DKT

- [ ] AUC 블로그정리
- [ ] DKT 논문 정리
- [ ] Riid 상위 모델 해보기
- [ ] 판다스 groupby

[Pandas 기초] 그룹(group)객체 생성 및 집계(agg) 연산

groupby()함수는 그룹객체를 만들어주는 함수로 Dataframe객체.groupby(기준이 되는 열이름) 로 사용된다. 이번에도 타이타닉 데이터를 불러와 특정열만 가져와보자. class열에는 first, second, third라는 3개의 값들이 들어 있다. 이 열을 기준으로 그룹객체를 생성해보자. 그룹객체는 반복문을 이용할 수 있다. 총 그룹이 3개이므로, 3개의 튜플형태를 반환한 것을 알 수 있다. 튜플형태로

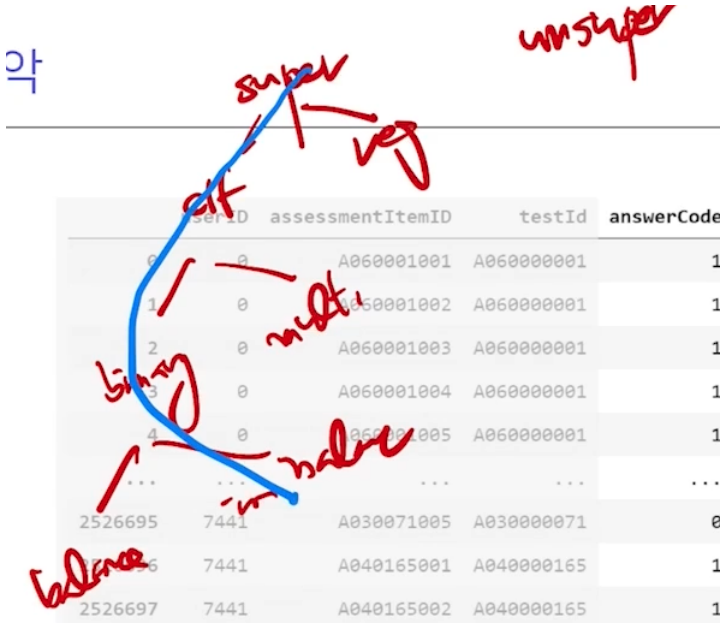
https://yganalyst.github.io/data_handling/Pd_13/

Riiid: Comprehensive EDA + Baseline

Explore and run machine learning code with Kaggle Notebooks | Using data from multiple data sources

https://www.kaggle.com/code/erikbruin/riiid-comprehensive-eda-baseline/notebook#2.-Baseline-model

- [ ] 7442 유저 9454 아이템아이디 1537개의 시험지아이디 912개의 문제종류knowledge tag
- [ ] 전체 interaction에 대해 65.45 퍼센트가 출력이 1인 데이터



