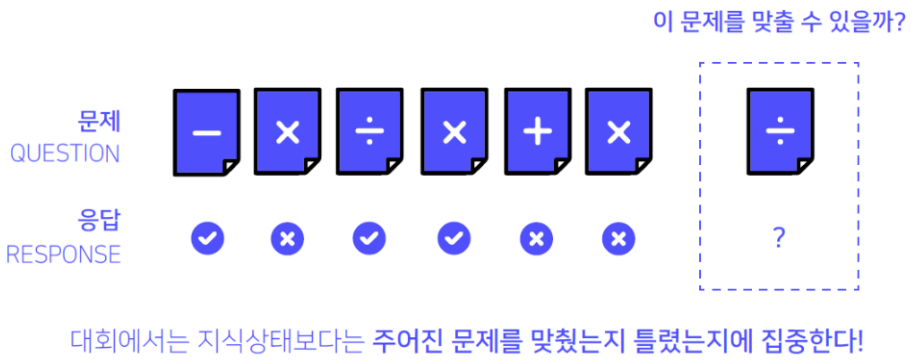
DKT WrapUp Report

Recsys - 2조(Recommy) 팀원이름 : 김동환, 김영서, 박재성, 전예원, 진성호

1. 프로젝트 개요

1.1 개요

- 프로젝트 주제



DKT는 Deep Knowledge Tracing의 약자로 “지식 상태”를 추적하는 딥러닝 방법론임. 이번 DKT대회에서는 학생들의 과거 문제 풀이 리스트와 정답여부가 담긴 Iscream 데이터셋을 이용하여 최종 문제를 맞출지 틀릴지 예측하는 것이 목표

- 진행기간 : 5월 2일 (화) 10:00 ~ 5월 25일 (목) 19:00

1.2 환경

- (팀 구성 및 컴퓨터 환경) 5인 1팀, 인당 V100 서버를 할당받아 주피터랩과 VSCODE에서 사용
- (협업 환경) Notion, Github, WandB
- (의사소통) 카카오톡, ZOOM, Slack, gather town

1.3 프로젝트 구조 및 사용 데이터셋의 구조도(연관도)

```
├── EDA
│   └── EDA_JAESEONG.ipynb
├── FeatureEngineering
│   ├── ELO.ipynb
│   ├── feat_class.ipynb
│   └── feat_time.ipynb
├── Read.md
├── code
│   ├── LGBM
│   ├── README.md
│   ├── TabNet
│   ├── XGBoost
│   ├── __init__.py
│   ├── dkt
│   ├── lastquery
│   └── lightgcn
├── data
│   ├── train_data.csv
│   └── test_data.csv
└── ensemble
    └── ensemble.py
```

2.프로젝트 팀 구성 및 역할

전체	문제정의, 계획 수립, 목표 설정, EDA, 모델 실험, 튜닝
김동환	모델링 및 데이터파이프라인 구성
김영서	Bert, lstm 구현 및 실험
박재성	LGBM, XGBoost 구현, Feature Engineering, Ensemble
진성호	LastQeury 구현, Hyperparameter Tuning 기초 구현, Git 관리
전예원	TabNet, CatBoost 구현 및 실험

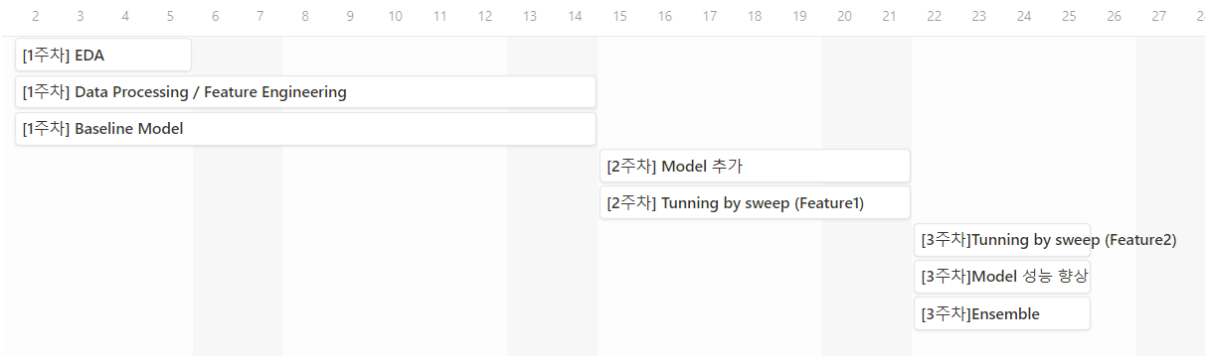
3 프로젝트 수행 절차 및 방법

3.1 팀 목표 설정

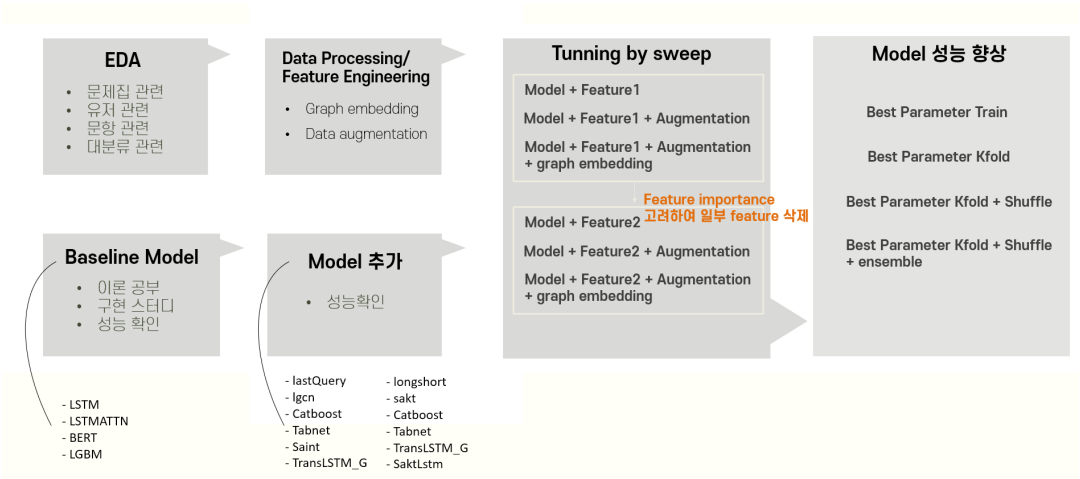
1. wandb, Github와 같은 실험 및 협업 툴 사용해보기
2. Baseline 이외의 모델 각자 선정해서 깊게 분석해보기
3. 원활한 소통을 통해 서로 많이 배우고 성장하기

