랩업리포트

Wrap-Up Report, RecSys04 Team RECCAR

▼ 목차

▼ Part 1. 프로젝트 Wrap Up

▼ 1-1. 프로젝트 개요

1-1-1. 프로젝트 목표

총 7,442명의 사용자가 존재합니다. 이 때, 이 사용자가 푼 마지막 문항의 정답을 맞출 것인지 예측하는 것이 최종 목표입니다.

1-1-2. 프로젝트 구조

```
📦level2_dkt_recsys-level2-recsys-04
 ┣ 📂boosting
```

```
│ ├ 📜cat_boost.py
│ ├ ...
│ └ 📜xg_boost.py
├ 📂Boosting_ipynb
│ ├ 📜bae_xgboost.ipynb
│ ├ ...
│ └ 📜suyeon_LGBM2.ipynb
├ 📂EDA
│ ├ 📜bae_eda.ipynb
│ ├ ...
│ └ 📜ysh_eda.ipynb
├ 📂gkt
│ ├ 📜code_parse_ysh.ipynb
│ ├ 📜EDA.ipynb
│ ├ 📜layers.py
│ ├ 📜metrics.py
│ ├ 📜models.py
│ ├ 📜processing.py
│ ├ 📜README.md
│ ├ 📜train.py
│ ├ 📜train2.py
│ ├ 📜utils.py
│ └ 📜__init__.py
├ 📂MF
│ ├ 📜suyeon_recommenders.ipynb
│ ├ 📜...
│ └ 📜suyeon_Surprise2.ipynb
├ 📂saint_plus
│ ├ 📜augment.ipynb
│ ├ 📜augmentation.py
│ ├ 📜config.py
│ ├ 📜dataset.py
│ ├ 📜lab_ysh.ipynb
│ ├ 📜README.md
│ ├ 📜requirements.txt
│ ├ 📜saint.py
│ ├ 📜train.py
│ └ 📜HSUNEH_SAINT.ipynb
├ 📜.gitignore
├ 📜README.md
└ 📜__init__.py
```

1-1-3. 데이터셋 구조

주요 데이터는 `.csv` 형태로 제공되며, train/test 합쳐서 총 7,442명의 사용자가 존재합니다.

- `test_data.csv` : 6698명의 사용자(user)에 대한 정보를 담고 있는 메타데이터입니다.
- `train_data.csv` : 743명의 사용자(user)에 대한 정보를 담고 있는 메타데이터입니다.

위 데이터는 아래의 6개의 columns를 가지고 있습니다.

- `userID` 사용자의 고유번호입니다. 총 7,442명의 고유 사용자가 있으며, train/test셋은 이 `userID`를 기준으로 90/10의 비율로 나누어졌습니다.
- `assessmentItemID` 문항의 고유번호입니다. 총 9,454개의 고유 문항이 있습니다. 이 일련 번호에 대한 규칙은 DKT 2강 EDA에서 다루었으니 강의 들어보시면 좋을 것 같습니다.

- `testId` 시험지의 고유번호입니다. 문항과 시험지의 관계는 아래 그림을 참고하여 이해하시면 됩니다. 총 1,537개의 고유한 시험지가 있습니다.
- `answerCode` 사용자가 해당 문항을 맞췄는지 여부에 대한 이진 데이터이며 0은 사용자가 해당 문항을 틀린 것, 1은 사용자가 해당 문항을 맞춘 것입니다.
- `Timestamp` 사용자가 해당문항을 풀기 시작한 시점의 데이터이다.
- `KnowledgeTag` 문항 당 하나씩 배정되는 태그로, 일종의 중분류 역할을 합니다. 태그 자체의 정보는 비식별화 되어있지만, 문항을 군집화하는데 사용할 수 있습니다. 912개의 고유 태그가 존재합니다.

1-1-4. 협업 도구

Slack

파일 공유, 논의해야할 사항을 Slack을 이용해 공유하였습니다.

Github

• Git Flow 브랜치 전략

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cdd024e3-8449-48c4-8a0e-8a4f7860075d/bandicam_2022-12-11_15-00-29-433.mp4

• Github Issues 기반 작업 진행

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/79476f08-3063-4f04-ac2d-c6488a638667/bandicam_2022-12-11_15-34-51-594.mp4

• Github Projects의 칸반 보드를 통한 일정 관리

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5813722f-306e-472c-b04e-49325629b484/bandicam_2022-12-11_15-07-25-690.mp4

Notion

실험결과 Parameter와 RMSE를 쉽고 명확하게 공유하기 위해 노션을 활용했습니다.

