boostcamp ai tech

Deep Knowledge Tracing

사용자의 문제 풀이 기록을 바탕으로

Recsys 04 RECCAR 조

김성연 배성재 양승훈 조수연 홍재형 황선태

목차

- EDA
  - •유저 데이터
  - •Timestamp
  - •assessmentitemID
- 모델 구상
  - • 베이스라인 Code의 DL Model 실험
  - • 평점의 평균 값을 예측에 사용
  - • 데이터의 특성
  - • 간략한 데이터 전처리
- 모델링
  - • Saint 모델
  - • MF 모델
  - • LGBM 모델
  - • Graph 모델
  - • Boosting 모델
- 모델 고도화
  - • Optuna
  - • Ensemble
- Q & A

EDA

Dataset

총 7442명의 유저 데이터
Train 유저 수 : 6698
Test 유저 수 : 744

유저별 문제 풀이 수

```
1  print(dat['userID'].value_counts().max())
2  print(test['userID'].value_counts().min())
3  print(dat['userID'].value_counts().min())
```

✓ 0.1s

```
1860
15
9
```

문제별 풀이 횟수

```
1  print(dat['assessmentItemID'].value_counts().max())
2  print(dat['assessmentItemID'].value_counts().min())
```

✓ 0.4s

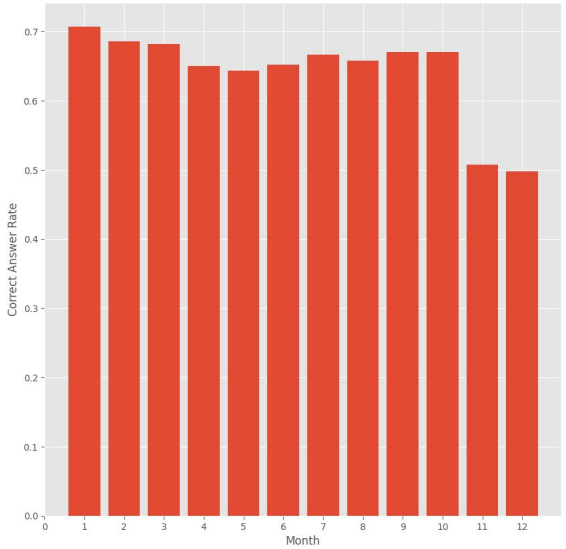
```
500
46
```

EDA

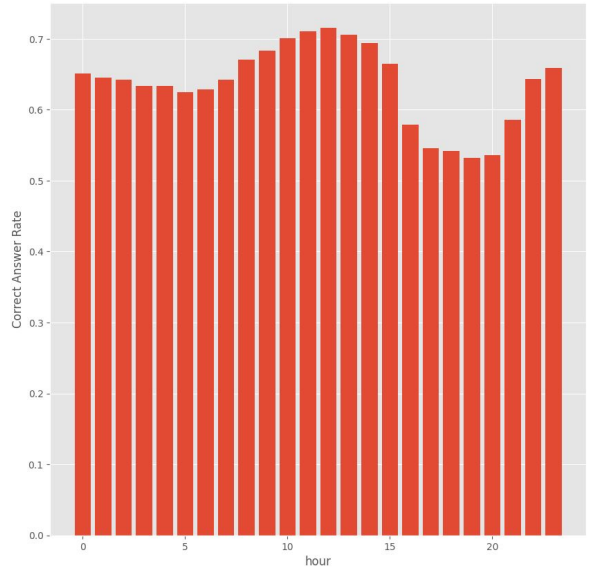
Dataset

Timestamp 를 다양한 날짜 변수로 확인

month와 정답률



hour과 정답률



유저 단위로 인접한 Timestamp로 유저의 문제 풀이 시간을 알아볼 수 있다. (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

EDA

Dataset

assessmentItemID

	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
...	...	...	...	...	...	...

A060001001

A 0 6 0 001 001 (A/0/학년 혹은 난이도/0/시험지번호/문제번호)

- 첫번째 값은 모두 A 이므로 의미가 없다.
- 세번째 값은 학년이나 난이도로 추정
- 5~7번째 값은 시험지 번호, 8~10번 값은 문제 번호

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

모델 구상

Test set

Vaild set 생성

Vaild set

Valid set을 Test set 환경과 최대한 유사하게 구성

Test Set

```
_train = dat[dat['answerCode'] >= 0]
_test = dat[dat['answerCode'] < 0]
```

- **Test data에서 744개의 userID 별 마지막으로 푼 문제**

Valid Set

```
_train['train_valid'] = 0
_train.loc[_train.drop_duplicates(subset='userID', keep = 'last').index, 'train_valid'] = -1
_valid = _train[_train['train_valid'] == -1]
_train = _train[_train['train_valid'] == 0]
```