3.2 프로젝트 수행 절차

- 타임라인

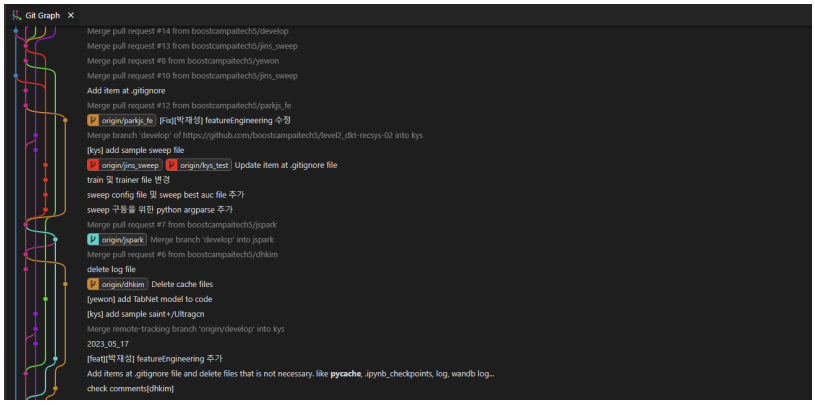


프로젝트 프로세스

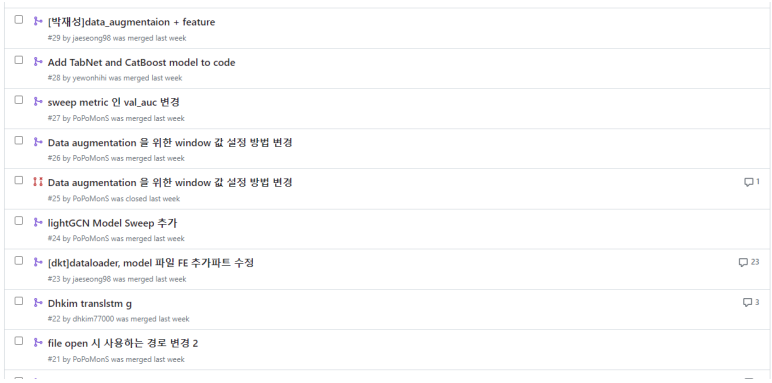


3.3 협업 문화

- **Github**
  - **Branch 기능**
    - main : 배포될 branch. 문제가 없어야 하며, 충분한 검증을 통해 안정적이어야만 하는 branch
    - develop : 새로 개발된 기능들을 기존 코드와 붙여 검증을 진행하는 branch. 충분한 검증이 완료되어 안정적이다 생각되면 main branch 로 merge.
    - 개인 branch : 팀원 각자 새로 개발하는 기능들을 넣어 테스트하고, develop 으로 pull request 요청을 할 branch

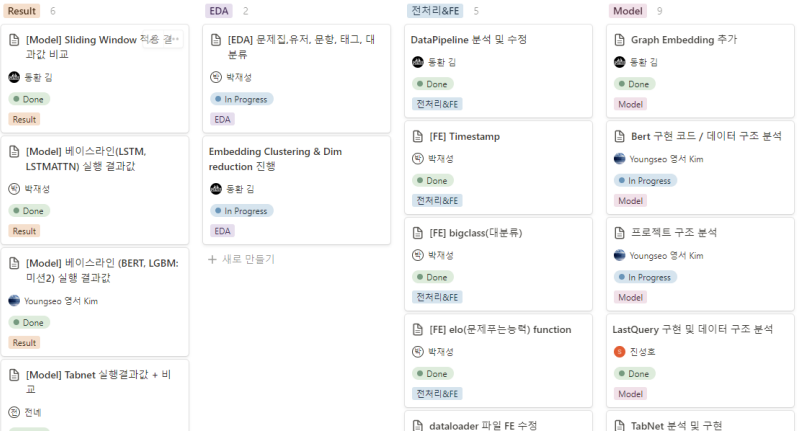


- **Pull Request(PR)**
  - 개인 branch에서 작업한 내용을 develop에 push하기 전 PR을 열어 검토

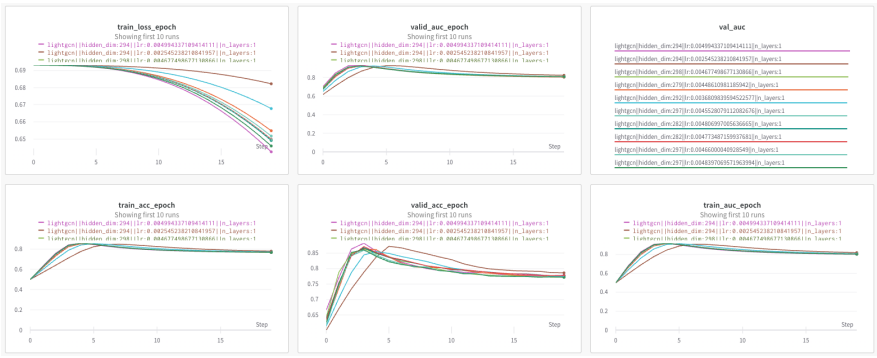


- **Slack으로 이슈 공유**
  - Github와 Slack 알림을 연동하여 PR, Push 등 변경사항을 확인

- **Notion**
  - 프로젝트 진행 상황 실시간 공유



- **WandB**
  - 실험 결과 시각적으로 공유
  - Sweep으로 hyper parameter 자동 튜닝



실험했던 여러 모델 중 하나

- **Offline Meeting**
  - 대회 후반부, 빠른 의견교환 및 효율성 향상을 위해 Offline Meeting 을 진행

4 프로젝트 수행 결과

4.1 탐색적 분석 및 전처리

- **데이터 소개**

train/test 합쳐서 총 7,442명의 사용자가 존재함. 한 행은 한 사용자가 한 문항을 풀었을 때의 정보와 그 문항을 맞췄는지에 대한 정보가 담겨있음. 데이터는 모두 Timestamp 기준으로 정렬되어 있으며 이 때 사용자가 푼 마지막 문항의 정답을 맞출 것인지 예측하는 것이 최종목표

