것이 일반적으로 더 좋은 성능 향상을 이끌어 낼 수 있음을 깨달음

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떠한 효과가 있었는가?**
  - 데이터 기반 의사결정을 내릴 수 있도록 노력함
  - 이러한 의사결정을 내릴 수 있도록 EDA -> Action -> Verify를 진행할 수 있는 환경을 만드는 것에 초점을 맞춤
  - 이러한 환경에 요구되는 조건인 Baseline Model 구현 및 Performance Measure 설정이었기 때문에 프로젝트 초기에 설정하는데 시간을 활용함
  - 데이터 기반 의사결정을 통해 팀원들에게 진행상황을 쉽게 납득시킬 수 있어 매우 유용했음
  - 설정한 가설들을 기반해서 모델을 선정함
  - Level1에서 진행했던 프로젝트와 달리 확실한 근거를 가지고 모델을 선정함
  - 과거 학창시절 경험을 기반해 두가지 가설(Recent-Dependent / Permutation Independent)를 설정해서 Self-Attention Mechanism을 선정해 Transformer Encoder를 사용하기로 결정했음

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  - 유사한 사례 및 SOTA 모델에 대해서 검색을 하지 않음
  - 딥러닝 프로젝트를 시작하기에 앞서 유사한 사례 및 SOTA 모델을 찾아보는 것이 가장 기본임
  - 설정한 가설들을 기반해 모델 선정에만 그치는 것이 아니라 진보된 모델(ex> saint)를 검색해서 활용해봤으면 더 좋은 성능을 이끌어 낼 수 있었을 것 같음
  - Optuna + OOF 전략에 대해 크게 맹신함
  - Fold 별로 데이터가 유사할 경우 OOF 전략을 통해 성능 향상 없음
  - Optuna가 항상 Optimal Solution을 도출해내는 것은 아니라는 것을 깨달음

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?**

- WandB 사용해서 인사이트 얻어보기
- 유사한 사례 및 SOTA 모델에 대해서 충분히 검색해보기
- Importance 기반 Recursive Feature Selection 활용하기
- Optuna + Grid Search를 활용해서 HyperParameter Tuning

석예림_T5110

- **나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?**
  - 협업을 원활히 진행

    팀만의 Git rule을 정하였고, Git을 통해 각 기능에 따른 branch를 생성하고 Github을 통해 공유함
  - 부스팅 모델에 대해 모델 및 Feature Engingeering 진행

- **나는 어떤 방식으로 모델을 개선했는가?**
  - Feature Engineering을 통해 모델을 개선하려고 노력하였다.
    특히 ELO 평점을 도입하여 개선하려고 노력했었는데, `assessmentItemID`, `testId`, `KnowledgeTag` 기반 elo_score의 열을 추가하였다. 하지만 `assessmentItemID` 기반 elo_score는 성능이 개선되었지만 나머지 Feature에 대해서는 결과적으로 개선하지 못했다.

- **내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?**
  - Git - 프로젝트 코드 관리 및 협업에 있어서 커밋과 브랜치를 관리하여 Git의 유용함을 경험하고 숙련도를 쌓았다.
    - Git flow를 적용해 현업에서 사용하는 Git 사용법을 익혔다.
    - commit convention을 통해 commit 메시지를 통일시켰고, 어떠한 목적의 commit인지 확인하기 쉽도록 하였다.
    - slack의 알림 기능을 통해 놓치지 않고 팀원의 진행상황을 공유하도록하였다.
    - Project를 통해 issue기반 해야할 일을 나열하고, 완료된 작업과 해야할 작업을 구분하여 프로젝트의 진행상황을 한눈에 보기 쉽도록 하였다.
      - 팀원 각자 맡은 작업이 무엇인지 쉽게 파악할 수 있었다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  - 익숙하지 않은 Git을 도입하기 위해 많은 시간을 소비하였고, 코드 공유가 이루어졌지만, 대회 시간이 촉박하여 Git을 통한 공유가 원활하게 이루어지지 않았다. PR을 활용하지 않아 구두로 code의 input, output 등을 설명하며 코드리뷰를 진행하였다.
  - 또한 wandb를 활용하여 실험 결과를 공유하고 싶었지만, 새로운 도구를 도입하기에 시간이 부족하였다.
  - 이해하고 사용한 것이 아니라 결국 ELO feature에 대해 모델에 사용하지 못했었다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?**

- 다음 프로젝트에는 협업 툴 환경세팅에 너무 많은 시간을 투자하지 않고, 습관처럼 사용하도록 익숙해 지도록 하겠다.
- 모델을 많이 작업하지 못했던 관계로 다음 프로젝트에서는 직접 모델을 작성하고 많은 실험을 진행해 보고 싶다.

임소영_T5172

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  지난 대회 때, model 의 하이퍼파라미터나 다양한 기본 로직들을 조금씩 수정해가며 성능을 향상시키지 못해 아쉬웠기에 이번 대회 학습목표는 model 을 직접 만져보며 성능을 향상시킬 수 있는 방법에 대해 깊게 고민해 보는 것이었다. sequential data 를 다루는 model 의 성능이 보다 높게 측정되었기에, graph 계열 model 은 깊게 만져볼 수 있는 시간이 적었으나 좁혀진 길 덕분에 sequential data 를 처리하는 transformer 계열 model 에 대해 보다 더 진지하게 임할 수 있었다. Transformer 에 대해 복습하는 시간을 가지고, 코드를 디테일하게 분석하고 개발하였다.