- **Test set을 제외한 모든 userID 별 마지막으로 푼 문제 7442개**

중요한 건 꺾이지 않는

Validation Set

모델 구상

데이터 접근 방식

sequential

SAINT+

- **userID 별 Timestamp** 순서 풀이 기록을 바탕으로
  마지막 문제 예측

Boosting

- 유저 단위로 그룹핑 한 뒤 다양한 파생변수를 만들어
  부스팅 모델을 적용하여 예측

GKT

- 문제 **& UserID &** 정답 여부 간의 관계를 바탕으로 예측

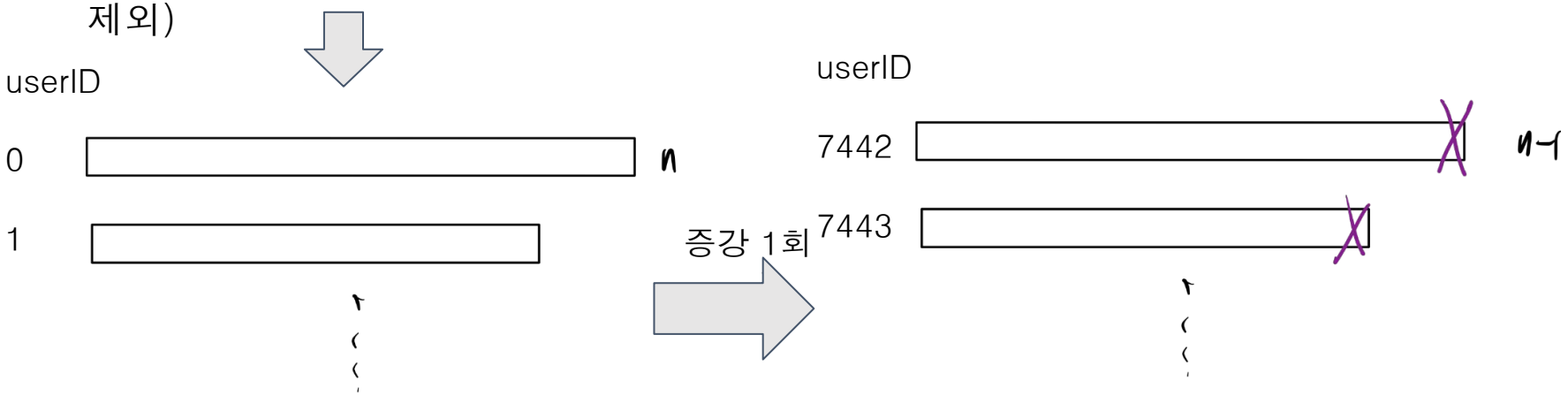
모델 구상

데이터 접근 방식

sequential

데이터 증강

train + test(answer == −1 제외)




```
1  data[data['userID']==0]
```
✓ 0.4s

	userID	assessmentItemID	testId	answerCode
0	0	A060001001	A060000001	1
1	0	A060001002	A060000001	1
2	0	A060001003	A060000001	1
3	0	A060001004	A060000001	1
4	0	A060001005	A060000001	1
...	...	...	...	...
740	0	A080129002	A080000129	1
741	0	A080129003	A080000129	0
742	0	A080129004	A080000129	1
743	0	A080129005	A080000129	0
744	0	A080129006	A080000129	0

745 rows × 6 columns

```
1  datasum[datasum['userID']==7442]
```
✓ 0.7s

	userID	assessmentItemID	testId	answerCode
0	7442	A060001001	A060000001	1
1	7442	A060001002	A060000001	1
2	7442	A060001003	A060000001	1
3	7442	A060001004	A060000001	1
4	7442	A060001005	A060000001	1
...	...	...	...	...
739	7442	A080129001	A080000129	1
740	7442	A080129002	A080000129	1
741	7442	A080129003	A080000129	0
742	7442	A080129004	A080000129	1
743	7442	A080129005	A080000129	0

744 rows × 6 columns

모델 구상

데이터 접근 방식

Non-sequential

전반적인 시퀀셜 하지 않은 모델들.