assessmentItemID

사용자가 푼 문항의 일련 번호로, **총 10자리**로 구성

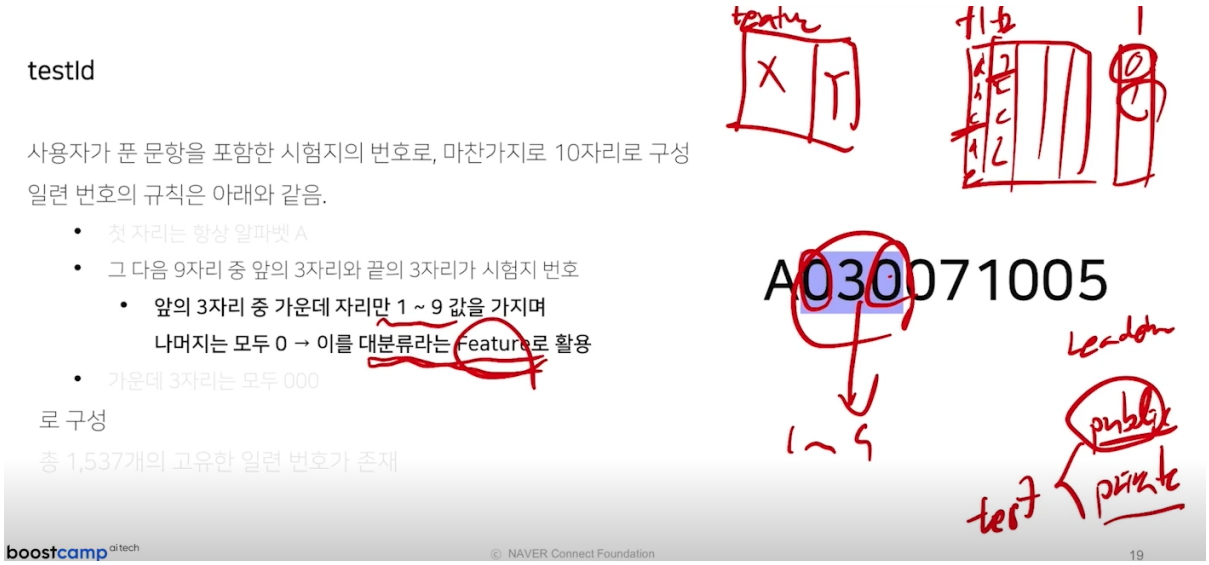
일련 번호의 규칙은 아래와 같음.

- 첫 자리는 항상 알파벳 A
- 그 다음 6자리는 시험지 번호
- 마지막 3자리는 시험지 내 문항의 번호

로 구성

총 9,454개의 고유한 일련 번호가 존재

~~A~~030071005



☐ public/private dataset score 와 cv score 가 동시에 조금이라도 상승하면 의미있는 feature 라는 것

☐ feature engineering 을 신경써보자

기술 통계량이란?

- 일반적으로 데이터를 살펴볼 때, 가장 먼저 살펴보는 것은 **기술 통계량**입니다.
- 보통 데이터 자체의 정보를 수치로 요약, 단순화하는 것을 목적으로 하며
- 우리가 잘 알고 있는 **평균, 중앙값, 최대/최소**와 같은 값들을 뽑아내고,
  EDA 과정에서는 이들을 유의미하게 **시각화**하는 작업을 거칩니다.
- 분석은 최종 목표인 **정답률과 연관 지어 진행**하는 것이 유리합니다.

1.2 기술 통계량 분석

사용자 분석

- 한 사용자가 몇 개의 문항을 풀었는지
  (보통 groupby 명령어를 통해 찾아낼 수 있습니다.)
  평균 339 문항, 최소 9문항, 최대 1,860문항
- 학생 별로 정답률이 어떻게 되는지
  평균 62.8%, 최소 0.0%, 최대 100.0%, 중앙값 65.1%

특성 별 빈도 분석 종합

특성	고유값	평균 문항수	최소값	1분위수	2분위수 (중앙값)	3분위수	최대값
학생 userID	7,442명	339문항	9문항	78문항	232문항	519문항	1,860문항
문항 assessmentItemID	9,454개	267회	50회	250회	250회	300회	500회
시험지 testId	1,537개	1,643회	200회	1,500회	1,500회	1,800회	4,400회
태그 KnowledgeTag	912개	2770회	50회	1,137회	2,500회	4,500회	14,350회

☐ 최근 몇개의 문항에 대한 정답률을 feature에 활용할것인가?

그 밖의 생각해볼 수 있는 것들

- 더 많이 노출된 시험지는 정답률이 높을까?
- 같은 시험지의 내용이나 같은 태그의 내용을 연달아 풀면, 정답률이 오를까?
  - 비슷한 개념의 문항을 연달아 풀면 성취도가 올라가는 현상
- 정답을 특별히 잘 맞추는 시간대가 있을까?

Riiid Techblog - Medium

Read writing from Riiid Techblog on Medium. 뤼이드 테크 블로그는 교육 현장에서 실제 학습 효과를 입증하고 그 영향력을 확대하고 있는 뤼이드의 AI 기술 연구, 엔지니어링, 이를 가장 효율적으로 비즈니스화 하는 AIOps 및 개발 문화 등에 대한 실질적인 이야기를 나눕니다.

