Deep Knowledge Tracing Wrap up Report

Recsys 7조 유쾌한발상
이채원, 김소미, 서현덕, 백승주, 유종문

1. 프로젝트 개요

A. 개요

Deep Knowledge Tracing이라는 지식 상태 추적 방법론을 사용하여 각 학생의 과목별 이해 수준을 파악할 수 있는 모델을 학습하는 것이 프로젝트 목표이다. 모델을 평가하기 위해 학생별로 특정 문제에 대한 정답 여부를 맞추도록 하고, 실제 값과 ROC AUC를 계산하도록 했다.

B. 활용 장비(도구)

- 개발환경: VSCode를 사용한 UpStage GPU(V100)와 SSH로 연결.
- 협업툴: GitHub, wandb, Jira, Confluence
- 의사소통: 카카오톡, Zoom, 필요시에 오프라인으로 프로젝트 진행

C. 기대 효과

- 사용자의 학습 데이터를 바탕으로 적합한 난이도의 문제를 추천할 수 있다.
- 사용자에게 최적의 커리큘럼을 제공할 수 있다.

2. 프로젝트 팀 구성 및 역할

전체	문제 정의, 계획 수립, 목표 설정, EDA, Feature Engineering, 모델 실험
이채원	Feature Selection, NMF, TabNet, outlier 분석, Ensemble, Hyper parameter exploration
유종문	모델 탐색, 데이터 분할 전략, XGBoost 구현, Soft Voting, 모델 평가기준 생성
김소미	Feature Selection, Permutation Importance, Wandb 시각화, CatBoost 구현
서현덕	Feature Selection, Cross validation, 가설 설정 및 검증
백승주	모델 탐색, Transformer 계열 모델 개선, Augumentation, LGBM 구현, 데이터 분할

3. 프로젝트 수행 절차 및 방법

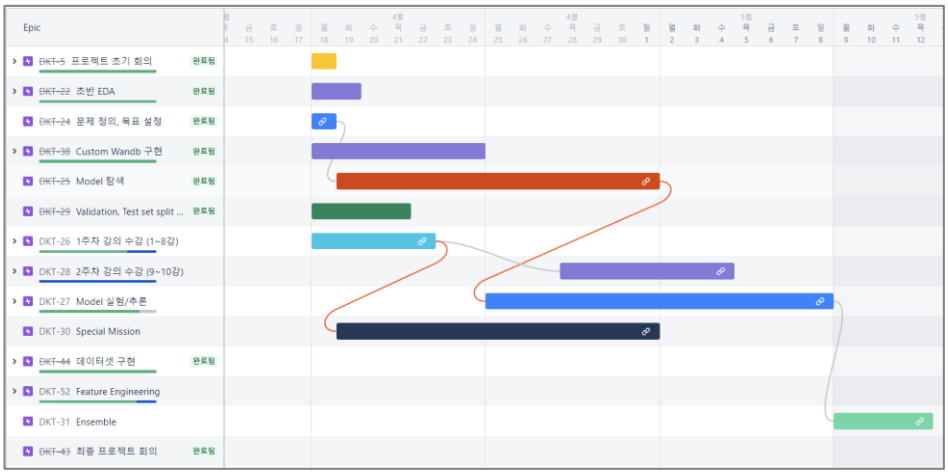
A. 목표 설정

① 리더보드 Top2 진입
② Wide & Deep 하게 모델 탐색 및 실험
③ 프로젝트를 체계화하여 관리

B. 프로젝트 사전 기획

① 프로젝트 일정

1주차: 문제 정의, EDA, 모델 탐색, 검증전략 수립, 프로젝트 작업 환경 설정 (Jira, Confluence, Github, Wandb, …)
2주차: 모델 구현 및 탐색, Feature Engineering
3주차: 모델 실험, Feature Engineering
4주차: Feature Selection, Ensemble



② 역할 분담

EDA를 진행하고자 하는 주제들을 모두 리스트업하고 각자 나누어서 분석한 결과를 공유하였다. 이후에 EDA 결과를 바탕으로 feature engineering을 분할하여 진행하였다.

③ 대회기간에서 사용할 그라운드 룰 생성

(1) 기한 내에 못할 것 같으면 미리 말하기
(2) 한 방법을 다 같이 고민해 보고 안되면 다음 방법으로 넘어가기
(3) 언제든 도움 요청하기
(4) 개선할 점과 단점을 이야기할 때 기분을 생각해서 이야기하기
(5) 코딩 컨벤션 정하기
(6) 업무 현황을 잘 공유하고 소통하기
(7) jupyter notebook 파일 업로드 시, README와 markdown으로 설명 추가하기
(8) 실험 기록 관리, model 파일(.pth, .pt), probability 파일 관리 및 경로 통일
(9) 플랫폼 관리자를 두어 각자 맡은 플랫폼 관리 진행
(10) 서로 올린 코드 리뷰하기

④ GitHub 버전 관리 규칙

1. **Commit** 시, 다음과 같은 Header 양식을 사용하기로 하였다 : [TYPE][Author] contents (#issue number)
   TYPE 종류: FEAT, FIX, DOCS, STYLE, REFACTOR, TEST, CHORE
2. 사용할 **Branch**와 각 이름은 다음과 같이 정하였다: feature/{기능}, hotfix/{#issue}
   사용 Branch: main, develop, feature, hotfix

C. EDA

① 문제 난이도 분석
② 학생 성적 분석
③ 순서 분석
④ Outlier 이상치 탐지
⑤ Train data와 Test data의 분포 차이
⑥ User의 실력에 따른 정답률 분석

Link1: https://somi198.atlassian.net/wiki/spaces/DKT/blog/2022/04/21/1736731/EDA
Link2: https://somi198.atlassian.net/wiki/spaces/DKT/pages/98305/Exploratory+Data+Analysis

D. Data Processing

[Augmentation]	Sliding Window, Pseudo Labeling
[Preprocessing]	Random Shuffle, Answer Rate, User별 문제풀이가 적은 경우 제거