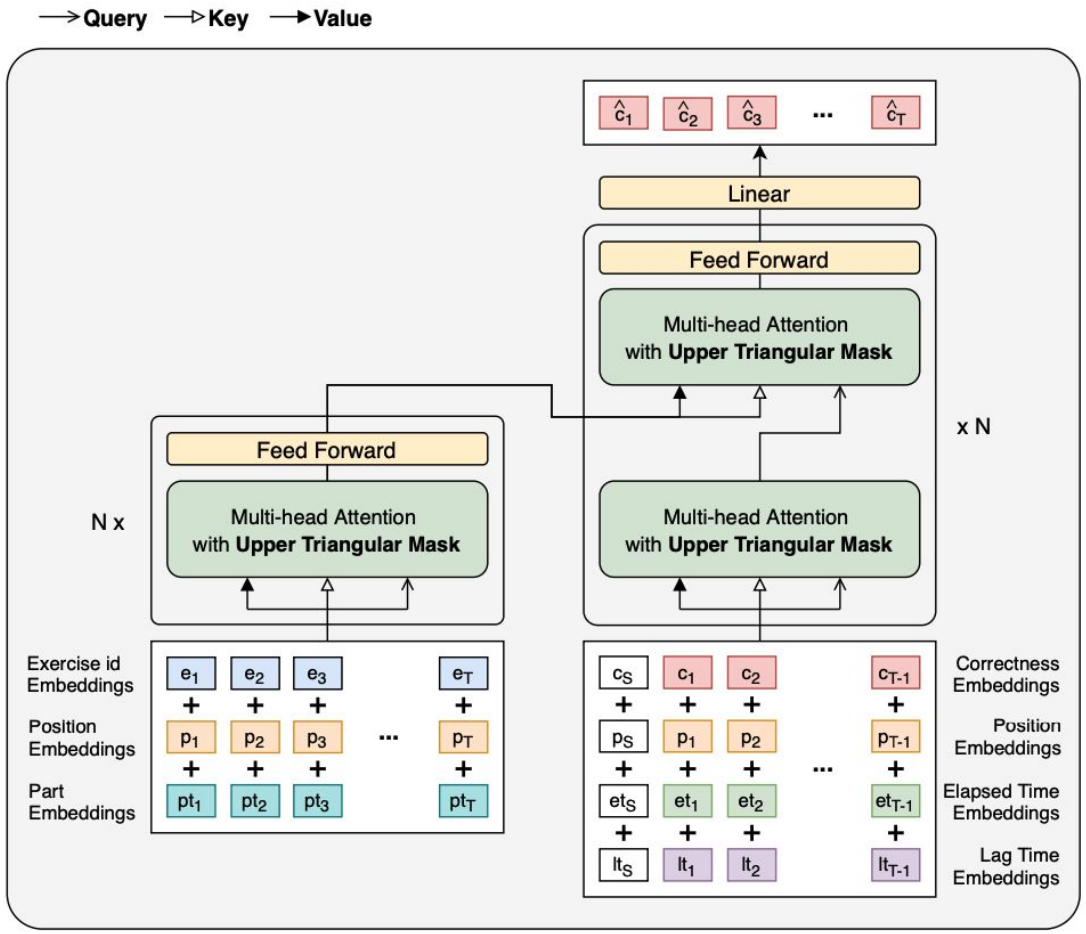
Boosting

- **문제 (assessmentItemID) 정보 단위로 모델을 만들어서 예측**

LightGCN / MF

- **문제 & UserID & 정답 여부 간의 관계를 임베딩을 바탕으로 예측**

모델링 - 1.SAINT+



- **DKT** 문제 해결 방법 중

  Transformer 기반, 문제 푼 시간 중요시

- 학생이 푼 문제들의 **Sequential** 정보가 답에 도움이 되지 않을까?

1. SAINT+

- 우리 대회와는 다른 **Data Input**

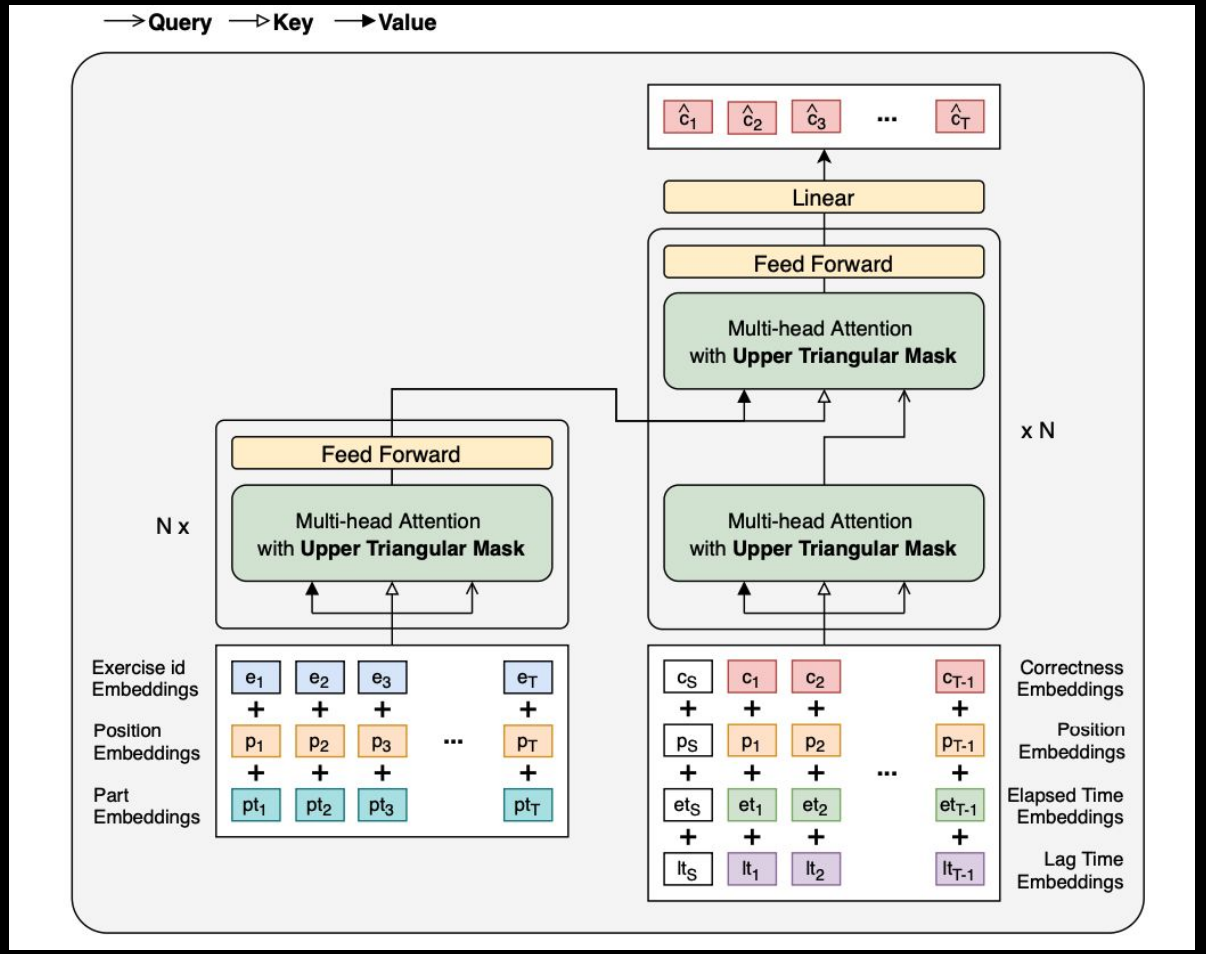
  (문제 푼 시간, 카테고리 1개 필요)