1-1-5. 개발환경(AI Stage Sever)

- OS: Ubuntu 18.04.5 LTS

- GPU: Tesla V100-SXM2-32GB

▼ 1-2. 프로젝트 팀 구성 및 역할

- 김성연_T4040(팀장)

⇒ EDA, 전반적인 팀 프로젝트 타임라인 잡기, 부스팅 모델 적용 작업.

- 배성재_T4097(팀원)

⇒ 협업을 매끄럽게 하기 위한 코드 정리 및 Git 담당하기, ELO 등 피처엔지니어링 진행.

- 양승훈_T4122(팀원)

⇒ 전반적인 그래프 모델 탐색, GKT, Saint+ 모델 적용 작업

- 조수연_T4208(팀원)

⇒ EDA 진행, 부스팅 모델과 MF 모델 적용 작업.

- 홍재형_T4233(팀원)

⇒ GKT 모델 적용 작업, Optuna, K-Fold Cross Validation 등을 이용한 모델 고도화.

- 황선태_T4236(팀원)

⇒ 전반적인 시퀀셜 모델 탐색, Saint+와 LightGCN 모델 적용 작업.

▼ 1-3. 프로젝트 수행 절차 및 방법

1-3-1. EDA

유저 데이터 분석

- 총 유저 7442명 중 10%인 744명 유저가 test, 나머지 669명 유저가 train 데이터 셋으로 구성되어 있습니다.
- 유저별로 마지막 문제를 풀었는지 맞추는 방식 입니다.
- 가장 문제를 많이 푼 유저는 1860개, 적게 푼 유저는 9개 문제를 풀었습니다. test 내 유저 기준으로 최소 14문제는 풀었습니다.
- 가장 많이 풀린 문제는 500번, 가장 적게 풀린 문제는 46번 풀렸습니다.- 판다스의 to_datetime 함수를 이용해 문제 푼 시점을 다양한 날짜 변수 별로 살펴봤습니다.

  => 푼 달이나 시간 별로 정답률이 유의미하게 달라집니다.- 유저 단위로 푼 시간이 인접한 문제의 시간 차이를 구할 수도 있음.(solve_time)

  => 처음 푼 문제는 기록이 없지만 test 데이터는 마지막에 푼 문제이기 때문에 괜찮음.

  => 모든 유저가 문제를 한번에 다 풀진 않기 때문에 크게 튀는 값은 후처리 필요함.

- 'assessmentItemID' 변수에 대해.

  => 'A/0/6/0/001/001' -> (A/0/학년이나 난이도/0/시험지번호/문제번호)

=> 첫 값은 모두 A, 두번째와 네번째 값은 모두 0, 따라서 의미 없음.

=> 세번째 값은 학년으로 추정됨. 숫자가 커질수록 대체로 낮아짐.

=> 5~7번 값은 시험지 번호, 8~10번 값은 문제 번호.

=> 뒷 문제일 수록 정답률이 떨어지는 것으로 보아 어려운 문제로 추측. 그리고 1번 문제는 solve_time 변수가 튀는 값이 상당히 많음.

1-3-2. Modeling

LGBM

- paper-link : https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
- User based 추천 모델
- 최고 성능 : 0.7853

CatBoost

- paper link - https://arxiv.org/pdf/1706.09516.pdf
- Categorical 변수에 최적화 된 Boosting , Item based 추천 모델
- 최고 성능 : 0.7853

MF

- paper link - https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf
- User-Item 행렬을 활용해서 예측하는 모델
- Surprise library - SVDpp 모델 사용
- 최고 성능 : 0.7927

LightGCN

- paper link - https://arxiv.org/abs/2002.02126
- 베이스라인으로 주어진, Graph를 활용한 모델이다.
- UserID와 AssessmentItemID만을 사용해 Graph를 생성하고 정답률을 예측하는 것이 특징입니다.
- 하이퍼파라미터 튜닝을 통해 성능 개선을 시도했고, 효과가 있었습니다.
- 최고 성능 : 0.8272