E. Feature Engineering (github link)

Original	UserID, assessmentItemID, testID, KnowledgeTag
Static	assessMean, assessSum, assessStd, testMean, testSum, testStd, tagMean, tagSum, tagStd
Bigclass	bigClass, bigClassCount, bigClassAcc, bigClassCumAcc, bitClassAccCate, bigClassElapsedTimeAVG
AnswerCode	recAccuray, recCount, cumAccuracy, cumCorrect, accuracy, totalAnswer, seenCount
Time	day, month, year, wday, hour, weekNum, elapsed, elapsedTimeClass
KnowledgeTag	KTAcc, KTAccuracyCate, tagCluster, tagCount, userLVbyTag, userLVbyTagAVG, tagLV, tagClass
TestId	testLV, userLVbyTest, userLVbyTestAVG
Diffficulty	elo, eloTag, eloTest

F. 모델 개요

Sequence / Transformer	LSTM-ATTN, BERT, Saint, LastQuery Transformer, Long-Short Term Memory (LSTM)
Boosting	Light Gradient Boosting Machine (LGBM), XGBoost, CatBoost
Graph	Light Graph Convolution Network (LGCN)
Collaborative Filtering	NMF, Multi-VAE
Rule Based	bigClassAccuracy, KnowledgeTagAccuracy

G. 모델 선정 및 분석

(1) Seq / Transformer 계열

DKT baseline과 과제로 주어진 모델들로 사용자가 푼 문제의 sequence를 모델에 학습시킬 수 있었다. RNN 계열의 모델에 attention을 적용한 모델들의 높은 성능을 보여주었다. 특히 LSTM에 Attention을 적용시킨 LSTM-Attention 모델이 빠른 학습과 높은 AUC를 보여주었다.

(2) Boosting 계열

Feature Engineering이 되고 나서 LGBM의 성능이 먼저 가장 좋게 올라갔다. LGBM은 특정 Feature에 대해서 overfitting이 되는 경향을 보여주었고, 이후에 사용한 CatBoost에서는 overfitting을 방지해 주어 더 나은 성능을 보여주었다. 마지막으로 실험한 Boosting 모델은 XGBoost였고 Overfitting의 위험성은 있지만, LGBM보다 전반적으로 더 나은 성능을 보여주었다. 전체적으로 CatBoost를 제외하고 overfitting의 위험성이 큰 모델들이기에 hyper paramater tuning이 중요하게 작용하였다.

(3) Graph 계열

Graph 모델은 Light GCN을 사용하였다. 기존 GCN에 비해서 모델 구조는 훨씬 단순하여 computation 시간은 적지만, 성능은 높아서 선택하게 되었다. 결과적으로 AUC 점수가 0.8에 근접했지만 다른 feature의 정보를 사용할 수 없다는 것이 큰 단점으로 와닿았다.

(4) CF 계열

상호작용 이외의 다른 feature들을 반영하기 힘들었지만 그런 점을 반영하지 못함에도 꽤나 좋은 성능을 보였다. 0.73 정도였지만, feature를 추가할 수 없어서 많은 실험을 할 필요는 없었다.

(5) Rule-Based

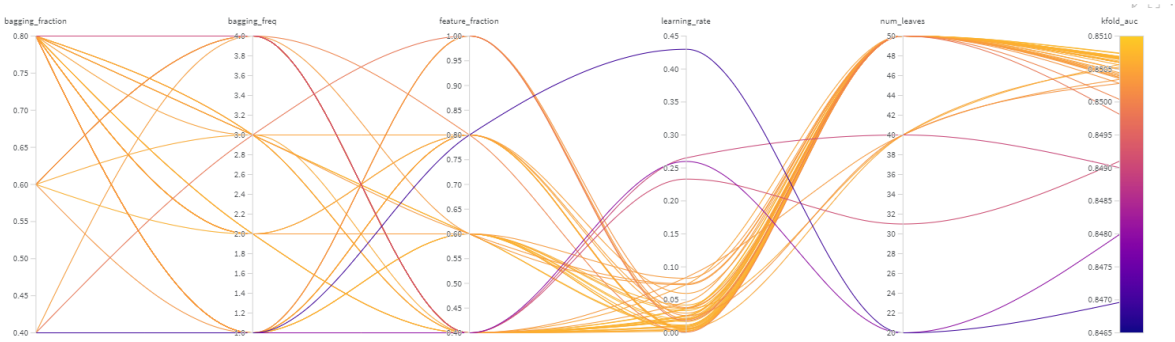
정답 여부에 직접적으로 연관성이 있다고 생각한 학생별 과거 대분류(또는 KnowledgeTag) 정답률을 정답으로 그래도 사용한 bigClassAccuracy, knowledgeTagAccuracy 모델들은 baseline 성능인 ROC AUC 0.5보다 더 높은 0.7 정도의 성능을 보였다.

H. 협업 과정

① Git의 활용

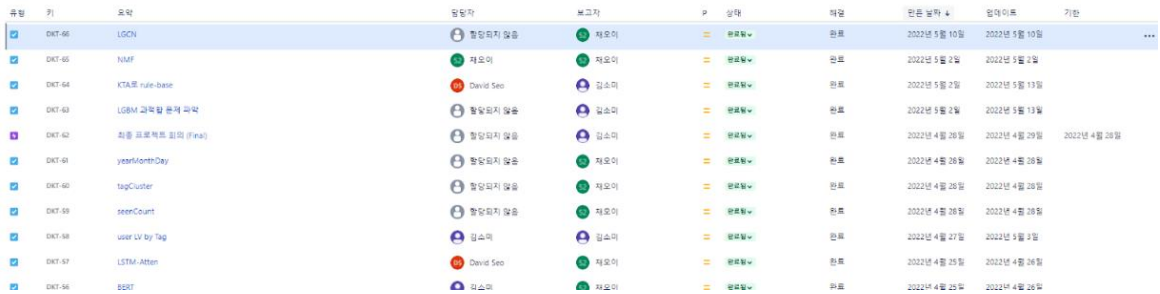
GitHub의 branch를 적극적으로 활용해서 각 모델 간의 실험을 독립적으로 설정하였고, 모든 팀원들이 실행한 코드들을 github를 통해 공유하며 협업을 할 수 있었다. 필요한 경우, pull request, merge conflict를 하여 더 빠른 코드 교환이 이루어질 수 있게 되었다.