-> 전처리 통해

문제 푼 시간 각각 계산,

KnowledgeTag 를 카테고리로 선정

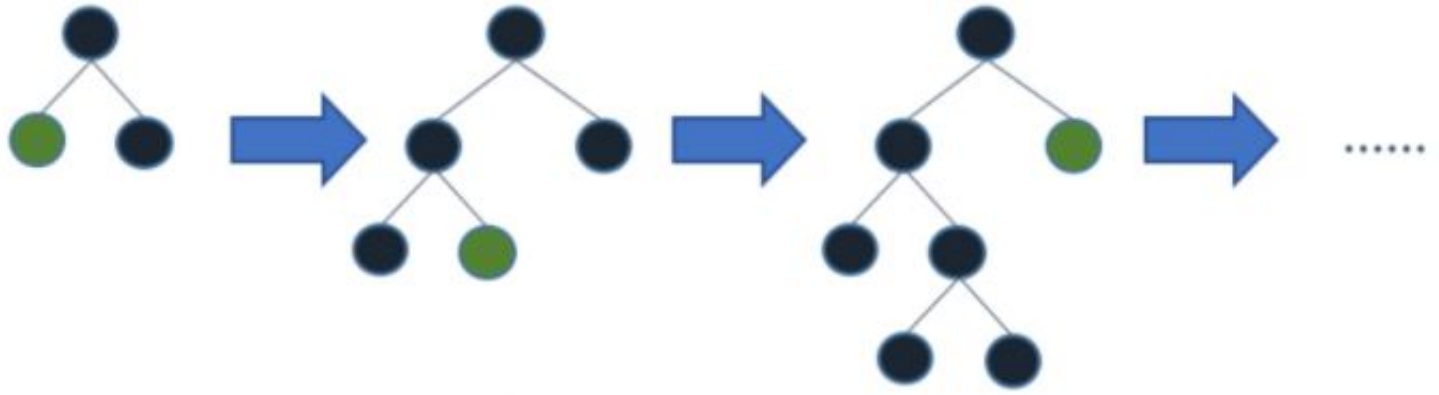
- Pytorch Lightning



모델링 - 2.LGBM (유저 단위 부스팅 모델)

LightGBM

- Light GBM(Light Gradient Boosting Machine)은 트리 기반의 학습 알고리즘인 gradient boosting 방식의 프레임워크이다.
- 유저의 특성을 요약한 수치형 변수들에 강한 LGBM을 사용
- **leaf-wise tree growth**의 경우 loss 변화가 가장 큰 노드에서 분할하여 성장하는 수직 성장방식이다. 비대칭적인 트리를 생성하지만 loss 변화를 기준으로 분할하기 때문에 level-wise에 비해 예측 오류손실이 작거나 빠르게 도달할 수 있다.
  단, 데이터가 적을 경우 overfitting의 가능성이 있다. 따라서 데이터 증강을 통해서 이 위험을 피하고자 하였다.