SAINT+

- paper link - https://arxiv.org/pdf/2010.12042.pdf
- 유저가 푼 문제들의 Sequence data를 트랜스포머 모델을 활용하여 분석하고 다음 문제의 정답률을 예측
- RIID 대회에서 사용했던 SAINT모델(https://github.com/Shivanandmn/SAINT_plus-Knowledge-Tracing-)을 깃 클론 하여 수정하여 사용했습니다. 현재 데이터 셋을 이에 맞추기 위해 전처리 작업을 진행했으며 이와 함께 데이터 증강하는 작업도 진행했습니다.
- 데이터 증강은 유저가 마지막으로 푼 문제 정보를 제외한 항목을 따로 만들어, 이를 새로운 유저로 생성하여 증강하는 방식을 사용했습니다.
- 최고 성능 : 0.71

GKT

- paper link - https://rlgm.github.io/papers/70.pdf
- 데이터가 Graph와 Sequential 모델에 모두 적합한 형태라고 생각해서, 그 둘을 결합한 GKT 모델을 적용했습니다.
- 데이터를 유저 별로 문제 푼 시간 순서로 나열했습니다. 학습 속도와 성능을 높이기 위해 데이터 sequence의 최대 길이를 제한하고, 그 길이를 초과하는 Sequence들은 잘라서 데이터 증강에 활용했습니다.
- 그래프 생성 방식에는 Dense / Transition / MHA / VAE / PAM 등이 있고, 그래프 생성 기준을KnowledgeTag로 설정했습니다.
- 모델 학습 과정에서 많은 시간 소요와 에러 발생으로 어려움이 있었습니다. 그래프 생성 기준을 AssessmentItemID로 하거나, Batch size를 늘리면 CUDA out of Memory 발생했습니다.. 또한 생각보다 성능이 높지 않았습니다.
- 최고 성능 : 0.7690

1-3-3. Feature Engineering

ELO

kaggle Riiid! 대회 simple elo rating

https://www.kaggle.com/code/stevemju/riiid-simple-elo-rating/notebook

`elo` : 유저와 아이템 elo를 sigmoid하여 얻은 문제 난이도 값
`elouser` : 유저의 문제풀이 수준
`eloitem` : 문제의 난이도
`elotag` : tag 별 난이도
`elotest` : testid 별 난이도

user / item logging

`answer_mean` : 유저가 평균적으로 얼마나 맞췄는지

`answer_cnt` : 유저가 몇 문제 풀었는지

`time_mean` : 유저가 평균적으로 몇 분안에 문제를 맞췄는지

`tag_mode` : 유저의 최빈 태그

`user_item_mean` : 유저가 지금까지 푼 문제에 평균 정답률은 어떻게 되는지

`item_mean` : 문제 평균 정답률

`item_sum` : 문제 총 풀어진 개수

`item_time_mean`: 문제 평균 풀이시간

`last_answerCode_N` : N번째 문제 전 정답 여부 (0,1)