`userID` 사용자의 고유번호

`assesmentItemID` 문항의 고유번호

`testId` 시험지의 고유번호

`answerCode` 사용자의 해당 문항 정답 여부에 대한 이진데이터

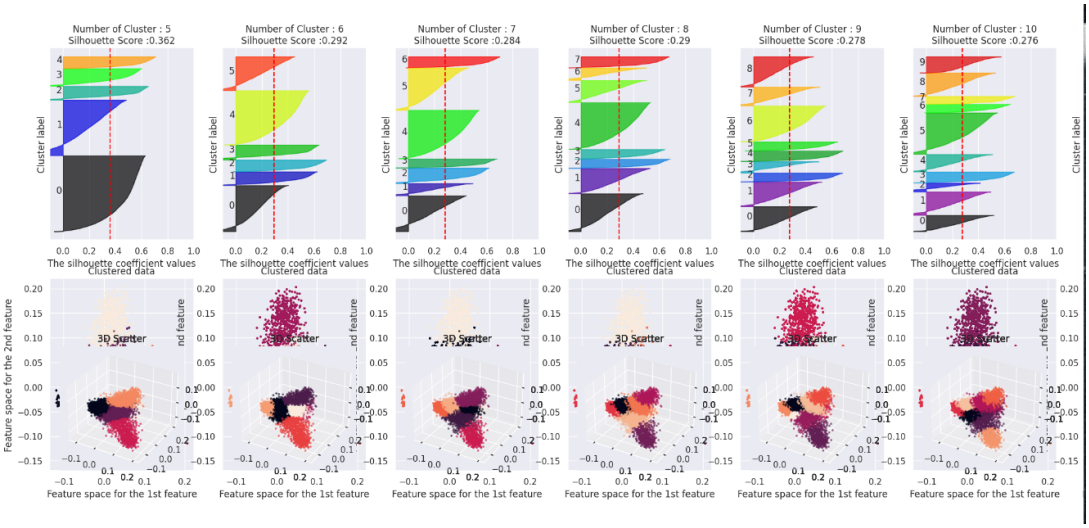
`Timestamp` 사용자가 해당문항을 풀기 시작한 시점의 데이터

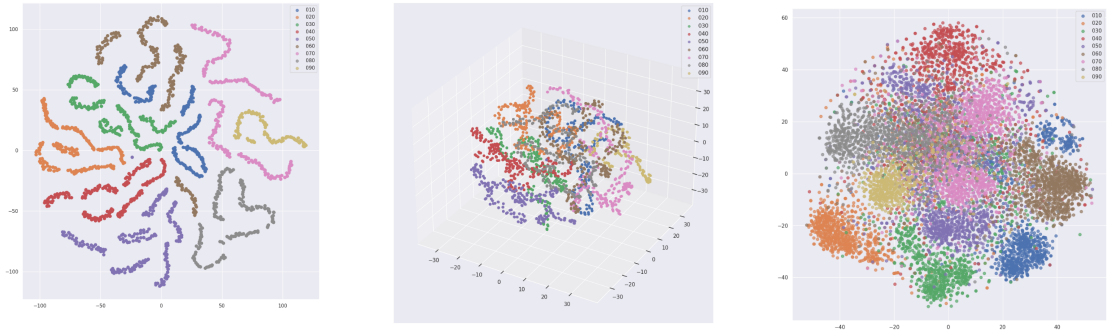
`Knowledge` 문항 당 하나씩 배정되는 태그

	userID	assessmentItemID	testId	answerCode	Timestamp	KnowledgeTag
0	0	A060001001	A060000001	1	2020-03-24 00:17:11	7224
1	0	A060001002	A060000001	1	2020-03-24 00:17:14	7225
2	0	A060001003	A060000001	1	2020-03-24 00:17:22	7225
3	0	A060001004	A060000001	1	2020-03-24 00:17:29	7225
4	0	A060001005	A060000001	1	2020-03-24 00:17:36	7225
...	...	...	...	...	...	...
2266581	7441	A030071005	A030000071	0	2020-06-05 06:50:21	438
2266582	7441	A040165001	A040000165	1	2020-08-21 01:06:39	8836
2266583	7441	A040165002	A040000165	1	2020-08-21 01:06:50	8836
2266584	7441	A040165003	A040000165	1	2020-08-21 01:07:36	8836
2266585	7441	A040165004	A040000165	1	2020-08-21 01:08:49	8836

2266586 rows × 6 columns

- **EDA**
  - LGCN으로 학습된 Embedding vector를 차원 축소와 클러스터링을 통해 분석
  - 문항이 꽤 유의미하게 나뉘고 앞에 3번호에 따라서 나뉘는 것을 확인
  - Word2vec으로 학습된 Embedding Vector를 차원 축소와 클러스터링을 통해 분석
  - LGCN와 다른 양상으로 나뉘었고 마찬가지로 앞에 3번호에 따라서 나뉘는 것을 확인 및 순서 정보가 반영된 embedding 이 훨씬 효과적인 것도 확인





4.2 Feature Engineering

- **Data Augmentation & Shuffling**
  - Stride를 두어서 max sequence length를 벗어난 데이터도 학습 데이터로 활용할 수 있도록 구현

- Shuffling으로 데이터의 마지막 순서만 제외하고 뒤섞어 데이터셋의 양을 늘리고 sequence에 대한 dependency도 학습이 될 수 있도록 구현

- **파생변수**

기본 주어진 피쳐를 메모리 기법을 이용하여 6개의 피쳐에서 21개의 피쳐로 증가시킴. 메모리 기법이란 주어진 데이터 이전/이후 데이터들을 포함하는 메모리를 feature로 포함시킴으로서 순서 모델(Sequence Model)를 사용하지 않고 일반적인 지도 학습 모델(ex: Light GBM)을 사용하여 훈련할 수 있음

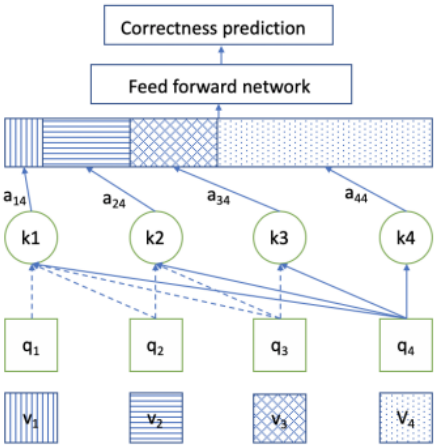
Base	userID, assesmentItemID, testId, Timestamp, Knowledge, answerCode
Category	question_N, bigclass, dayname
Numeric	user_correct_acc, user_total_answer, user_acc, test_mean, test_sum, tag_mean, tag_sum,
Bigclass	bigclass, bigclasstime, bigclass_acc, bigclass_sum, bigclass_count
Time	elapsed, elapsed_cumsum,elapsed_med, month, day, hour, dayname
etc	elo

4.3 모델링

- 모델 개요

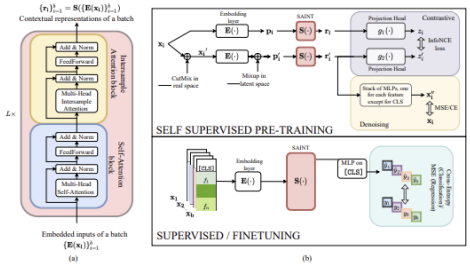
Sequence Model

- TransLSTM
  - Transformer로 Feature들의 특성을 파악 후 이를 다시 한 번 LSTM에 넣어주어서 output으로 사용
- saktLSTM
  - 학생들의 과거 상호작용에서 관계있는 것들을 Transformer의 Attention을 이용해 찾고 예측하는 모델
  - 기존 SAKT는 Tabular data에 초점이 맞춰진 모델로 Sequence를 반영하기 위해서 Feed Forward Network가 아닌 Attention을 거친 뒤에 LSTM 단으로 예측한 값을 output으로 사용
  - 문제를 맞췄는지 틀렸는지에 따라 문제 별로 다르게 임베딩이 되고 위치 정보 또한 지도록 고려되도록 구현

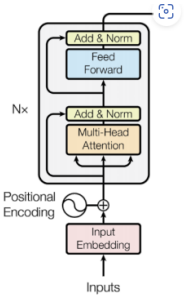


- LongShort
  - long term 효과와 short term 효과를 각각 다르게 학습 후 두 효과를 동시에 고려할 수 있도록 long을 예측하는 부분과 short을 예측하는 부분을 분리해서 구현한 모델
  - 같은 길이의 sequence를 받아 long은 전체를 학습하고 short는 뒷부분은 padding 후에 학습해 이 둘을 합쳐 최종 output으로 예측

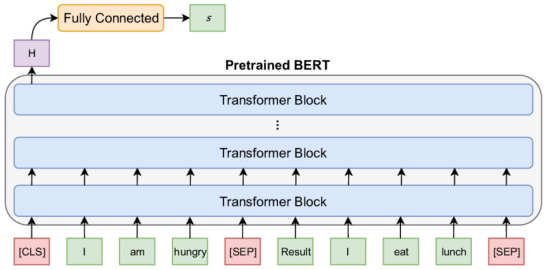
- SAINT
  - 정형 데이터에 Transformer 기법을 적용한 모델



- Bert
  - Transformer의 Encoder 구조를 중첩하여 data를 학습하고 분류를 위해 마지막 단에 Fully Connected Layer 를통과하여 output 생성