Leaf-wise tree growth

2. LGBM

Feature Engineering

`answer_mean`: 정답 평균

`answer_cnt`: 유저의 누적 정답문제 수

`last_answerCode(n)`: 유저의 최근 n개 문제의 평균 정답률

`time_mean`: 유저가 푼 문제들의 평균 풀이시간

`time_sum`: 유저의 누적 문제 풀이 시간

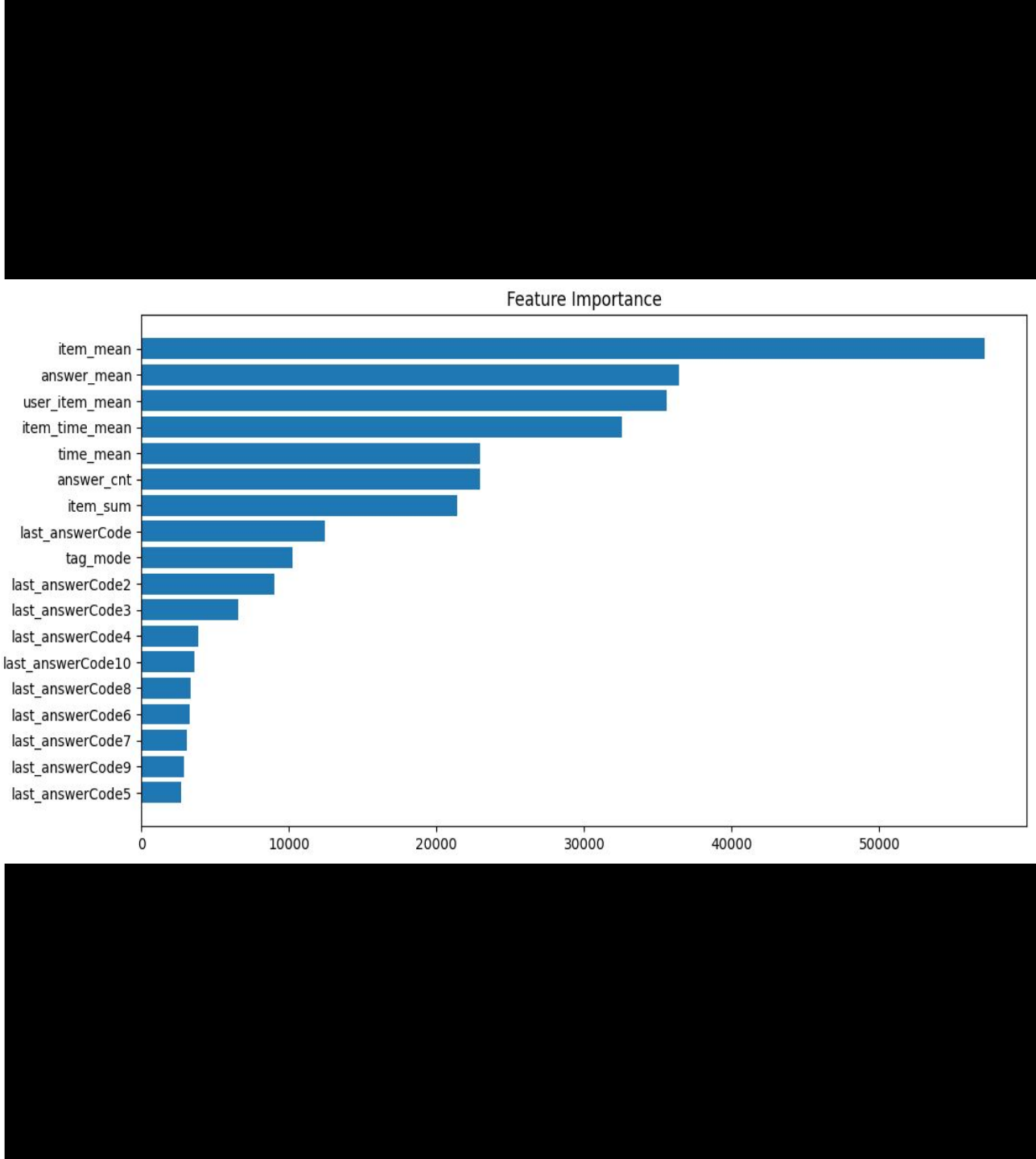
`tag_mode`: 유저에게 노출된 태그의 최빈값

`user_item_mean`: 유저가 풀어왔던 문제의 정답률

`item_mean`: 문제 기준 정답률

`item_sum`: 문제 기준 풀린 갯수

`item_time_mean`: 문제의 평균 풀이 시간



모델링 - 3. MF 모델

surprise

A Python scikit for recommender systems.

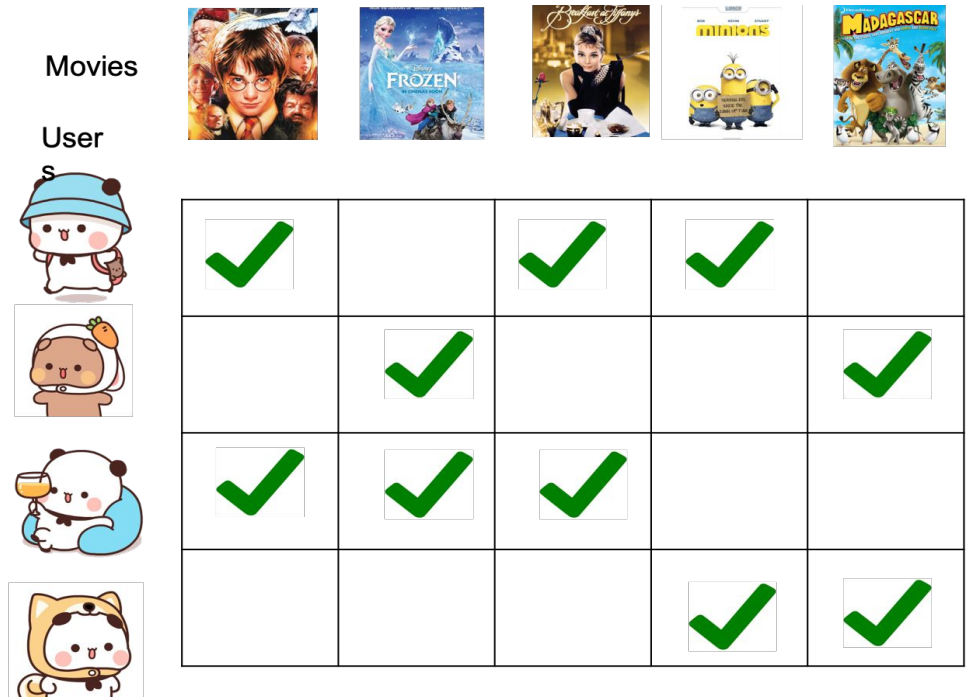
Surprise

- Scikit-learn API와 비슷한 형태로 제공하여 간편하게 추천 시스템 구현 가능한 라이브러리

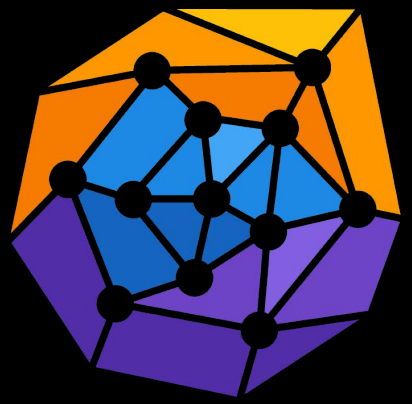
- 특이값 분해(SVD++) 알고리즘, 기존 SVD에서 추가적으로 암시적 rating이 더해진다.

- DKT에서 문제를 맞춤 (1) → 사용자가 아이템을 선호함으로 해석

- 주요 모델이 잘 예측하지 못하는 부분을 보완할 수 있을 것이라고 예측하여 추가



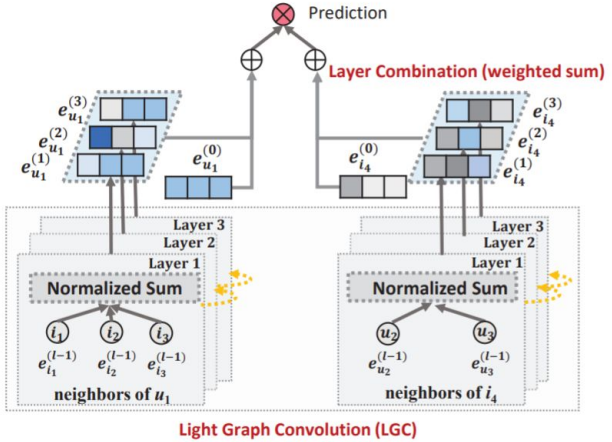
모델링 - 4. Graph



<LightGCN>

Model 설명

- 대중적인 Graph 모델
- User와 Item 간 상호작용 고려



LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

Xiangnan He
University of Science and Technology of China
xiangnanhe@gmail.com

Kuan Deng
University of Science and Technology of China
dengkuan@mail.ustc.edu.cn