② 실험관리를 위한 wandb



모델의 구조가 복잡한 transformer 계열의 모델들이나 Boosting 계열의 모델들은 overfitting의 위험성이 컸다. 그렇기에 최적의 파라미터를 찾는 과정이 필수적으로 요구되었고, wandb의 sweep 기능을 통해 가장 좋은 파라미터를 찾도록 하였다. 이뿐만 아니라 model.pt 와 같은 파일들도 wandb 공유 서버에 저장하여 모두가 실험 재현을 할 수 있게 되었다.

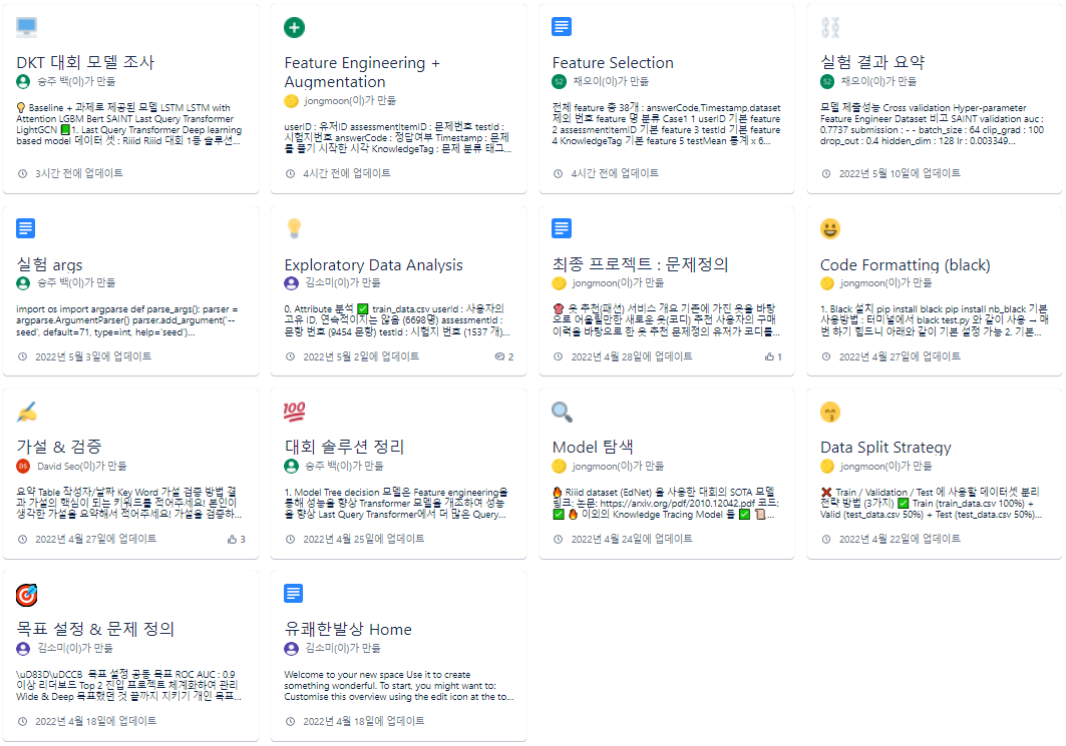
③ 일정관리를 위한 Jira



Jira의 로드맵과 보드를 활용하여 개인별 Epic(Main task)과 하위 이슈들을 쉽게 정리하고 관리할 수 있었다. 로드맵의 경우 전체 일정이 bar 형태로 한눈에 보여 어떤 태스크가 진행이 안 되고 있는지 파악할 수 있었고, 팀원들이 현재

어떤 작업을 진행하고 있는지 실시간으로 확인 가능하였다.

④ 다양한 실험정리, 인사이트 공유를 위한 Confluence Wiki



Confluence wiki를 통해서 각자 조사한 자료나 인사이트 또는 실험 결과들을 효과적으로 정리하고 공유할 수 있었다. Jira와 연동하여 일정에 맞게 진행사항을 확인할 수 있었고, 쉽게 정리하는 데 도움이 많이 된 플랫폼이다.

4. 프로젝트 수행결과

A. 시연 결과

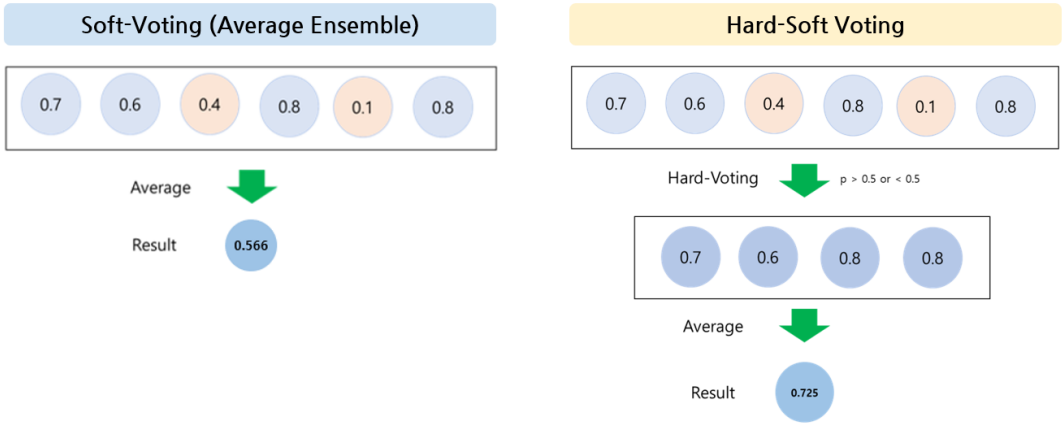
Model	Augmentation / Skills	AUCROC (제출)	Accuracy (제출)
LSTM	Sweep	0.7581	-
LSTM Attention	Sliding window, Continuous feature, k-fold	0.8499	0.7849
Last Query Transformer	Sliding window, Continuous feature, k-fold	0.8467	0.7339
Bert	Sliding window, Continuous feature, k-fold	0.8212	0.7366
Saint	Sliding window, Continuous feature	-	-
LGBM	Feature selection	0.8400	0.7419
XGBoost	Feature selection	0.8571	0.7715
CatBoost	Feature selection	0.8504	0.7608
LightGCN	-	0.8425	0.7608

B. 앙상블

독립적으로 실험한 모델들에 대해서 2가지 앙상블 기법을 적용하여 성능을 끌어올리도록 하였다. 이번 대회에서 사용한 앙상블 방법은 다음과 같다.