트랜스포머의 인코더 블록 부분



- LastQuery
  - Transformer의 Encoder + LSTM
  - $O(L^2)$의 시간 복잡도를 갖고있는 Transformer 모델을 마지막 Query vector 만 사용하도록 변형하여 $O(L)$의 시간 복잡도를 가짐
  - 이렇게 학습을 진행해도 괜찮은 이유는 마지막 L 번째 시퀀스 데이터 이후 오는 값만 예측하면 되기 때문
  - 모델의 특징으로인해, 시퀀스 데이터의 길이를 길게 늘릴 수 있음

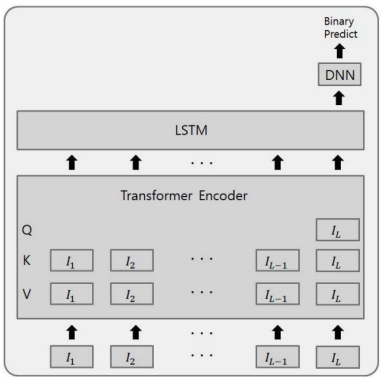


Figure 1. Model Structure

- LSTMATTN, LSTM, Bert
  - 기본 모델

Boosting Model

- Catboost
  - Target Encoding 방식으로 범주형 변수를 인코딩하고 Ordered Boosting 방식을 적용한 트리 구조 모델
- Tabnet
  - Tabular 데이터에 전처리 없이 Transformer와 Masking 기법을 적용한 AutoEncoder 모델

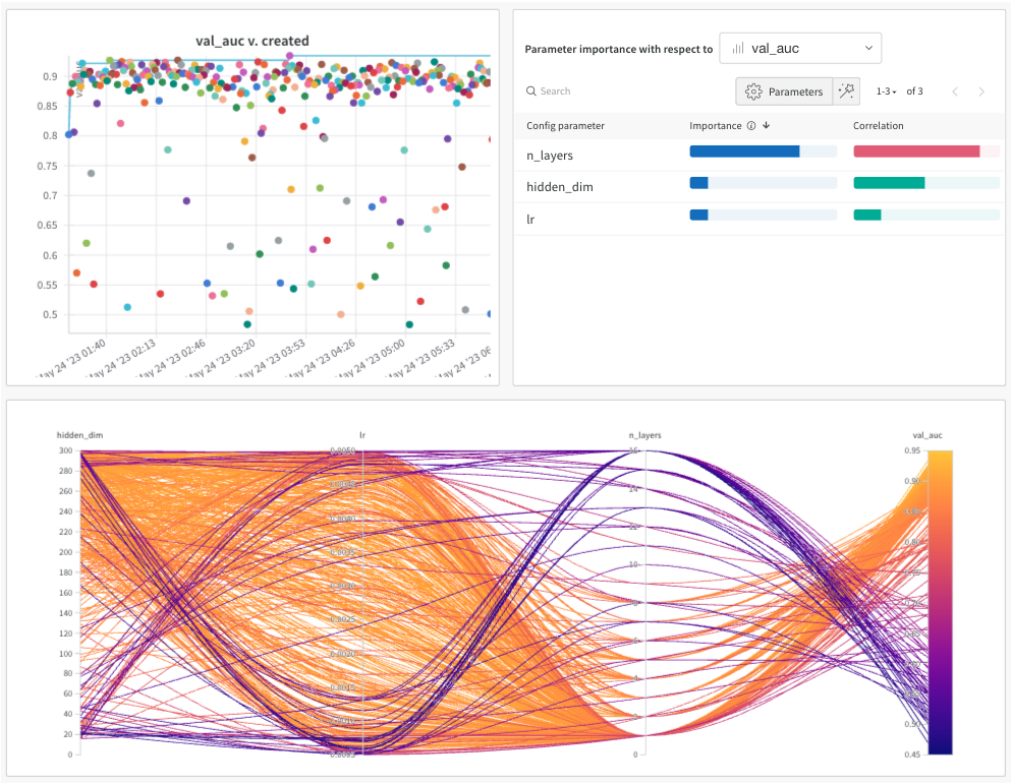
    - LGBM
        - 트리기반 학습 알고리즘 모델인 Gradient Boosting 방식을 leaf-wise 구조로 확장한 모델
    - XGBoost
        - 병렬처리 및 CART 앙상블 모델을 이용하여 기존 GBM을 개선한 모델
- 모델 선정 및 분석
    - sweep을 통해 가장 적합한 파라미터(n layer, hidden dim) 등을 찾고 가장 적합한 paramter를 가질 때의 성능으로 판단
    - 21개의 Feature를 사용할 경우 model 학습 속도가 너무 느려져 feature importance가 높은 feature 선택후 재학습
    - sequence model의 경우 batch size가 클수록, 학습률이 낮을 수록 학습속도가 느려서 train 시키기 적당한 config값으로 재설정
    - sequence model의 경우 graph embedding, shuffle 유무를 sweep config에 추가하여 실험
    - LGCN에서 학습된 embedding을 사용해 성능을 향상
        - Sequential한 정보로 얻을 수 있는 점과 Graph로 얻을 수 있는 정보가 달라 이를 같이 사용 시에 성능이 향상된 것으로 보임
    - 결과적으로 가장 성능이 좋았던 SAKTLSTM 과 TRANSLSTM, Bert를 채택
        - Transformer의 Attention 기법이 Input의 특성을 다른 모델보다 효과적으로 포착하는 것 같았음
    - Boosting 계열 모델에서는 리더보드 결과 성능이 가장 좋았던 LGBM을 채택
- 시연 결과
    - 각 모델 별 Valid 결과

Model	AUROC	ACC
`LSTM`	0.8337	0.7324
`LSTMAttn`	0.8287	0.7453
`Bert`	0.849	0.7945
`LGBM`	0.8496	0.7727
`XGBoost`	0.8473	0.7727
`SAKTLSTM`	0.8368	0.7859
`TransLSTM`	0.8405	0.7852
`LightGCN`	0.9343	0.8820
`LastQuery`	0.7740	0.4960
`TabNet`	0.8093	0.7623
`CatBoost`	0.8412	

    - LightGCN 의 경우 과적합이 매우 심하여 제외

4.4 하이퍼파라미터 튜닝

- WandB 의 내장 Library 인 Sweep 사용
- 각 모델별 돌아가는 main function 을 변형하여 objective function 구현
- 전체 Epoch 동안 측정된 AUROC 값 중 best 값이 최대값이 되도록 metric 설정
- 튜닝 방법은 Bayesian Optimization 으로 진행
- 여러번의 튜닝중 제일 높은 값의 AUROC 를 갖는 Hyperparameter 값 별도로 저장



실험 했던 여러 모델 중 하나

```
lstm:
  method: bayes # 매개변수를 찾는 방식
  metric: # 어떤 메트릭을 기준으로 좋은 실험을 찾을지
    name: val_auc
    goal: maximize
  parameters: # 변경하고 싶은 parameters
    # 범주형의 경우 실험하고 싶은 값들을 리스트로 넣어줍니다.
    # 연속형이더라도 실험하고 싶은 값들이 정해져있다면 리스트로 넣어줍니다.
    n_epochs:
      value: 30
    drop_out:
    # 연속형의 경우 최대/최소값을 기입해주면 그 범위 안에서 값을 적절히 선택하여 실험을 진행합니다.
      min: 0.0
      max: 0.5
    hidden_dim:
      values: [16, 32, 64, 128, 256, 512]
    batch_size:
      values: [64, 128, 256, 512]
    lr:
      min: 1e-4
      max: 1e-2
    max_seq_len:
      min: 5
      max: 50
    pos_int1:
      values:
        - True
        - False
```