Xiang Wang
National University of Singapore
xiangwang@u.nus.edu

Yan Li
Beijing Kuaishou Technology Co., Ltd.
liyan@kuaishou.com

Yongdong Zhang
University of Science and Technology of China
zhyd73@ustc.edu.cn

Meng Wang*
Hefei University of Technology
eric.mengwang@gmail.com

ABSTRACT

Graph Convolution Network (GCN) has become new state-of-the-art for collaborative filtering. Nevertheless, the reasons of its effectiveness for recommendation are not well understood. Existing work that adapts GCN to recommendation lacks thorough ablation analyses on GCN, which is originally designed for graph classification tasks and equipped with many neural network operations. However, we empirically find that the two most common designs in GCNs — feature transformation and nonlinear activation — contribute little to the performance of collaborative filtering. Even worse, including them adds to the difficulty of training and degrades recommendation performance.

In this work, we aim to simplify the design of GCN to make it more concise and appropriate for recommendation. We propose a new model named LightGCN, including only the most essential component in GCN — neighborhood aggregation — for collaborative filtering. Specifically, LightGCN learns user and item embeddings by linearly propagating them on the user-item interaction graph, and uses the weighted sum of the embeddings learned at all layers as the final embedding. Such simple, linear, and neat model is much easier to implement and train, exhibiting substantial improvements (about 16.0% relative improvement on average) over Neural Graph Collaborative Filtering (NGCF) — a state-of-the-art GCN-based recommender model — under exactly the same experimental setting. Further analyses are provided towards the rationality of the simple LightGCN from both analytical and empirical perspectives. Our implementations are available in both TensorFlow[1] and PyTorch[2].

*Meng Wang is the corresponding author.

[1]https://github.com/kuandeng/LightGCN
[2]https://github.com/gusye1234/pytorch-light-gcn

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Collaborative Filtering, Recommendation, Embedding Propagation, Graph Neural Network

ACM Reference Format:
Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401063

1 INTRODUCTION

To alleviate information overload on the web, recommender system has been widely deployed to perform personalized information filtering [7, 45, 46]. The core of recommender system is to predict whether a user will interact with an item, e.g., click, rate, purchase, among other forms of interactions. As such, collaborative filtering (CF), which focuses on exploiting the past user-item interactions to achieve the prediction, remains to be a fundamental task towards effective personalized recommendation [10, 19, 28, 39].

The most common paradigm for CF is to learn latent features (a.k.a. embedding) to represent a user and an item, and perform prediction based on the embedding vectors [6, 19]. Matrix factorization is an early such model, which directly projects the single ID of a user to her embedding [26]. Later on, several research find that augmenting user ID with the her interaction history as the input can improve the quality of embedding. For example, SVD++ [25] demonstrates the benefits of user interaction history in predicting user numerical ratings, and Neural Attentive Item Similarity (NAIS) [18] differentiates the importance of items in

Model 적용

- 베이스라인 코드 활용
- userID & assessmentItemID & answerCode 피처 활용

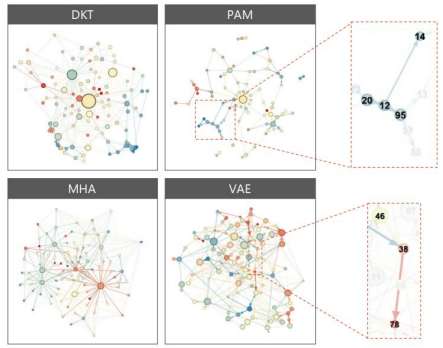
4. 그래프 모델

<GKT>

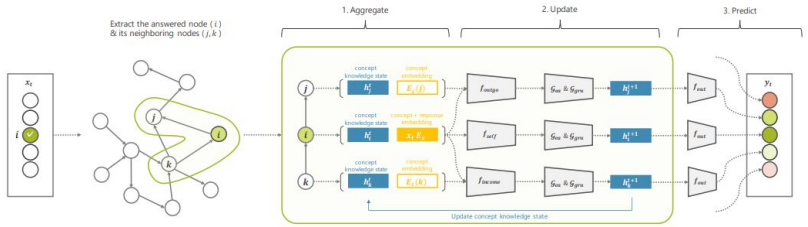
Model 설명
- Graph와 Sequential의 특성을 모두 고려한 모델
  - Graph : User와 Item 간 상호작용 고려
  - Sequential : User의 과거 풀이 기록 고려

- 다양한 종류의 Graph 구현 방식 (PAM / MHA / VAE) 적용 가능

- 그래프 집계 - 업데이트 - 예측 순서로 진행



적용 결과
- 그래프 구성에 KnowledgeTag 활용
- 최고 auc 0.7422 (private 0.7690)
- CUDA out of Memory 에러 발생
  - 그래프 부분이 메모리를 많이 차지



Published as a conference paper at ICLR 2019

GRAPH-BASED KNOWLEDGE TRACING: MODELING STUDENT PROFICIENCY USING GRAPH NEURAL NETWORK

Hiromi Nakagawa, Yusuke Iwasawa & Yutaka Matsuo
Department of Engineering, The University of Tokyo
{nakagawa,iwasawa,matsuo}@weblab.t.u-tokyo.ac.jp

ABSTRACT

We apply graph neural network (GNN) to a new area, *knowledge tracing*. Knowledge tracing predicts student performance on coursework exercises over time. From the viewpoint of data structure, coursework can be potentially structured as a graph. Incorporating such a graph-structured nature to the knowledge tracing model as a relational inductive bias can improve performance; however, previous methods, such as Deep Knowledge Tracing (DKT), do not consider such a latent graph structure. Inspired by the recent successes of GNN, we propose a GNN-based knowledge tracing method, graph-based knowledge tracing (GKT). Casting the knowledge structure as a graph, we reformulate the knowledge tracing task as a time series node-level classification problem in GNN. Since the knowledge graph structure is not explicitly given in many cases, we propose various implementations of the graph structure. Empirical validations on two open datasets showed that our method outperforms past methods in predicting student performance. Moreover, the model provides better interpretable predictions than the previous methods.