① Soft Voting: Inference 값의 평균을 취하는 앙상블 방법

② Hard Soft Average Voting: 특정 threshold를 기준으로 하여 더 많이 voting된 값들에 대해서만 평균을 취하는 앙상블 방법



[2가지 앙상블 기법 도식화]

앙상블은 일반 Soft-Voting으로 Bert, CatBoost, LGBM, LGCN, LSTM-Attention, XGBoost를 사용한 것이 최종 제출인 0.8590으로 가장 높았다.

C. 프로젝트 결과

2 (1 ▲)	RecSys_07조		0.8590	0.7823	137	21h

전체 16팀 중 2위

5. 자체 평가 의견

A. 좋았던 점

① 지난 프로젝트 회고를 통해 부족했던 부분을 채워 넣을 수 있었던 것

② EDA를 충분히 한 것

③ Jira, Confluence 등의 협업 툴을 적극 활용한 것

④ Sweep을 통해 hyper parameter tuning에 대한 궁금증을 푼 것

⑤ 데이터 분할을 통해 일반화 검증을 시도하려고 한 것

⑥ 팀원이 최종 결과에 모두 균등하게 기여한 것

⑦ 목표를 달성한 것

⑧ 다양한 feature를 설계해 보고 실험 최고의 성능을 내기 위해 실험해본 것

⑨ Github branch를 통해 코드와 결과를 빠르게 공유한 것

⑩ Python black formatter를 사용한 것

⑪ Feature 관리를 체계적으로 한 것

B. 아쉬웠던 점

① Boosting model로 실험을 많이 진행했는데, 특정 Feature에 대한 Overfitting 원인을 완벽하게 찾지 못한 것

② 강의에 나온 많은 기법을 다 적용해 보지 못한 것

③ Git issue를 사용해 보지 못한 것

1. 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

A. 우리 팀과 나의 학습목표는 무엇이었나?

이번 프로젝트에서 팀의 목표는 리더보드 Top2에 들어갈 수 있을 정도로 열심히 대회를 수행하고 Wide하고 Deep 하게 모델을 실험해보는 것이었다. 그리고 최종프로젝트를 수행하기 전 프로젝트를 체계적으로 관리하는 연습을 해보고자 하였다.

B. 개인 학습 측면

기본기의 향상: 저번 대회를 수행하며 코드를 만드는데 기본기가 부족해 많은 시도를 해보지 못하였다. 모델의 구조와 입력 출력 차원을 맞추는데 시간을 많이 허비한 것이 아쉬웠다. 이번 대회에서는 기본기에 충실하기 위해 노력하였고, 파이썬 문법, pytorch, pandas, numpy 사용법을 복습하면서 기본기를 향상시키고자 노력했다.

베이스 라인에 대한 이해: 저번 대회에서는 주어진 베이스라인의 구조를 거의 이해하지 못해서 거의 활용하지 못했기 때문에 주어진 코드를 최대한 이해해보려 했다. 그 결과 베이스라인 코드를 수정하여 다양한 실험을 할 수 있었다.

모델에 대한 이해: 이번 대회에서 Sequence, Tabular, Graph 형태의 데이터를 다루기 위한 모델을 보다 잘 이해할 수 있게 되었다.

C. 공동 학습 측면

대회 초반 Wandb를 사용해 실험을 관리하는 것을 다같이 학습하였다. 보다 체계적으로 실험 결과를 관리할 수 있었고 Wandb의 Sweep 기능을 통해 하이퍼파라미터를 최적화시킬 수 있었다.

2. 나는 어떤 방식으로 모델을 개선했는가?

A. 사용한 지식과 기술

Sequence한 데이터를 Transformer에 넣어주는 과정을 변형하여 연속형 feature 값을 Linear layer를 통해 임베딩 시켜주는 방법을 통해 Feature-engineering을 통해 얻은 연속형 feature들을 활용할 수 있도록 베이스라인 코드를 변경

Augmentation: sequence한 형태의 데이터를 늘려주기 위해 sliding window를 사용하여 사용할 수 있는 데이터를 증가시킴

K-fold : 모델이 특정 train셋에 편향된 학습결과를 보여주었기 때문에 모델을 일반화 시키기 위해서 k-fold 기법을 사용하였다. K-fold를 통해 모델의 성능을 정확하게 알 수 있었으며 제출 시에도 더 나은 성능을 얻을 수 있었다.

Pseudo labeling: 학습데이터를 더 많이 사용하기 위해 테스트 셋에서 라벨이 공개되지 않은 데이터의 라벨을 훈련셋으로 학습한 모델로 라벨링 해주고 다시 테스트셋과 훈련셋을 합친 전체데이터셋으로 재학습 시키는 방법을 적용시킴

Feature engineering: 사전에 EDA를 통해 얻은 인사이트와 가설을 바탕으로 여러 feature를 만들고 이를 여러 모델에 적용해 실험

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

기본적으로 제공된 베이스라인을 많이 분석하여 성능을 끌어올리기 위해 노력했다. 기존 코드에서 이산형 feature 만을 모델에서 사용할 수 있었지만, 연속형 데이터를 임베딩하여 사용할 수 있도록 개선하여 여러 feature를 실험해볼 수 있었다. 또한, 강의를 통해 학습한 여러 방법론을 이번 대회에 적용해보려 했고 그 결과 단독 모델로 AUC 0.8499에 이르는 성능을 얻을 수 있었다.

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

강의에서 학습한 기술들을 최대한 적용해보고자 하였다. 이전 대회에서는 대회에만 집중하여 강의를 통해 얻은 인사이트를 거의 활용해보지 못하였는데 이러한 부분을 보완하고자 이번 대회에서는 강의와 기본에 충실하려 노력하였다. 그 결과 기본적으로 제공된 베이스라인코드와 모델들을 완전히 이해할 수 있게 되었다.