4.5 앙상블

리더보드 제출 결과 성능이 가장 좋았던 모델들을 Weight 방식으로 앙상블 진행

translstm + tranlstm_kfold + LGBM (가중치 50 - 30 - 20)

→ **최종 7등 리더보드 AUC : 0.8398 ACC : 0.7661**

5 자체 평가 의견

- 잘했던 점
  - 다양한 모델을 시도 구현해보았고 이를 빠르게 할 수 있도록 Model Base 코드를 수정해 상속받아 이루어질 수 있게 하였음
  - LGCN의 Graph embedding을 사용하여 거의 모든 모델의 성능을 향상시켰음
  - 기존 Embedding과 다르게 정오답 여부 및 위치에 따라 다르게 embedding이 될 수 있게 시도하였고 좋은 성능을 얻었음
  - Git 협업을 통해 변경 사항을 바로바로 공유받았고 적용할 수 있었음
- 시도했으나 잘 되지 않았던 것들
  - Long term 효과와 Short term 효과 모두 고려할 수 있도록 Longshort 모델을 고안해보았지만 몇 epochs 이후에 Cuda OOM이 발생하게 되어 사용해보지 못했음
  - SAKTLSTM에 임베딩을 정오답 여부 및 위치에 따라 다르게 임베딩이 되도록 했는데 성능은 좋게 나왔으나 학습 시간이 너무 오래 걸려 제대로 된 튜닝이 이루어지지 못했음
  - Pull Request 시, 관련 내용의 코드리뷰를 다 같이 진행하고자 했으나, 시간에 쫓겨 제대로 진행되지 않음
- 아쉬웠던 점들
  - k-fold에 오류가 있어서 제대로 성능 향상이 이루어지지 않았음
  - 모델 구현 및 수정함에 있어서 명확한 이론적인 근거를 가지고 이루어지지 않아 아쉬움이 있었음
  - Sweep 관리 및 Git 협업 관리가 원활하게 되지 않아 시간을 많이 소비되었음
  - Valid에서 과적합되었으며 리더보드에서 성능이 많이 떨어지는 이유에 대해서 찾지 못했음

개인회고

▼ 김동환_T5028

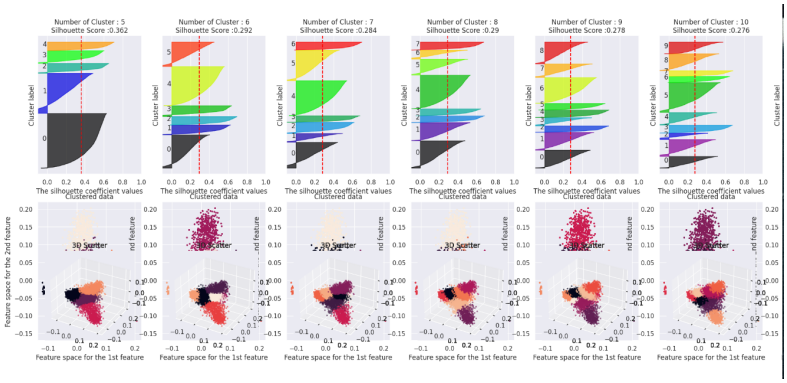
👀 맡은 역할 | 모델링 및 파이프라인

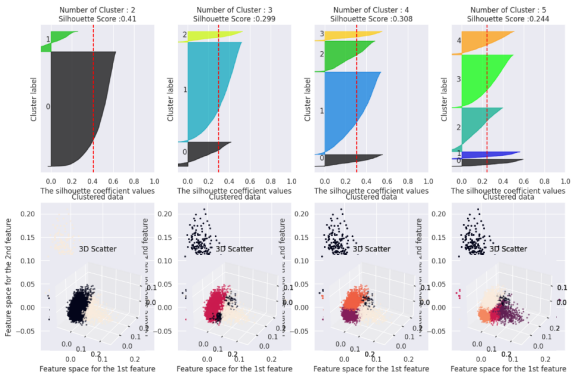
EDA

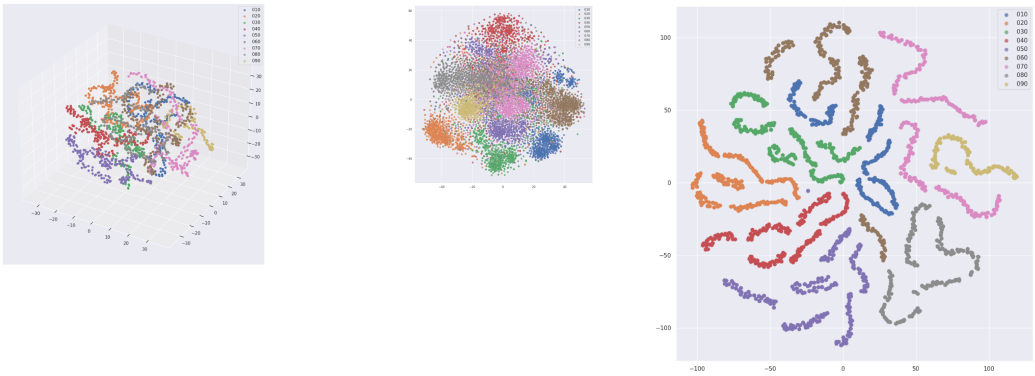
기본적인 임베딩 이외에 모델을 이용해 추가적으로 분석을 진행했다.

LGCN을 통해서 모델을 학습 후 얻어진 Embedding Vectors를 차원 축소와 클러스터링을 통해서 분석을 한 결과 유의미하고 군집화가 되는 것을 관찰했다.

유저의 경우는 문항 정보보다 유의미하게 구분이 되지는 않았지만 특이한 것은 다른 군집과 좀 동 떨어진 군집이 존재하는 것을 관찰할 수 있었다.







문항들의 Sequence 정보를 이용해 Word2vec을 이용해서 문항 사이의 관계를 임베딩해보았다. 라벨은 문항 앞의 ID로 벡터를 한 결과 매우 유의미하게 군집화가 된 것을 알 수 있었다. 그리고 한 가지 특이한 점은 Sequence 정보를 이용한 임베딩과 Graph 정보를 이용한 임베딩의 형상이 많이 달랐다는 것을 관찰할 수 있었다. 그래서 추후 Sequence 모델에 Graph 임베딩을 사용하도록 모델 부분을 수정했고 대부분의 모델이 AUC 기준 약 10~20% 향상되었다.

파이프라인 및 모델 베이스 코드 수정

Add Sliding Window, Data Shuffling, and K-FOLD to Dataloader

Sequence 모델의 파이프라인을 담당해 팀이 사용하기 쉬운 형태로 수정을 했다. Feature가 추가되면 일정한 템플릿에 맞게 기입을 하면 다른 파일 수정없이 돌아갈 수 있도록 작업을 했다. 또한 기본 베이스 코드는 일정 길이가 넘어가는 데이터는 버리게 되어있어 낭비되는 데이터가 많다고 판단해 sliding window와 data shuffling을 추가해 학습할 수 있는 데이터 양과 일반화 성능을 높일 수 있도록 하였다. 실제로 Data Shuffling을 한 것과 안 한 것의 성능 차이가 크게 났었다.