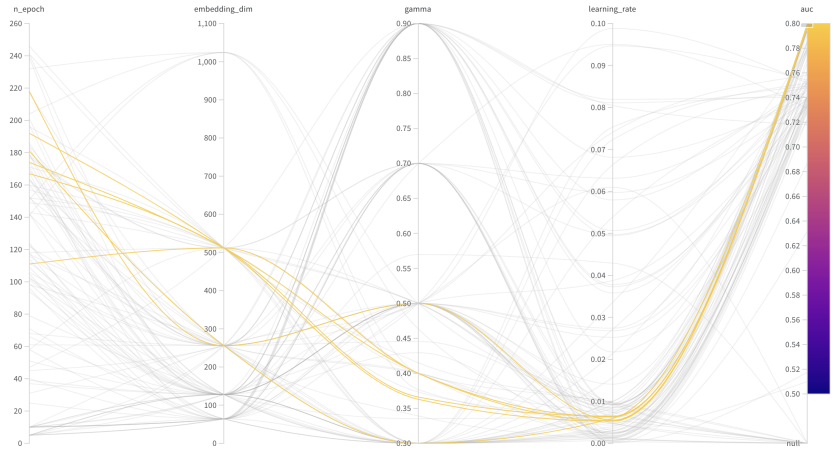
1-3-4. 모델 고도화

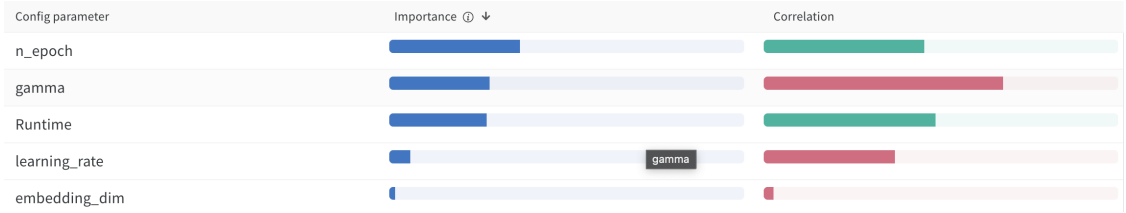
Optuna

- 비교적 빠르게 돌아가는 부스팅 모델에서 파라미터 튜닝을 도와 주는 optuna를 이용하면 파라미터에 랜덤값을 부여해서 학습시킬 수 있습니다.
- random_state, bagging_temperature, max_depth, random_strength, l2_leaf_reg, min_child_samples, max_bin 범위를 넓게 지정하고 성능이 잘 나오는 값들 사이로 범위를 좁히며 최적화를 진행했습니다.
- max_depth는 숫자가 커지면 학습 시간이 길어져서 성능이 좋았던 11로 고정했습니다.
- hyper parameter tuning을 할 때는 iterations을 500으로 지정해주었고 제출할 때에는 4000으로 지정했습니다.
- 처음에는 넓게 범위를 지정하였고 성능이 잘 나오는 값들 사이로 범위를 좁히는 방식으로 최적화 시켰습니다.
- iterations을 줄이고 learning_rate를 늘려서 학습에 걸리는 시간을 줄였습니다.
- 이진 분류기 성능 평가 방법인 AUROC score를 0.8289에서 0.8350까지 올렸습니다.

wandb sweep

LightGCN 을 wandb sweep을 통해 고도화 했습니다.





- 주요 파라미터들을 설정하고, 실험 값을 설정한 뒤에 random 방식으로 진행시켰습니다.
- wandb로 parameter 와 결과값의 관계를 파악하기 쉬웠습니다.
- parameter가 어떤 값을 가질 때 좋은 값을 가지는지 경향을 파악할 수 있었고, 최종 parameter 값 선택에 도움이 되었습니다.

K-Fold Cross Validation

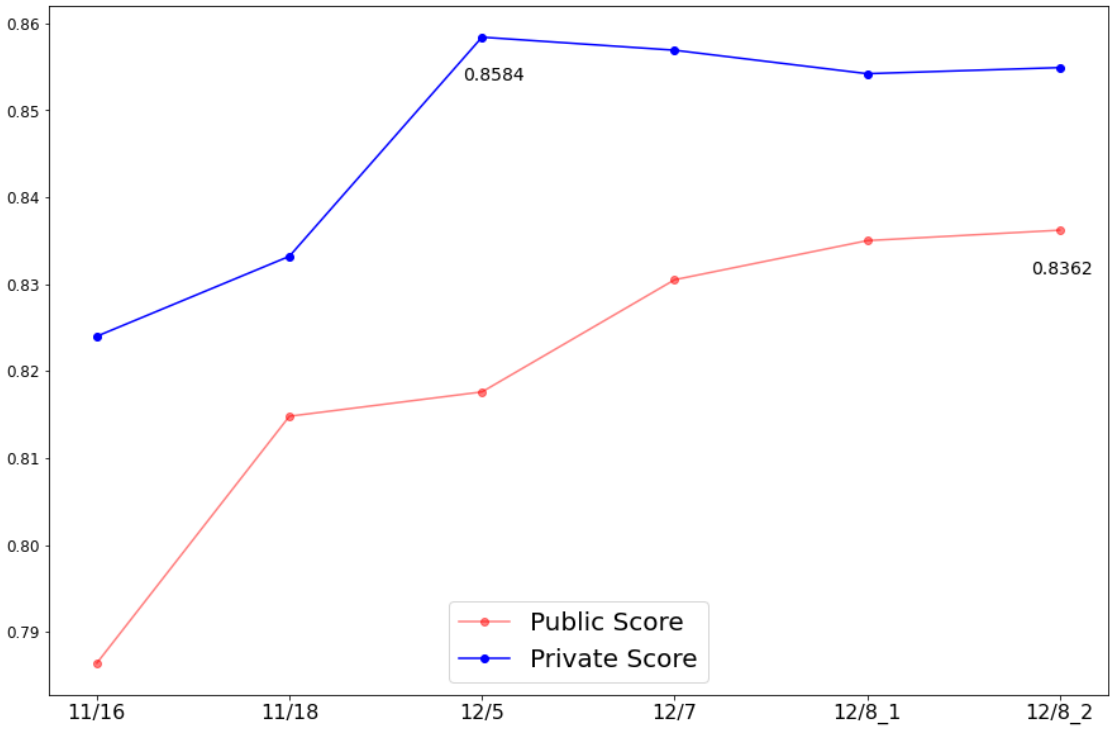
- 데이터를 k개로 분할한 뒤, k-1개를 학습용 데이터 세트로, 1개를 평가용 데이터 세트로 사용하는데, 이 방법을 k번 반복하여 k개의 성능 지표를 얻어내는 방법입니다.
- scikit learn에서 제공되는 K-Fold, StratifiedKFold 뿐만아니라 user를 기준으로 나눠주는 형식도 만들어서 학습을 시켰습니다.