저번 대회를 수행하며 대회에 적합한 모델, 빠르게 구현할 수 있는 라이브러리를 찾는 것의 중요성을 알게 되었다. 대회를 진행하면서 비슷한 유형의 대회에서 나온 시도한 모델과 기법들을 먼저 조사하여 선택과 집중을 할 수 있었다.

4. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

마지막 강의에서 소개한 UltraGCN 모델, Graph node embedding을 sequence형태로 transformer encoder로 넣어주는 기법을 구현하고 시도해볼 수 없었다. UltraGCN의 경우 논문을 이해하고 공개되어 있는 코드를 이번 대회에 맞춰 변경했어야 하는데 논문과 코드를 이해하지 못해 시도해 볼 수 없었다.

또한, Graph 계열의 모델을 잘 이해하지 못해 코드를 구현하고 실험해보지 못한 것이 아쉬웠다.

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

다음 프로젝트에서 Graph 모델을 공부하여 추천 시스템에서 활용할 수 있도록 시도해보면 좋을 것 같다.

1. 팀과 나의 학습 목표

이번에 진행되는 대회에서는 지난 대회인 Movie Recommendation에서 못했던 부분들을 많이 시도해보고자 목표를 설정하였다. 가장 먼저 협업 플랫폼을 기존 Notion에서 Confluence와 Jira로 변경하였다. Jira의 로드맵 기능을 활용하니 한 눈에 일정을 볼 수 있고 담당자를 지정하여 관리할 수 있는 것이 가능하여 더 효과적으로 프로젝트를 진행할 수 있게 되었다. 이번에는 큰 목표를 잡고 전체 리더보드에서 Top2에 드는 것이 목표로 설정하였고 프로젝트를 좀 더 체계화하여 관리하도록 하였다. 무엇보다 중요한 것은 이렇게 설정한 목표들을 끝까지 지키는 것이었다.

내 개인적인 목표는 쉽게 지치지 않고 열심히 대회를 완주하는 것이고, 코드를 작성할 때도 formatting이나 주석이 모두 작성된 완성된 코드를 작성하는 것이 목표였다. 그렇기에 초반에 코드를 가독성 있게 작성하려고 많은 노력을 했었다.

2. 모델을 개선시킨 방법

2-1. Transformer 계열

이번 대회에서 사용한 transformer 모델은 attention 구조를 사용한 LSTM, Last Query Transformer, Saint가 있었다. 기존 baseline에서는 feature engineering도 없었고, categorical feature만 사용되었기 때문에 continuous feature도 추가적으로 학습하게 만들었다. 중요한 것은 사용되는 feature와 hyperparameter로 생각이 되었고, 간단한 FE를 진행하고 wandb의 sweep 기능을 사용하여 튜닝을 진행했다.

2-2. Boosting 계열

Transformer계열의 모델들을 다루면서 다양한 Feature Engineering에 관심을 갖게 되었다. 실제로 사용할 수 있는 6개의 feature만 사용해서 최종적으로 40개가 넘는 최종 feature들을 만들어냈다. 처음에는 막연하게 그 수가 많을수록 더 많은 것을 참고하기 때문에 성능이 오를 것이라고 생각을 하였다. 하지만 결과가 보여주는 것은 적절한 feature들을 사용해야만 최적의 결과를 보여주는 것이었다. 이 feature를 선택하기 위해서 feature selection이라는 과정을 거쳤고, 더 상세하게는 wrapper를 사용하거나 직접 하나씩 조합을 하면서 실험 결과들을 기록하였다. 이렇게 선택한 feature들과 최적의 파라미터를 grid search를 통해서 찾아내고자 하였다.

2-3. Graph 계열

Graph 계열에서 사용한 모델은 Light GCN이다. LGCN은 유저와 문제풀이 정보 단 2개만을 사용해서 그래프를 생성하고 주어지는 입력에 대해 중간에 새로운 간선이 생길 확률을 예측하는 모델이다. GCN보다 구조가 더 단순하기에 학습속도도 빨랐고 그 성능도 좋았다. 하지만 한계점은 마음대로 그래프를 수정하기가 힘들고 다른 feature들을 전혀 사용하지 못한다는 것이 큰 한계점으로 드러났다.

2-4. Ensemble

최종 결과들을 가지고 2가지의 앙상블 기법을 적용하였다. 하나는 단순하게 모든 확률들의 평균을 내주는 soft-voting 방법이었고, 다른 하나는 threshold를 설정하여 이를 기준으로 더 많은 쪽의 확률들만 이용해서 soft-voting하는 방법이었다. 두 앙상블의 결과 차이는 크지 않았지만 그 편차가 큰 것에 대해서는 2번째 방법이 좀 더 성능을 올려주었다.

3. 내가 한 행동의 결과와 깨달음

특정 모델에 대해 성능이 잘 나오면, 그 모델만을 사용하려는 경향이 있었다. 단순히 그 모델의 튜닝을 하고 성능을 올리는 것보다 다른 모델들도 실험하는 것이 더 큰 성능을 가져다 줄 수 있을 것 같아서 다른 모델들도 가져와서 실험을 진행하였다. 단순히 그 결과가 좋지 않았다고 해서 버리는 것이 아니라, 후에 앙상블에 사용되어서 더 좋은 결과를 가져다 주는 것도 경험하였다. 특히 이번 대회에서는 구현했던 모든 모델들을 앙상블 한 결과가 private dataset에서 2번째로 좋은 성능을 보여주었다.

Python의 black을 사용하여 code style을 통일시키고 누구나 보기 쉽고 잘 읽히는 코드를 작성하려고 노력하였다. 팀원들과의 협업이 요구되는 대회이기에 더 신경을 쓰려고 했고, 결과적으로 좋은 피드백을 받았다.

4. 한계와 아쉬운 부분

Boosting 모델을 사용해서 학습을 진행할 때, 특정 feature에 대해서 overfitting이 발생하는 현상이 있었다. Train과 valid set 그리고 test set도 모두 겹치지 않게 나누어서 학습을 진행했는데, train과 valid set 모두 좋은 성능을 보여주지만 test set에서만 크게 안 좋은 AUC 결과를 보여주는 일이 있었다. 이 문제를 특정 feature가 너무 많이 관여하거나 정답을 크게 내포한 feature라서 안 좋게 나왔다 라고만 추측을 하였지 정확한 원인을 찾지는 못한 것이 아쉽다.