Add Graph Embedding, Residual Connection and Positional Interaction Embedding to Model Base

Sequence 모델의 Parent Class인 Model Base도 이에 맞게 수정을 했고 args로 주는 옵션에 따라 다양한 동작을 할 수 있도록 수정을 했다. Residual Connection, Positional Interaction, Graph Embedding 등을 추가했고 다른 모델들도 상속을 받아 그 기능을 사용할 수 있도록 수정을 했다. 결과적으로 성능이 좋았던 모델들은 residual connection과 graph embedding을 사용하는 모델이었던 것을 보면 결과에 좋은 영향을 주었던 것 같다.

모델링

Add TransLSTM, SAINT, SAKT, SATKLSTM and LongShort model

기본 모델 외에 다양한 모델을 구현해 추가했다. 베이스 모델로 실험을 했을 때 LSTMATTN이 가장 성능이 좋았었고 특히 Transformer단이 feature를 효과적으로 포착하는 것 같았다.

그래서 Transformer의 encoder로 feature의 특성을 추출 후 LSTM으로 sequential한 특징을 잡아낼 수 있도록 TransLSTM을 추가했고 일반적으로 이런 task에 주로 이용되는 SAINT와 SAKT를 추가했다. SAKT는 Sequential 데이터보다는 Tabular 데이터, 정형 데이터를 위해서 고안된 모델이므로 Feed Forward Network 대신에 LSTM을 이용해 데이터의 Sequential한 특징이 반영이 되도록 하였고 일반적인 SAKT는 성능이 좋지 못했지만 SAKTLSTM는 성능이 가장 좋은 모델 중 하나였다. 또한 단순히 문항 정보를 임베딩하지 않고 맞았는지 틀렸는지에 따라 다른 벡터를 갖도록 임베딩을 하고 위치 정보를 반영할 수 있도록 수정했다. 또한 유저의 최근 경향과 장기간의 경향을 모두 반영할 수 있도록 Translstm 기반으로 Long sequence와 Short Sequence를 따로 학습하는 Longshort도 구현을 했지만 메모리 문제로 학습이 이루어지지 못했다. 학습 후 가장 성능이 좋았던 것은 Graph Embedding과 Residual Connection을 이용하는 TransLSTM과 SAKTLSTM이였으며 AUC 기준 약 0.83~0.84를 얻을 수 있었다.

총평 및 아쉬운 점

이번 대회의 목표는 Git을 통한 협업을 잘 사용해보고 Sequential한 데이터, 모델을 다루어본 적이 없기에 성적이 잘 나오지 않더라도 이를 다루는 것에 익숙해지는 것을 목표로 삼았었다. 100프로 다 성취를 했다고 할 수는 없으나 데이터 파이프라인과 ML 모델이 아닌 Transformer 기반의 모델을 맡아 구현을 하고 다양한 시도를 할 수 있어서 이 부분에 대해서는 만족할 만한 성과를 이룬 것 같다. 특히 그저 전 기수가 한 것을 맹목적으로 따라하지 않고 나름의 근거와 목적을 가지고 모델을 구현해보고 실험한 것이 의미가 있었고 이전 기수나 다른 기수들에서는 볼 수 없었던 시도를 많이 한 것 같아 의의가 있었다.

다만 아쉬운 점도 분명히 있었다. 먼저 Sequential한 모델을 이것저것 다 해보다보니 상대적으로 맡은 모델의 개수가 많아 하나의 모델에 집중해 이론적으로 분석하고 깊게 알아보지 못했다. 나름의 근거와 목적을 가지고 시도했다고 했지만 이게 모델의 기저 이론을 완벽히 이해하고 한 것이 아니라 나름의 찝찝함과 아쉬움이 있었다. 또한 모델 무거워져서 학습 시간이 오래 걸린 모델도 많았고 메모리가 터지는 모델도 있어서 Sweep을 통한 튜닝이 제대로 이루어지지 않았다. 또한 k fold가 구현이 조금 잘못된 것인지 하나를 제외한 대부분의 모델이 Kfold 후에 성능이 오히려 낮았었다. 또한 Git을 통한 협업이 처음이고 시간에 쫓기다보니 코드 리뷰나 PR, Pull, Push 등이 명확한 체계를 가지고 이루어지지는 않았던 것이 아쉽다.

▼ 김영서_T504

👀 맡은 역할 | 모델링, 튜닝

총평 및 아쉬운 점

나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

- 우리팀과 나의 학습 목표는 무엇이었나?

  팀 목표 : 실험 및 협업 툴(wandb, Github) 사용, Baseline 이외 모델 선정하여 깊게 분석, 원활한 소통을 통해 서로 많이 배우고 성장하기

  나의 목표 : sequence model에 익숙해지기 + 전반적인 과정 이해하기+ 팀 목표

- 개인 학습 측면

  sequence model의 transformer구조에 대해 보다 깊히 이해해보았습니다.

  (1) transformer의 인코더만 사용하는 모델(bert, lastquery), 인코더-디코더(saint+)를 모두 사용하는 모델, 인코더와 lstm를 결합한 모델 등 다양한 형태로 활용가능함을 학습했습니다.

  (2) batch size, header의 수에 따라 모델이 많이 무거워 질 수 있음을 확인하고, 적절한 range 설정의 중요성을 깨달았습니다.

  (3)NLP 도메인 뿐만 아니라, 시간적 특성을 감안한 문제상황에서도 sequence model을 활용할 수 있음을 확인하였습니다.

- 익숙하지 않더라도 github, wandb를 활용하려고 노력했습니다.

  (1) local, 원격 환경에 대한 이해도가 올라갔고, 기능 단위로 branch를 작업하여 빠르게 작업을 공유하는 것이 좋음을 느꼈습니다.

  (2) wandb sweep을 활용하여, hyperparameter 변화에 따른 성능을 시각화해 볼 수 있어 유용하였고, 팀원들의 running 결과를 team project 보드에서 확인할 수 있어 편리하였습니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 아직 프로젝트 구조를 이해하는데 오랜시간이 걸려서 빠르게 모델을 수정하고, 데이터 구조를 바꾸는데 어려움이 있었습니다.
- Baseline 모델의 구조를 변형하여 다양한 시도를 해보고 싶었지만, 모델에 대한 이해를 하는데 오랜 시간이 걸려서 Baseline을 이해하고 사용하는데 그쳤다는 점이 아쉽습니다.
- wandb를 처음 사용하여 이름 설정, tag 설정을 활용하지 못하였습니다.
- 팀목표였던 baseline 이외의 모델을 구현및 이해를 제대로 수행하지 못한점이 아쉽습니다. 다른 과정에 시간을 많이 소요하기도 하였고, 코드를 수정하면서 전체 구조를 망가뜨릴까봐 적극적으로 작업하지 못했습니다.
- sequence model의 구조, 구현에 초점을 맞추다보니 data 처리 과정을 좀 더 이해해보지 못한 점이 아쉽습니다.
- 프로젝트 중반에 단일 모델을 가지고 다양한 실험 및 결과 제출을 해보지 못한 점이 아쉽습니다. 제출한 실험 결과를 통해 오버피팅의 원인을 찾을 수 있었을 것 같은데, 마지막 얼마남지 않은 제출 기회를 사용하기 어려움이 있었습니다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가

- wandb 이름 설정, tag 설정을 활용하여 여러가지 상황을 비교, 불필요한 시간 낭비를 최소화 하기
- dataload 구조를 이해하고 모델별로 핏한 데이터에 대해 고민해보고 싶습니다.
- 코드를 망가뜨릴까봐 걱정하지 말고 적극적으로 새로운 모델을 추가하는 등, 코드를 수정하기
- 프로젝트 중반에 튜닝하기 전, 다양한 feature 변화, 단일모델에 다양한 augmentation을 주어 결과를 제출해보는 실험을 많이 하여 이후 오버피팅과 같은 문제상황에 대한 인사이트를 낼 수 있는 정보를 많이 만들어두기
- 모델 튜닝 결과를 학습하는 과정에서 보다 자동화된 파이프라인 구축하기
- robust한 모델을 만들기 위한 ensemble 이전에 예측값 사이의 상관관계를 확인하는 과정을 거쳐 보다 근거있는 결과를 도출하기

이전 프로젝트와 비교해봤을 때, 어떤 점이 달라졌는가?