모델링 - 5.CatBoost (문제 단위 부스팅 모델)

CatBoost(문제 단위 부스팅 모델)

유저, 문제, 파생 변수 별 정답률 고려한 Rule base에 최적화된 모델

대부분 범주형 변수이며 이는 모델에서 categorical 변수를 따로 지정해서 사용

```
cat_model.fit(_train, _train_value, early_stopping_rounds=100, cat_features=cat_columns ,verbose=200, eval_set=(_valid, _valid_value))
```

Feature Engineering

`solve_time`: 유저의 문제 풀이 시간

`b_category`: assementitemID 3번째 값

`test_category`: assementitemID의 3 + 5~7번째값

`last_answerCode`: 유저가 최근 문제 정답

`elo`: 유저와 아이템 elo를 sigmoid하여 얻은 문제 난이도 값

`elouser`: 유저의 문제풀이 수준

`eloitem`: 문제의 난이도

`elotag`: tag 별 난이도

`elotest`: testid 별 난이도

5. CatBoost(문제 단위 부스팅 모델)

Why? 범주형 데이터 처리에 강한 이유

범주형(categorical) 피처 처리를 위한 알고리즘 도입

Table 2: Comparison with baselines: logloss / zero-one loss (relative increase for baselines).

	CatBoost	LightGBM	XGBoost
Adult	**0.270 / 0.127**	+2.4% / +1.9%	+2.2% / +1.0%
Amazon	**0.139 / 0.044**	+17% / +21%	+17% / +21%
Click	**0.392 / 0.156**	+1.2% / +1.2%	+1.2% / +1.2%
Epsilon	**0.265 / 0.109**	+1.5% / +4.1%	+11% / +12%
Appetency	**0.072 / 0.018**	+0.4% / +0.2%	+0.4% / +0.7%
Churn	**0.232 / 0.072**	+0.1% / +0.6%	+0.5% / +1.6%
Internet	**0.209 / 0.094**	+6.8% / +8.6%	+7.9% / +8.0%
Upselling	**0.166 / 0.049**	+0.3% / +0.1%	+0.04% / +0.3%
Kick	**0.286 / 0.095**	+3.5% / +4.4%	+3.2% / +4.1%

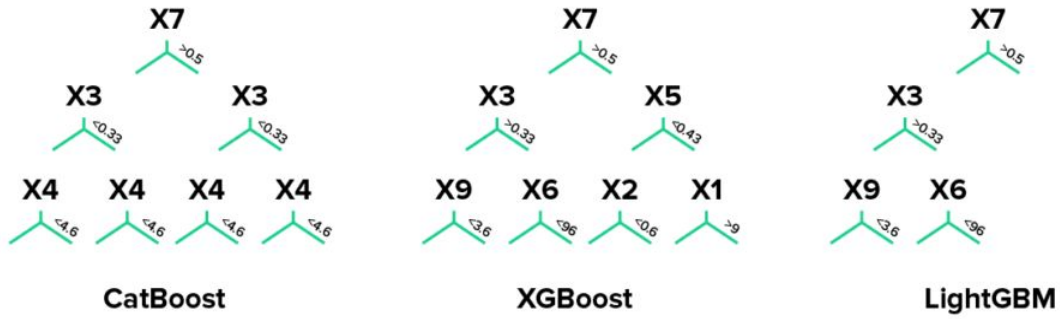
CatBoost: unbiased boosting with categorical features

Liudmila Prokhorenkova[1,2], **Gleb Gusev**[1,2], **Aleksandr Vorobev**[1], **Anna Veronika Dorogush**[1], **Andrey Gulin**[1]
[1]Yandex, Moscow, Russia
[2]Moscow Institute of Physics and Technology, Dolgoprudny, Russia
{ostroumova-la, gleb57, alvor88, annaveronika, gulin}@yandex-team.ru

Abstract

This paper presents the key algorithmic techniques behind CatBoost, a new gradient boosting toolkit. Their combination leads to CatBoost outperforming other publicly available boosting implementations in terms of quality on a variety of datasets. Two critical algorithmic advances introduced in CatBoost are the implementation of *ordered boosting*, a permutation-driven alternative to the classic algorithm, and an innovative algorithm for processing categorical features. Both techniques were created to fight a *prediction shift* caused by a special kind of target leakage present in all currently existing implementations of gradient boosting algorithms. In this paper, we provide a detailed analysis of this problem and demonstrate that proposed algorithms solve it effectively, leading to excellent empirical results.