Graph 모델에서는 LGCN이외의 다른 시도들 Ultra-GCN을 사용해보지 못했던 것이 아쉬웠다. 더 간단하지만 좋은 성능을 보여줄 것이라고 생각이 된다.

5. 개선사항, 새롭게 시도해볼 수 있는 것들

한가지 아쉬웠던 것은 시간이 빠듯해서 코드 리뷰하는 시간을 많이 갖지 못했던 것이다. 다른 사람이 작성한 코드를 통해서 배울 수도 있고, 내가 작성한 코드의 문제점을 파악할 수 있는 좋은 시간이 될 수 있었을 것이라고 생각된다. Git의 branch 전략을 사용해서 main, develop, feature, eda와 같은 크게 분류하여서 진행되었는데 중간에 pull request과정을 많이 하지 않아서 브랜치 간의 차이가 컸던 적이 많았다. 그러다 보니 불편한 점이 많았었는데 필요한 부분에 대해서는 그 때마다 요청을 해서 검토한 후에 merge하는 것이 효율적이라고 생각된다.

1. **나는 내(팀) 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**
   나와 우리 팀의 학습 목표는 모두가 추후 프로젝트를 위해 최대한 다양한 경험을 하는 것이었다. 이를 위해 개인 측면에서는 기본 강의 내용에 충실하되 필요 시 새로운 내용을 적극적으로 찾아보고자 했고, 팀 측면에서는 자율적인 의견 제시를 장려했으며 원활한 소통을 위해 다양한 협업 툴을 활용했다.

2. **나는 어떤 방식으로 모델을 개선했는가?**
   A. Cross Validation
      훈련 데이터 셋을 8 대 2 로 나누어 검증하는 기본 validation 기법은 실제 성능과 차이가 다소 크다고 느꼈기 때문에 더 효과적인 validation 기법을 찾고자 노력했다. K-Fold cross validation, stratified cross validation, 등을 실험해 봤으며, stratified k-fold cross validation 이 가장 실제 성능과 유사한 validation 성능을 보여준다는 것을 발견했다. 하지만, K-Fold 기법은 시간이 오래 걸린다는 단점이 있었기 때문에, 훈련 셋과 테스트 셋을 통합해서 stratified validation 을 하는 기법을 사용했다.
   B. Feature Engineering
      EDA 를 통해 얻은 데이터 특성을 고려하여 학생, 문제 대분류, 문제 태그, 등과 관련된 여러가지 feature 를 만들었다. 그 중 학생별 대분류 정답률, 문제별 평균 정답률이 높은 모델 성능에 기여했다.
   C. Feature Selection
      만들어낸 feature 들을 어떻게 조합하느냐에 따라 성능이 크게 변하는 것을 보고 최적의 조합을 찾는 feature selection method 를 연구했다. Filter method 와 forward selection 을 조합한 방법과 Filtering method 와 wrapper method 를 조합한 방법을 사용하여 성능을 최대치로 끌어올릴 수 있는 feature 들의 조합을 탐색했다.

3. **내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?**
   Validation 기법과 feature manipulation 의 중요성을 깨달았다. 제출 횟수가 하루에 10 번으로 제한되었기 때문에, validation 성능이 실제 성능과 많이 다르다면 올바른 가설을 세우기 어려우며 제출 횟수를 낭비할 우려가 있었다. 또한 현업에서는 잘못된 validation 기법을 사용하게 되면 기회 비용이 더 더욱 높을 것이라고 생각했다. 따라서 효과적인 검증 방법을 찾고자 노력했고,

   실험한 기법 중 Stratified k-fold cross validation 이 가장 효과적이었다. 다만, 해당 방법은 시간이 너무 오래 걸린다는 단점이 있었기 때문에, 대회에서 주어진 훈련 셋과 테스트 셋을 통합하여 무작위로 검증 셋을 추출하는 stratified validation 방식을 채택했다. 해당 방식은 feature 와 모델에 대한 실험을 할 때 도움이 되었다.

   또한, feature manipulation 의 중요성을 깨달았다. 특히 feature selection 이 feature engineering 만큼이나 중요하다는 것을 깨달았다. 우리가 갖고 있는 feature 40 개를 넣었을 때보다, 적절한 feature 를 14 개 선택해서 넣었을 때 훨씬 더 높은 성능을 냈다. Filter method 와 forward selection method 를 혼합한 feature selection 기법으로 선택한 14 개의 feature 를 XGBoost 모델에 넣었을 때 단일 모델로서는 최고 성능인 0.8572 AUC 를 얻을 수 있었다.

4. **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**
   이번 대회는 과거와 달리 모델링보다는 데이터와 검증 기법에 초점을 맞춰서 진행했다. 지금까지 성능을 올리기 위해 적절한 모델을 찾는 것이 가장 중요하다고 생각했었다. 실제로 과거에 성능을 수직 상승시켰던 것은 대부분 모델링의 결과였다. 하지만 이번 대회를 통해 모델링 못지않게 데이터 가공과 실험 기법 연구가 중요하다는 것을 깨달았다. 여러가지 요소를 함께 종합적으로 연구하고 개발했을 때, 최대치의 성능을 얻을 수 있는 것 같다.

5. **마주한 한계는 무엇이며, 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?**
   선행 연구 탐색이 부족했던 점과 모델 과적합을 해결하지 못한 것이 이번 대회의 한계였다. 우리 팀이 처음부터 스스로 아이디어를 브레인 스토밍하는 것도 성장에 도움이 됐지만, 선행 연구를 더 찾아보고 이해한 뒤 그것에 추가적으로 아이디어를 고민했다면 더 더욱 성장했을 것 같다. 또한, LGBM, XGBoost와 같이 머신러닝 모델을 사용했을 때, 훈련 데이터와 검증 데이터에 대해 과적합 되는 feature들을 발견했는데, 그 이유를 명확히 밝혀내지 못한 점이 큰 아쉬움으로 남아있다. 다음 프로젝트에서는 이러한 아쉬움들이 없도록 더 노력해야겠는 생각이 들었다.