- 데이터 준비, 데이터 전처리, 모델 학습과 모델 성능을 올리기 위한 다양한 시도를 해보면서, 각각의 과정의 우선순위를 확인해볼 수 있었습니다.
- wandb를 활용하여 모델을 튜닝하면서 range 설정을 해보고, 모델의 특징을 살펴볼 수 있었습니다.
- 최대한 팀원들과 소통하려고 노력하였고, github과 zoom등 협업 환경에 보다 익숙해질 수 있었습니다.
- 문제 상황을 공유하여 빠르게 해결하고, 다른 팀원들이 동일한 상황을 예방할 수 있었습니다.
- 특정 모델을 깊게 이해해보는 시간을 가지면서 모델사용의 타당성을 고민해볼 수 있었습니다.

▼ 박재성_T5090

👀 맡은 역할 | EDA & FeatureEngineering, 모델링, 튜닝

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- DKT 대회에서 나의 목표는 wandb와 깃헙과 같은 다양한 툴을 사용해보는 것이었다. wandb를 통해 다양한 실험들의 성능을 다시 한번 확인할 수 있었고 아직도 어색하지만 깃헙에 issu도 올려보고 pull request도 올려보면서 조금 더 협업 툴에 익숙해질 수 있었다.
- LGBM 모델을 선택하면서 새로운 모델이 아니기에 모델에 대한 공부보단 EDA와 피쳐 엔지니어링에 집중했다. 메모리기법으로 유저들이 문제를 풀었던 누적시간이나 누적횟수, 정답률 평균등등을 새로운 피쳐로 만들어서 리더보드 기준 AUC 0.73에서 0.79까지 성능을 올릴 수 있었다. 하지만, 여러 실험을 통해 다양한 피쳐를 만들때 Valid에서 0.86까지 올라갔으며 어떤 피쳐는 0.92까지 올라가게 하였다. test 데이터에서는 매우 떨어졌기에 엄청난 과적합으로 판단했다.
- 대회가 중후반으로 간 시점에서 시퀀스모델을 처음부터 학습할 용기가 없어 Tabluer 모델 중 하나은 XGBoost를 구현했는데 성능은 lgbm과 비슷했지만 속도가 매우 느려서 사용하지는 못했다. 하지만 XGBoost를 통해 Shap 기법을 사용해 볼 수 있었으며 test결과에 대해서 피쳐들의 영향력을 확인해볼 수 있었다.
- wandb를 통해 하이퍼파라미터 튜닝을 진행하였다. 실험을 체계적으로 확인해볼 수 있는 것 뿐만 아니라 베스트 파라미터까지 저장하는 방식을 통해 매우 유용하게 활용하였다. 무엇보다 백그라운드 실행때문이긴하지만 추적시간 250시간을 넘

어본 것도 만족스러웠다

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 아직은 어설프지만 깃허브를 통해 협업을 진행해 본 점이 만족스러웠다. 다만, 컨벤션 정립이 되지 않아 커밋 메세지가 지저분하고 깃 충돌이 많이 났던 것, PR에 대해서 코드 리뷰가 부실한 점이 아쉬웠다.
- 맡았던 LGBM 모델을 사용해서 성능을 최대한으로 끌어내본 점이 만족스러웠다. 다만, 다른팀보다 낮은 성능이 아쉬웠으며 시퀀스모델에 대해 깊게 파보지 못한 것도 아쉬웠다.
- 강건한 모델을 만들기 위해서는 학습데이터에 특화된 피쳐를 많이 만드는 것이 옳지 못하다고 배울 수 있었다. 추가로 테스트 데이터가 744개로 매우 적은 개수를 가지고 있어 public과 private의 점수 차가 심했는데 이렇게 테스트 데이터의 분포를 생각해서 강건한 모델을 어떻게 만들 것인가 고민해 볼 필요성을 느꼈다.
- 한 달 동안 전체 팀목표, 주마다 목표, 개인 목표 등 목표를 바탕으로 계획을 진행하지 못한 것이 아쉬웠다. PM이나 확실한 R&R의 부족이 원인인 것 같다.
- 결과적으로 EDA는 데이터를 보면 볼 수록 느껴지는게 많은데 초반에 빠르게 EDA를 아쉽게하고 다시 EDA로 돌아오지 않았던 것 같음. 머신러닝 프로세스를 빠르게 돌고 피쳐엔지니어링으로 돌아가는게 원래 계획인데 한 사이클을 돌고 얻은 인사이트를 바탕으로 가설 및 EDA를 해보도록 하자
- 시퀀스 모델에 새로운 피쳐를 추가하는 법에 대해서 설명을 듣고 코드를 작성하였다. 그 과정에서 시퀀스 모델과 코드에 대한 공부가 부족했던 탓에 많이 헤맸던 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇일까?

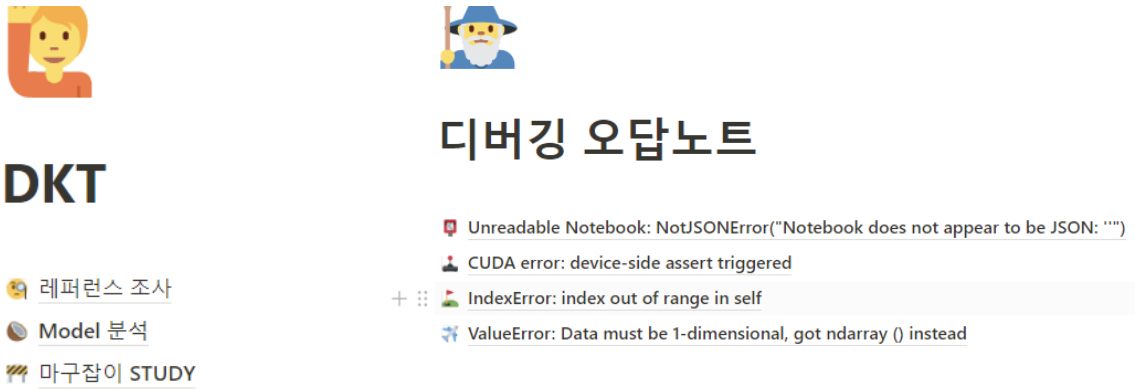
- 깃헙 코드리뷰, 커밋메시지 등 컨벤션을 확실히 정해보기
- 정형데이터를 벗어나 새로운 모델에 대해 도전해보기
- 목표와 역할을 좀 더 체계적으로 정립할 수 있었으면 한다.
- 베이스라인이나 팀원들의 코드를 이해하는 시간이 꼭 필요할 것 같다.
- 매번 대회 마지막에 앙상블을 진행할 시 제일 좋았던 모델 2개를 앙상블 하고 마무리하는데 다음번에는 팀원들이 얘기해주었던 prediction의 분포를 확인해보고 차이가 나는 모델들을 앙상블 해보거나 stacking 앙상블을 진행해보고 싶다.