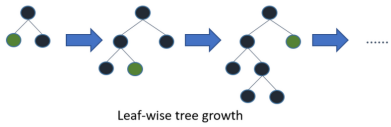
https://medium.com/@riiidtechblog

- [ ] RFE/RFECV
- [ ] tree feature importance
- [ ] kaggle 대회 순서도 블로그 포스팅
- [ ] voting 평균 루트 제곱 파워 앙상블 ranking auc seed change

[LightGBM] LGBM는 어떻게 사용할까? (설치,파라미터튜닝)

Light GBM은 트리 기반의 학습 알고리즘인 gradient boosting 방식의 프레임 워크이다. Light GBM은 나무를 수직으로 확장한다. 반면 다른 알고리즘은 나무를 수평으로 확장한다. 따라서 기존의 알고리즘은 수평으로 확장하여 포화 트리를 만드는 방향으로 학습하는 반면 leaf-wise tree growth인 LGBM은 최대 delta loss가 증가하도록 잎의 개수를 정한다. leaf-wise알고리즘은 다른 level-

https://greatjoy.tistory.com/72



The Illustrated Transformer

저번 글에서 다뤘던 attention seq2seq 모델에 이어, attention 을 활용한 또 다른 모델인 Transformer 모델에 대해 얘기해보려 합니다. 2017 NIPS에서 Google이 소개했던 Transformer는 NLP 학계에서 정말 큰 주목을 끌었는데요, 어떻게 보면 기존의 CNN 과 RNN 이 주를 이뤘던 연구들에서 벗어나 아예 새로운 모델을 제안했기 때문이지 않을까 싶습니다.

https://nlpinkorean.github.io/illustrated-transformer/

LightGBM Classifier in Python

Explore and run machine learning code with Kaggle Notebooks | Using data from Breast Cancer Prediction Dataset

https://www.kaggle.com/code/prashant111/lightgbm-classifier-in-python/notebook

시간이 비정상적으로 짧으면 찍은거아닌가?

있는 feature 부터 정리하고

새로운 feature 삽입

▼ LGBM Feature Engineering

	train_loss	val_loss	val_auc	acc	leaderboard
original	0.558699	0.680181	0.690458159	0.609865470	0.7345
A)orig-test_sum	0.559685	0.67378	0.709775059	0.605906313	0.7380
B)orig-tagsum	0.55254	0.676997	0.69899620	0.613856342	0.7374
orig-usertotal	0.551845	0.661024	0.720938679	0.614243323	0.7270
orig-usercorect	0.552205	0.672252	0.707737824	0.612852664	0.7205
A-usertotalans	0.553732	0.690532	0.681860965	0.592481203	
A') A-tagsum[Best1]	0.555026	0.690478	0.696325110	0.587023686	0.7454
A-usercorrect	0.552524	0.686687	0.689221609	0.597174866	0.7310
B-usertotalans	0.551974	0.677551	0.705983157	0.606454720	
B-tagmean	0.564091	0.69903	0.688826085	0.582484725	
orig-usertotal-usercorec	0.55578	0.672129	0.7047388113	0.614410905	
A'-usertotal	0.553419	0.698642	0.686733098	0.592805755	0.7390
A'-tagmean	0.564419	0.683215	0.703756374	0.597409068	0.7405
A'+time(int)	0.554581	0.67988	0.686771019	0.607374190	0.7410/0.6667
A'+bigclass	0.559728	0.681469	0.689352514	0.605381165	0.7427
A'+Timeclass	0.554156	0.671642	0.685525243	0.6123567513	0.7407
A'+Time+timeclass	0.554716	0.672936	0.6843500298151461	0.6108619830592925	0.7390
orig+time(int)	0.559417	0.671644	0.710533589	0.600305498	
orig+time(log)	0.555324	0.678428	0.687759888	0.60886895	