1. 학습목표

개인 학습 목표는 모델 구현 능력을 키우고, 학습목표에 맞춰 모델을 고도화해보며 Feature Engineering과 Feature Selection 최대한 많이 경험해보는 것으로 정했다. 팀의 공동 학습 목표는 프로젝트를 체계화하여 관리해보고 Wide & Deep하게 모델을 실험해보는 것으로 결정했다. 개인 학습 측면에서는 각 Feature의 성능을 빠르게 확인할 수 있는 Boosting 계열 모델을 공부하고 사용해봄으로써, 다양한 Boosting 계열 모델의 장단점과 알고리즘을 알 수 있었다. 또한, Feature Selection의 지표 중 하나인 Sklearn의 Permutation Importance을 구현하여 중요한 Feature를 선택하며 최적의 조합을 찾을 수 있었다. 공동 학습 측면에서는 Jira와 Confluence 협업 툴을 새로 도입하여 적극 활용하였다. Jira에서 각 팀원별 에픽과 하위 이슈를 등록하여 일정 관리를 했고, 여러가지 가설과 검증 결과를 Confluence wiki를 통해 체계적으로 관리했다. 또한, 프로젝트 그라운드 룰과 Git 버전 관리 규칙을 초반에 정함으로써 각 팀원이 만든 Feature들을 수월히 통합할 수 있었다.

2. 모델 개선 방법

A. Feature Engineering
단순히 사용자의 정답률을 사용하기보다 사용자가 푼 문제의 시험지 난이도와 태그 난이도를 반영하여 사용자 실력 수준을 매기고자 했다. 따라서 기본 feature인 시험지와 태그별 정답률을 구하여 시험지와 태그 난이도를 먼저 정의했다. 새롭게 구한 시험지와 태그 난이도를 반영한 유저의 실력 수준을 구할 수 있었고, 최종적으로 모델 성능에 긍정적인 영향을 미치는 것을 확인할 수 있었다.

B. Feature Selection
EDA를 통해 만든 약 40개의 feature 중에서 유의미한 feature를 뽑아내기 위해서 LGBM 모델을 통해 각 feature의 성능을 확인하고자 했다. 하지만 feature의 조합에 따라 다른 성능을 보이는 것을 알게되었고, 단순히 Tree 모델에 해당 feature가 얼마나 기여하는 지가 중요한 것이 아니라 얼마나 긍정적인 영향을 미치는 지가 중요하다는 것을 깨닫게 되었다. 따라서 Sklearn의 Permutation Importance 패키지를 활용하여 긍정적인 feature를 뽑아낼 수 있도록 코드 구현을 했고, 중요도와 표준편차를 출력하며 어느 정도로 의미가 있는지 파악할 수 있었다.

C. Modeling
LGBM을 통해 다양한 feature를 학습하고자 했지만, Residual fitting 방식으로 학습하기 때문에 bias는 줄일 수 있어도 과적합이 쉽게 일어날 수 있다는 단점이 있었다. 이를 해결하고자 Boosting 계열인 CatBoost에 대해 탐색했다. CatBoost는 과적합 문제와 하이퍼파라미터에 따라 성능이 달라지는 민감한 문제를 해결하는 것에 초점이 맞추어져 있어 사용하기 적합하다고 판단했다. CatBoost 모델은 범주형 변수와 연속형 변수에 대한 데이터 정의를 따로 지정해줘야 했기 때문에 범주형 변수에 대해서만 따로 전처리를 진행했다. CatBoost 모델과 앞서 설명한 Permutation Importance를 통해 AUC 0.8504인 우수한 성능을 얻을 수 있었다.

D. Visualization
LGBM에서 k-fold Cross Validation을 사용할 때 각 fold별로 feature의 중요도가 다르므로, Wandb에 Permutation Importance와 Feature Importance가 막대 그래프로 기록될 수 있게 코드를 구현했다. 여러 번의 fold를 진행하면서 상위권에 있는 Top5 feature는 유의미한 feature라 판단하는 방식으로, 시각화 된 그래프를 보며 feature selection을 좀 더 수월히 할 수 있었다.

3. 수행 성과 및 깨달음

Feature Selection을 할 때 조합에 따라 각 feature의 중요도가 달라지고, 이것이 성능에 많은 영향을 미치는 것을 깨닫게 되었다. 어떤 조합에서는 긍적적인 영향을 미치던 feature가 다른 조합에서는 부정적인 영향을 미치는 것을 보며 조합 선택을 잘 할 수 있는 방법에 대해 고민했다. Permutation Importance 활용하여 유의미한 핵심 feature를 선택할 수 있었고, 이 고정 feature에 다른 feature를 하나씩 추가하면서 성능실험을 했다.

다양한 Boosting 계열 모델을 사용해보면서 LGBM과 GBM, XGBoost 등 하이퍼파라미터에 민감한 모델은 적절한 튜닝이 이루어져야 한다는 것을 깨닫게 되었다. 따라서, Wandb의 sweep을 통해 최적의 파라미터를 찾으며 sweep 사용에도 익숙해질 수 있었다.

4. 새롭게 시도한 변화

이번 대회에서는 문제정의와 목표를 명확히하고, 폴더 경로와 파일명 규칙을 미리 정하고 시작하여 불필요한 충돌을 줄일 수 있었다. 또한, 이전 대회에 대한 회고를 통해 이번 대회에서 꼭 필요한 태스크에 대해 리스트업을 하여 같은 아쉬움이 남지 않도록 했다. 새로운 협업 툴인 Jira와 Confluence를 활용하여 일정관리와 역할 분담을 손쉽게 할 수 있었다.

5. 한계 및 아쉬웠던 점

이번 대회에서 Graph 기반 모델을 사용해보지 못한 것이 아쉬웠다. 또한, LGBM에서 과적합이 자주 발생했는데, 과적합 원인을 분명하게 찾아내지 못한 것이 한계였다. 특히 정답률을 직접적으로 반영한 feature들을 학습했을 때 과적합이 발생한 것이라 예상했지만 이 가설에 대해서 제대로 검증을 해보지 못해서 아쉬움이 남는다. 또한, time과 관련된 feature는 모두 범주형 변수로 처리해서 모델을 학습시켰는데, sequence 모델을 통해 sequential한 특징까지 살리지 못한 것도 아쉬웠다.