▼ 전예원_T5184

👀 맡은 역할 | 모델링 및 튜닝

내 학습목표 달성을 위해 노력한 점

- 이번 대회에서의 나의 목표는 베이스라인을 이해하고 주어진 모델 이외의 다른 모델을 추가해 보는 것이었다. 이를 위해 베이스라인의 train과정과 dataloader 과정의 코드를 하나씩 살펴보았고, 주어진 모델이 적용되는 원리를 이해하려고 하였다. 또한, 모델을 적용하기 앞서 EDA를 간단히 하여 feature들의 특성을 살펴보았다.
- 또 다른 목표로는 대회가 진행되는 과정과 공부한 내용을 잘 정리하는 것이었다. 이는 대회를 진행하며 찾아본 reference, 모델의 원리, 데이터 처리 방법 등을 개인 노션에 정리하였고, 디버깅 오답노트도 작성하여 필요할 때 빠르게 찾아볼 수 있게 하였다.



전과 비교하여 새롭게 시도한 변화

- 가장 큰 시도는 베이스라인에 없는 새로운 모델인 CatBoost와 TabNet을 추가해본 점이다.
- 저번 대회에서는 wandb만 사용하였지만 이번에는 같은 팀원분이 적용해주신 sweep을 사용하였다. config 파일로 찾고자 하는 hyper parameter의 범위를 설정하고 자동으로 찾아볼 수 있게 하였다.
- 저번 대회와 동일하게 Github으로 협업하였지만 이번 대회에서의 새로운 시도는 pull request를 날려본 것이었다.
- 코드를 작성하고 디버깅할 때 조금 더 편리해지는 요소를 적용하였다. tqdm으로 반복 루프에 대해서 진행 상황을 확인할 수 있었고, pdb로 실행 중간에 할당되어 있는 값들을 확인할 수 있었으며, nohup이나 tmux로 터미널이 종료되어도 코드가 백그라운드에서 돌아갈 수 있도록 하였다.

아쉬웠던 점 및 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 점

- 이번에는 feature들을 많이 분석해보지 못한 것 같다. 훈련과정에 유의미한 feature들인지 고민해보고 가설을 세워서 적용해 봤어야 하는데 모델에 집중하느라 이런 과정을 깊게 진행하진 못했던 것 같다. 다음 프로젝트에서는 이런 상관관계를 고려한 feature engineering과 feature selection을 하고 싶다.
- 맡은 모델의 성능을 많이 향상시키지 못했다. 다음 프로젝트 진행 전에 해당 부분을 다시 보고 원인을 찾아보도록 할 것이다.
- Pull Request를 날림으로서 팀원들이 공유하고자 하는 부분을 확인하고 merge하였지만, 자세한 코드를 확인해보지는 못했던 것 같다. 다음 프로젝트에서는 코드 리뷰도 하면서 좋은 코드를 작성해보도록 노력하고 싶다.
- 훈련과정에서 overfittting 발생 시 valid의 결과와 실제 제출하는 결과의 차이가 많이 날 수 있다는 것을 몸소 깨달았다. 다음에는 더 많이 제출하여 맞는 방향으로 흘러가고 있는지 확인할 것이다.
- 파이썬/파이토치 기반의 추천 라이브러리인 Recbole에 대해서 알게 되었는데 다음 프로젝트에서 적용해보고 싶다.
- 단순히 weight를 주어 앙상블 하는 것 이외에 다른 앙상블 기법을 적용해보고 싶다.

▼ 진성호_T5209

👀 맡은 역할 | 모델링 & Git 관리 & HyperParameter 기초 작업

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

강의 진행 중 나왔던 Riiid 대회 1등 모델인 LastQuery 가 인상 깊어 관련 내용 조사하기 시작했다.

주어진 데이터가 시퀀스 데이터인 이상, 시퀀스 길이를 최대한 길게 늘려 학습시키면 그 시퀀스 흐름에 따라 앞으로 풀 문제의 정답을 더 잘 예측할 수 있을거라 가설을 세웠고, 더불어 feature engineering 을 많이 할 필요 없어, 빠르게 모델을 돌려볼 수 있을거라 생각했다.

강의때 공유된 자료와 논문을 참고하며 모델을 구현했지만, 학습이 전혀 되지 않았다.

matrix 의 형태 변환이 잘 이루어 지지 않는 부분이 있었고, 수정하여 학습 진행하는 것을 확인했다. 하지만 ACC 가 0.5 이상 상승하지 않는 문제점이 있었다.

hyperparameter 를 잘 바꾸면 올라가지 않을까 가정하고, tuning base code 를 빠르게 완성시켜 확인해봐야겠다고 생각했다.

이전 대회 때 사용했던 Hyperopt 와는 다른 tool 을 사용하고 싶었기에, WandB 의 Sweep 을 찾아보고 대회 base code 에 적용하여 돌아가는 것을 확인했으며, 이후 Lastquery 모델에 적용하여 튜닝을 진행했다.

하지만 성능은 향상되지 않았고, 시간은 제한적이기에 우선순위를 조정하여 다른 모델 튜닝을 돌렸고, 팀원들과 협력하여 대회를 마무리할 수 있었다.

모델링 및 튜닝 작업을 통해 결과를 잘 뽑아내는 것도 중요하지만, 그런 결과를 내기 위해 어떠한 과정을 거쳤는지도 중요하다.

프로젝트는 팀플로 진행되고, 협업은 필수이기에 Git 협업 프로세스를 만들어 진행했다.

branch 를 구분(main/develop/개인), pull request 를 통해 구현한 기능 설명, 코드리뷰 이후 merge 를 진행하자는 제안을 했고, 다들 동의하여 그대로 진행되었다.

규모도 크지 않고, 단기/단발성인 프로젝트이기에, git을 사용한 협업 프로세스가 다소 거추장스러웠지만, 팀원 모두가 잘 따라주어 결과적으로 잘 진행된 것 같다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

우선 LastQuery, 처음 경험하는 모델이긴 했지만 가이드도 있었고 잘 구현했다고 생각했는데, 결과적으로 성능이 제대로 나오지 않은 것 같아 매우 아쉬웠다.

결과적으로 구현되긴 했지만, Sweep 을 적용하는데 많은 시행착오가 있었다. 계정과 연동이 안된다거나, 측정 metric 설정이 제대로 안된 것, sweep 자체적으로 에러가 났을 때 디버깅이 잘 되지 않은 것 등등. 많은 시도로 인한 시간적 한계를 많이 느낀 것 같다. 구현 이후에도 튜닝 실험을 진행하며 결과들이 정리되지 않고 난잡하게 올라가 한눈에 딱 정리되지 않는 것이 아쉬웠다. 그래도 서로 다른 도구를 두 개 사용해 보았으니, 이후에 또 다른 tool 을 사용하는데 있어 빠르게 적응할 수 있을거라 생각한다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

mlops 를 작은 기능부터 구현한다는 느낌으로, 어떤 model 과 data 가 들어오더라도 사용할 수 있는 나만의 hyperparameter tuning module 구현을 시도할 예정이다.

더불어, 튜닝 실험을 자동화 시키고, 체계적으로 정리할 수 있는 구조도 고민하여, 최종적으로는 mlops 의 전체적인 그림을 혼자서 구현해보려 한다.