제목 없음

제목 없음

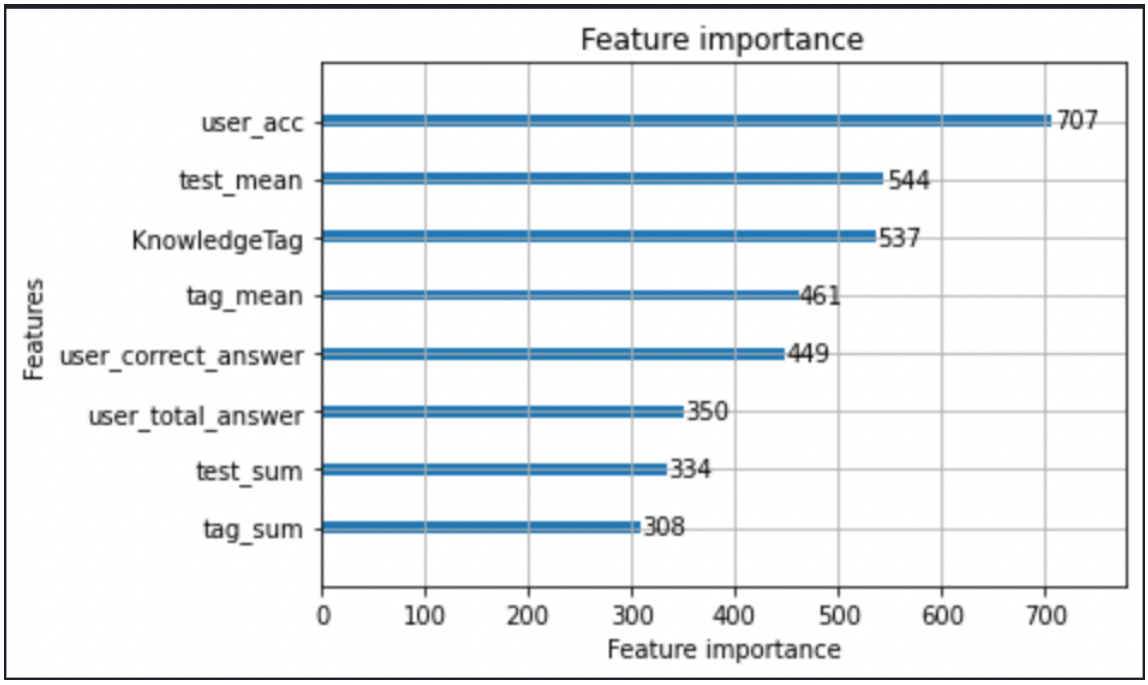
제목 없음

제목 없음