6. 개선사항, 새롭게 시도해볼 수 있는 것

다음 프로젝트에서는 그래프 기반 계열의 모델을 활용해보고 싶다. 마지막으로, Recbole과 같은 외부 라이브러리를 최대한 활용하여 제한없이 다양한 실험을 해보고 싶다.

1. **나는 내(팀) 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

팀 공통적으로 프로젝트 관리를 체계화하기로 하여 Jira, confluence를 도입해 프로젝트만을 관리할 수 있도록 했다. 또한, 지금까지 여러 모델을 사용해서 성능을 높이는 방법으로 프로젝트를 진행해왔기에 모델을 계열별로 탐색하되, 데이터 특성에 맞는 모델 계열을 정해서 깊게 파보기로 했다. 그래서 baseline과 special mission을 활용하여 모델 탐색과 적용을 수월하게 했다. 이번 대회의 데이터양이 DL을 사용해서 성능 향상을 많이 보기에는 충분하지 않다고 판단하여 ML 모델 중 안정적이게 높은 성능을 내는 boosting 계열에 집중했다. 그렇기에 기존 data에서 feature engineering, selection을 통해 성능을 올려 리더보드 Top 2진입의 목표를 달성하고자 했다. 개인적 목표로는 hyper parameter tuning보다는 feature engineering과 selection을 통해서 설명할 수 있으면서 근거있는 성능향상을 이루고자 했다.

2. **나는 어떤 방식으로 성능을 향상시켰는가?**

Hyper parameter tuning은 많은 노력을 들이지는 않고도 적당히 좋은 값들을 알아내고 싶어서 wandb의 sweep을 사용했다. 그러기 위해서 wandb를 기본적으로 다루는 공부를 하였고, 적당히 오래 돌려서 얻은 값으로 사용했다.

Boosting 계열 뿐만 아니라 NMF, TabNet도 개인적으로 적용시켰고, 있던 모델들과 팀원들이 적용한 sequence, transformer 모델들도 시도했지만 눈에 띄는 성과가 없어서 탐색하고 적용한 것으로 마무리 지었다. Feature 탐색과 적용에 Boosting 계열이 빠르고 성능도 준수해서 계속해서 LGBM을 사용했다.

가장 집중했던 것은 feature engineering 이었다. Feature을 만들거나 없애고 합치기 위해서는 그렇게 행동하는 근거가 필요했다. 그래서 EDA에 힘썼다. 첫 번째로 얻은 것은 outlier였다. 문제 풀이 수가 극히 적으면 sequence가 학습될 만큼 쌓이지 않았을 것이라고 가정했고, 유저의 정답률이 너무 낮거나 높은 유저나 분포에서 빠져나온 학생을 outlier로 가정하고 학습 시 제거했다. 그 결과 조금의 성능 향상이 있었다. Feature를 최대한 많이 만든 다음 적절하게 잘 나타내는 feature selection을 통해서 성능을 올리는 전략을 세웠기에 EDA를 통해 유의미한 값들이 나올 것 같으면 feature로 만들어서 test 했다. 많은 feature들을 만들고 feature들 간의 correlation, permutation importance를 고려한 후에 해당 feature가 생성되게 된 원래의 feature들이 겹치지 않도록 selection을 시도했다. 하지만 조합마다의 importance가 달랐고, feature를 고르는 과정이 내가 생각한 하나의 가설일 뿐이라서 좋은 결과를 얻지는 못했다. 그래서 조원과 함께 wrapper를 통해 하나씩 feature를 쌓아가며 성능을 개선했다.

마지막으로 성능을 끌어올리기 위해서 앙상블을 시도했다. 많은 조합이 있었지만 좋은 모델들만 사용하는 것이 아닌, 다양한 계열에서의 모델을 모두 사용하는 것이 average ensemble에는 유리할 것이라는 판단이 서서 6개의 모델을 사용해서 앙상블 했다.

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

많은 Feature를 만들자는 목표 아래에서 EDA가 필수적이었기 때문에 EDA를 많이 했던 것이 데이터를 이해하는 데 도움이 많이 되었다. 하지만 EDA는 어느 정도 눈에 띄는 상관관계들을 보여주는 것이 유의미하고 그 내용을 바탕으로 feature로 만드는 것은 latent 한 정보가 아니라 feature로 우리가 생각한 답을 주게끔 만드는 행동이라는 것을 overfitting을 일으키는 몇몇 feature들을 분석하면서 깨달았다. Validation, feature engineering 방법 등을 바꾸면서 overfitting을 해결하려고 노력했었지만 끝까지 overfitting이 나는 feature도 남았다. 결국에는 DKT 자체가 문제를 맞힐지 아닐지에 대한 목적보다 넓게 보았을 때 문제의 수준과 유저의 수준을 알 수 있게끔 하는 feature(elo, elapsedTime, bigClassAccuracy, recCount)들이 중요하다는 것을 깨달았다.

4. **마주한 한계/교훈는 무엇이며, 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?**

feature를 많이 만들자는 목표 아래에서 어떻게 그 feature가 줄 의미를 깊이 고려하지 않고 무작정 만들어서 성능이 잘 나오는 feature를 골라내었더니 성능은 분명 나아졌지만 내가 만든 feature를 설명하기는 어려워졌다. 의미를 줄 것이라는 가정과 믿음으로만 실행하고 '왜 의미가 있을까?'를 생각하지 못하여 아쉽다.

sequence형이 아닌 tabular data 형태로 바꾸어서 문제를 풀고자 했었다. 그래서 유저 한 명의 정보를 한 행에 요약하는 data preprocess를 거친 후에 모델에 적용하려고 했다. 지금까지 모델의 task는 해당 유저가 그 sequence의 마지막에 문제를 맞출지 아닐지 binary classification이었는데, preprocess를 거치며 sequence가 한 행으로 요약되며 regression 문제로 바뀌게 되었다. 문제를 단지 해결만 했을 뿐이지 처음에 충분히 고려하지 않고 진행부터 시켰던 부분이 아쉽다.

먼저 코드를 만드는 행동으로 생각을 옮기기 전에 이 행동이 어떤 의미인지, 해도 될지 유효성과 의미를 반드시 정리하고 실행하면 더욱 좋을 것 같다.