- **나는 어떤 방식으로 모델을 개선했는가?**

  positional embedding 을 포함하는 모델과, 포함하지 않는 모델을 concat 하여 결과를 내는 GTN 모델을 새롭게 개발해보고, 그 과정 속에서 block 별로 모든 positional embedding 을 고려하는 모델도 고안해내여 더 디테일하게 position 값을 익힐 수 있도록 노력하였다.

  그러던 중, transformer 관련 업무가 다른 팀원에게 인계됨에 따라 boosting 모델을 새롭게 맡게 되었고, LGBM 에 알맞는 feature 들을 제거하고, 추가해보면서 성능이 align 되는지 확인하였다. 그 과정 속에서 feature engineering team 의 features 중 어떤 feature 가 중요한 역할을 하는지 깨닫고, 더 디테일하게 파고들어 feature 생성 로직 중 오류가 있는 부분에 대해서는 새로운 방안을 제시하는 등 feature 를 정교하게 다듬었다.

- **내가 한 행동의 결과로 어떤 지점을 달성하고 어떠한 깨달음을 얻었는가?**

  LGBM 의 성능이 가장 높아 다른 모델과의 앙상블에 항상 사용되었다. 단일 모델로도 성능이 꽤 높게 나왔으나 correlation 이 낮은 모델과의 앙상블을 이용해 오버피팅을 방지하고자 하였다. Transformer / DeepFM / XGBoost / CatBoost 등 팀원들이 개발한 모델과의 앙상블을 통해 private data 로 1등을 달성할 수 있었다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떠한 효과가 있었는가?**

  모델 파트를 새롭게 맡아보고, 주도적으로 실험을 진행한 것. 원래는 모델에 대해 공포감이 큰 편이었는데, 막상 이것저것 만져보고 설정도 새롭게 해보고, 코드도 짜보니 별거 아니라는 생각이 들었다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  성능상, sequential data 를 고려하지 않는 graph 계열보다, 고려하는 transformer 계열이 더 좋았기에, 초기 도전부터 graph 계열 모델을 모두 포기하고 transformer 에 모든 사력을 다한 것이 아쉬웠다. graph 특성 상, edge 로 표현할 수 있는 것이 적다보니 여러 interaction 중 유의미한 것만 남기는, 새로운 방식의 edge 표현법도 고안해보고 싶었는데 시간상 / 능력상 시도해보지 못한 idea 가 많은 것 같아 아쉬웠다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?**

  baseline 코드 기준으로 성능이 낮다고 해서, 선택과 집중이라는 명목 하에 아예 한 쪽으로 몰아서 생각하기보다는 더 넓게 사고할 수 있도록 노력해야겠다. 또한 이 프로젝트를 모르는 사람들에게도 원활히 설명할 수 있도록, 프로젝트에 대한 이해도를 더 높이기 위해 다음 프로젝트 때에는 솔루션 발표도 해보고 싶다.

전증원_T5185

- **개인 목표**
  - FM 모델을 DKT대회에 적용해보기
  - Git을 사용하여 협업해보기

- **나는 어떤 방식으로 모델을 개선했는가?**
  - FM에 관련된 지난 강의들을 들으며 FM 모델에 대한 개요, 감을 잡기 위해 노력했다.
  - 다른 사람들이 구현해놓은 FM을 보며 참고하였다.
  - FM을 구현하고 그 후 FFM, DeepFM등으로 점차 성능을 높이기 위해 노력했다.

- **내가 한 행동의 결과로 어떤 지점을 달성하고 어떠한 깨달음을 얻었는가?**
  - 지난 강의들을 들으며 기초가 가장 중요하다는 점을 깨달았고, 새로운 것들을 배우기 전에 기초적인 부분을 먼저 점검해야겠다는 생각이 들었다.
  - 다른 사람들이 구현해놓은 FM을 보며 이해가 안되는 부분을 보며 pytorch에 대한 이해가 더욱 필요하다는 것을 깨달았다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**
  - 첫번째 대회에서는 MF모델을 다뤘는데, 구현상 오류가 있었는지 제대로 성능이 나오지 않았다. 하지만 이번 대회에서는 첫번째 대회에 비해 시간적인 여유가 있었기 때문에 모델에 대해 대략적으로 이해를 한 후 적용을 하여 dataset 구성부터, 모델까지 제대로 적용을 할 수 있었다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  - 모델을 논문을 보며 직접 구현한게 아닌, 남이 구현한 FM모델을 DKT 대회의 data에 알맞게 변형시킨점이 아쉬웠다.
  - 프로젝트때 Git을 사용하기로 했는데 아예 사용하지 못했다.
  - Data processing 과정에서 가정을 하고 증명을 한 후 data processing을 진행해야 하는데, 가정만 하고 증명을 하지 않고 data processing을 진행하였다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**
  - 논문을 바탕으로 직접 모델을 구현해보고 싶다.
  - 다음 프로젝트에서는 협업 프로세스를 바탕으로 Git을 사용해보고 싶다.

- 다음 프로젝트에서는 data의 확실한 경향성을 확인한 후 data processing을 진행해야겠다.