Tree growth examples:



모델 고도화

Hyper parameter Tuning

Optuna

Optuna 함수 정의

모델	valid score	public score	private score
튜닝전	0.8422 0.7632	0.8289 0.7581	0.8564 0.7742
튜닝후	0.8455 0.7669	0.8350 0.7554	0.8542 0.7634

- 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 진행하였음
- max_depth는 숫자가 커지면 학습시간이 길어져서 성능이 좋았던 11로 고정하였음
- hyper parameter tuning을 할때는 iterations을 500으로 지정해주었고 제출할때에는 4000으로 지정하였음

모델 고도화

Hyper parameter Tuning

Optuna

Verifying the best parameters from Optuna

- Optuna 에서 찾은 최적의 하이퍼 파라미터 성능을 검증하고자 함
- train set, validation set 나눈 것 이용

model	train score	valid score	public score	private score
CatBoost	0.8934	0.8455	0.8542	0.8350
LightGCN	0.8985	0.8014	0.7869	0.8120
LGBM	0.8219	0.8010	0.7869	0.8236

모델 고도화

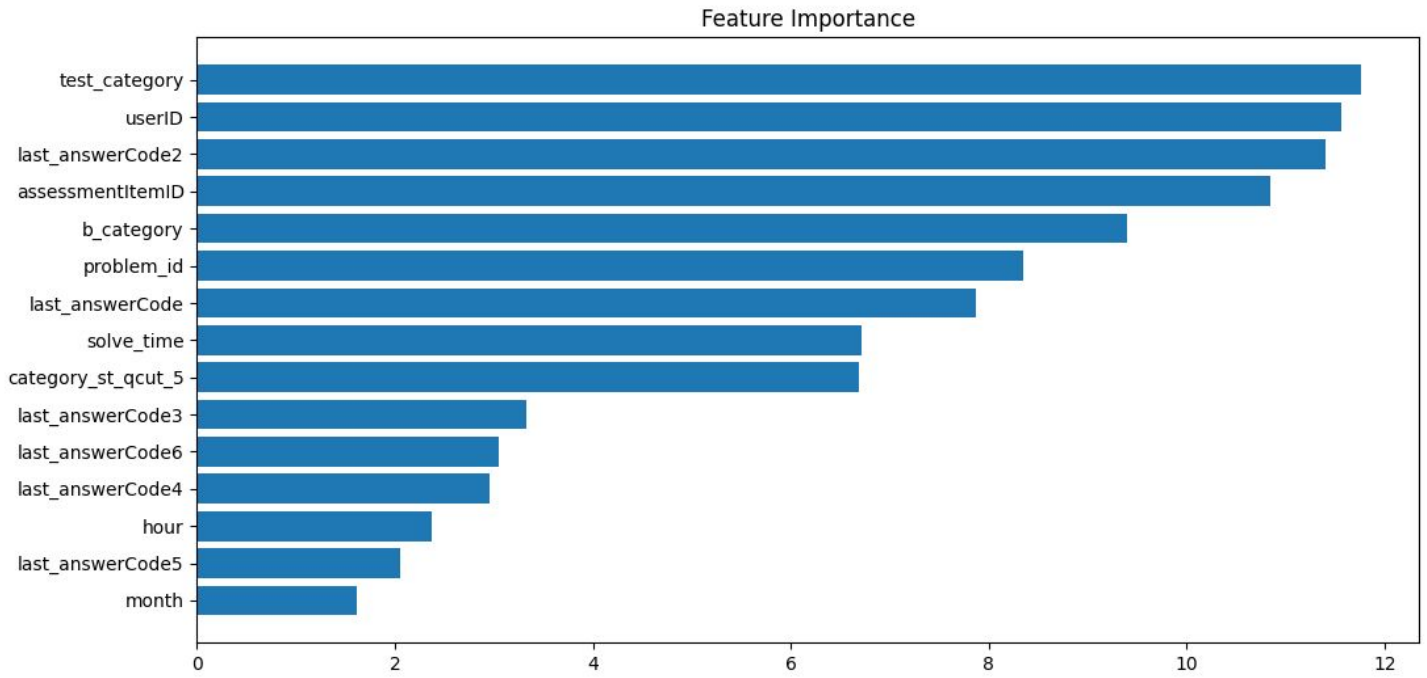
Hyper parameter Tuning

Optuna & Feature Importance

OPTUNA

Feature Importance

- CatBoost 모델의 변수 중요도 시각화 →



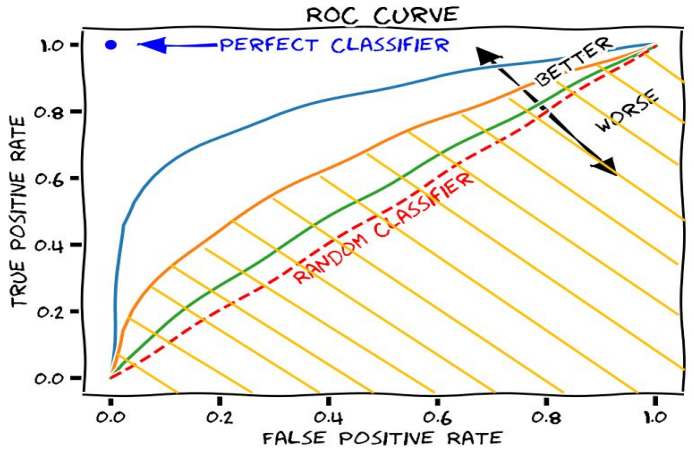
- 변수 중요도 순서

1. test_category
2. userID
3. last_answerCode2
4. assessmentItemID

…

모델 고도화

Ensemble



- AUC 지표는 threshold(기준점)을 바꿔가며 TPR과 FPR을 기록하여 면적을 구하는 방식
  → 즉 맞출 확률이 높은 순으로 pred 값의 크기 순위를 잘 매기는 것이 중요

model	public score	private score
catboost + MF	0.8362	0.8549
catboost + LGBM	0.8353	0.8530
catboost + MF + LGBM	0.8358	0.8543

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

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

Thank You