앙상블

- 단순히 모델별 가중평균을 한다면 auc가 왜곡될 수 있습니다.
- 모델별로 min값이 0이고 max값이 1인 MinMaxScaler를 적용한 뒤 min-max scaler를 적용하여 auc 지표에 맞는 예측값을 뱉는 앙상블을 할 수 있었습니다.

▼ 1-4. 프로젝트 수행 결과

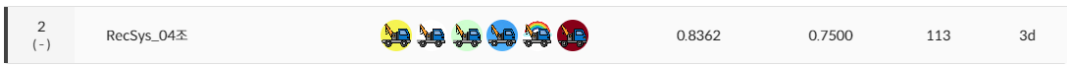
시간 별 모델 public, private auc score 기록



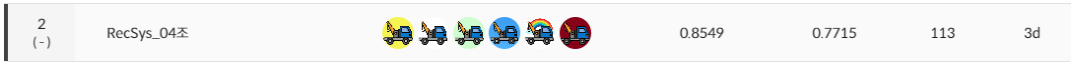
	모델	public auroc	private auroc
11/16	Catboost	0.7864	0.8240
11/18	Catboost FE	0.8148	0.8332
11/21	XGboost	0.7868	0.8109
11/23	lgbm baseline	0.7345	0.7005
11/28	lgbm 튜닝	0.7853	0.8245
12/1	GKT-dense	0.7175	0.7134
12/5	Catboost Elo	0.8176	0.8584
12/6	SVDpp	0.7889	0.7927
	SAINT +	0.7104	0.7104
12/7	cat0.85 mf0.1 lgbm0.05	0.8305	0.8569
	lightGCN 튜닝	0.7869	0.8272
	Catboost Elo 튜닝	0.8188	0.8610
12/8	cat 튜닝	0.8350	0.8542
	cat 튜닝 0.9 mf 0.1	0.8362	0.8549

대회 결과

- public



- private



▼ 1-5. 자체 평가 의견

잘했던 점

- Public Score와 Private Score 모두 2등으로 순위가 많이 바뀐 대회에서 robust한 모델 구축에 성공했습니다.
- 완벽하진 못했지만 Git Branch 전략을 효과적으로 사용했습니다.
- Boosting 기반, Sequential 기반 그리고 Graph 기반 다양한 모델을 시도 했습니다.
- 모델 고도화 시간을 충분히 확보하여 순위를 올릴 수 있었습니다.

시도 했으나 잘 되지 않았던 것들

- 유사한 Riiid 대회에서 수행한 딥러닝 전략들을 이용해 가시적인 성과를 내지 못했습니다.
- 서로의 코드를 잘 이해해보자는 목표로 pair를 이루어 진행했는데 효과가 크지 못했습니다.
- Git 이슈와 프로젝트 기반 개인 상황 공유를 시도했으나 끝에 가서는 조금 흐지부지 됐습니다.

아쉬웠던 점들

- 추천 외부 대회가 있어 온전히 대회에 집중하지 못했습니다.
- 논리적인 접근이 조금 부족했습니다.
- 성과를 위해 너무 마음을 급하게 먹어 모델링에 온전히 몰입하지 못했습니다.
- 딥러닝 모델을 low-level부터 구현해보지 못했습니다.

▼ Part 2. 개인 회고

▼ 김성연

- **이번 프로젝트에서 나의 목표는 무엇이었는가?**

1) 새로운 팀원들과 함께 협업이란 어떤 것인지 익히는 시간을 가지고 싶었습니다.

2) 딥러닝 엔지니어링 실력을 늘리고 싶었습니다.

3) 저번 대회와 마찬가지로 높은 순위를 기록하고 싶었습니다.

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**