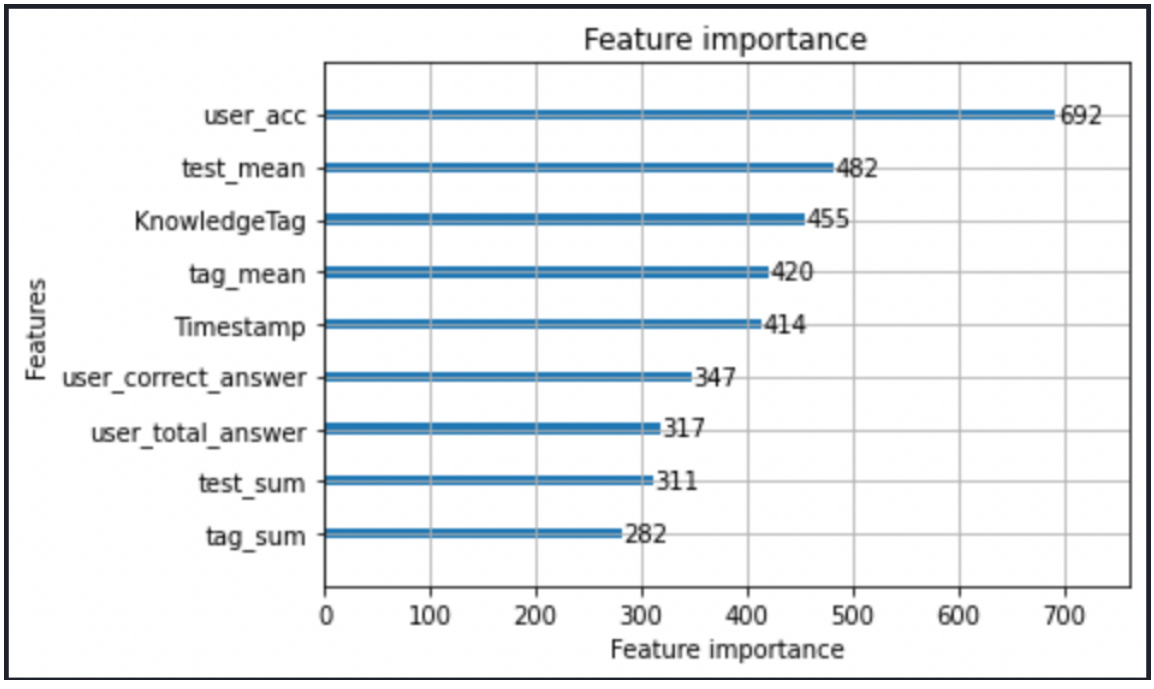
제목 없음

제목 없음

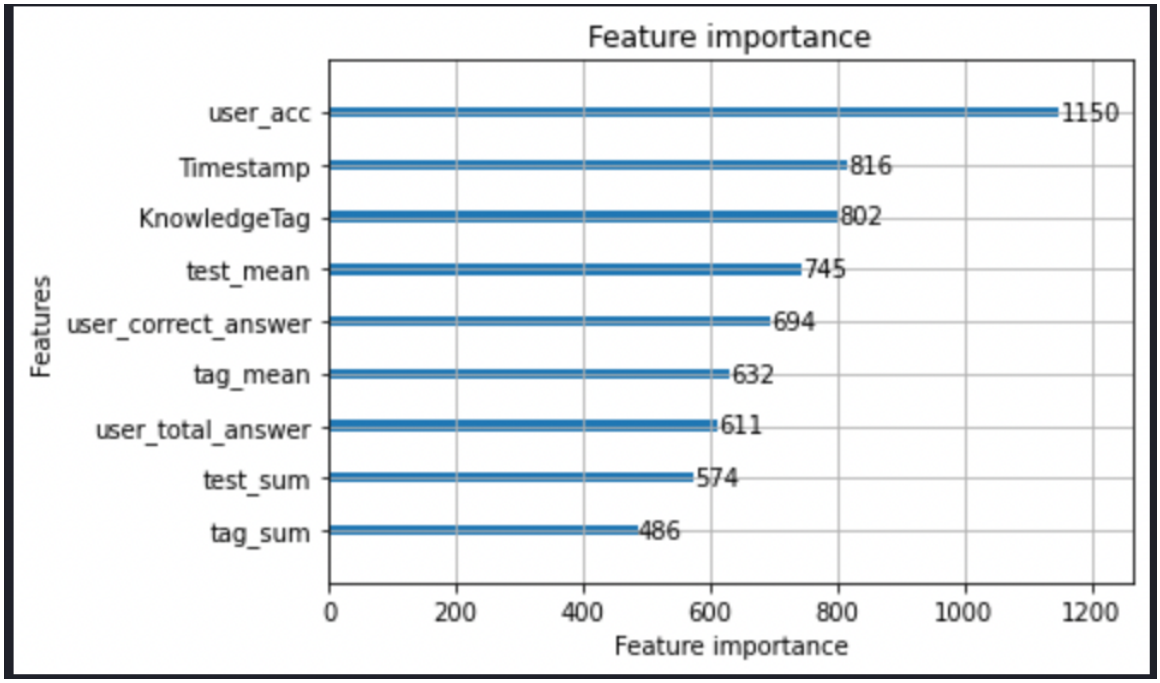
original:



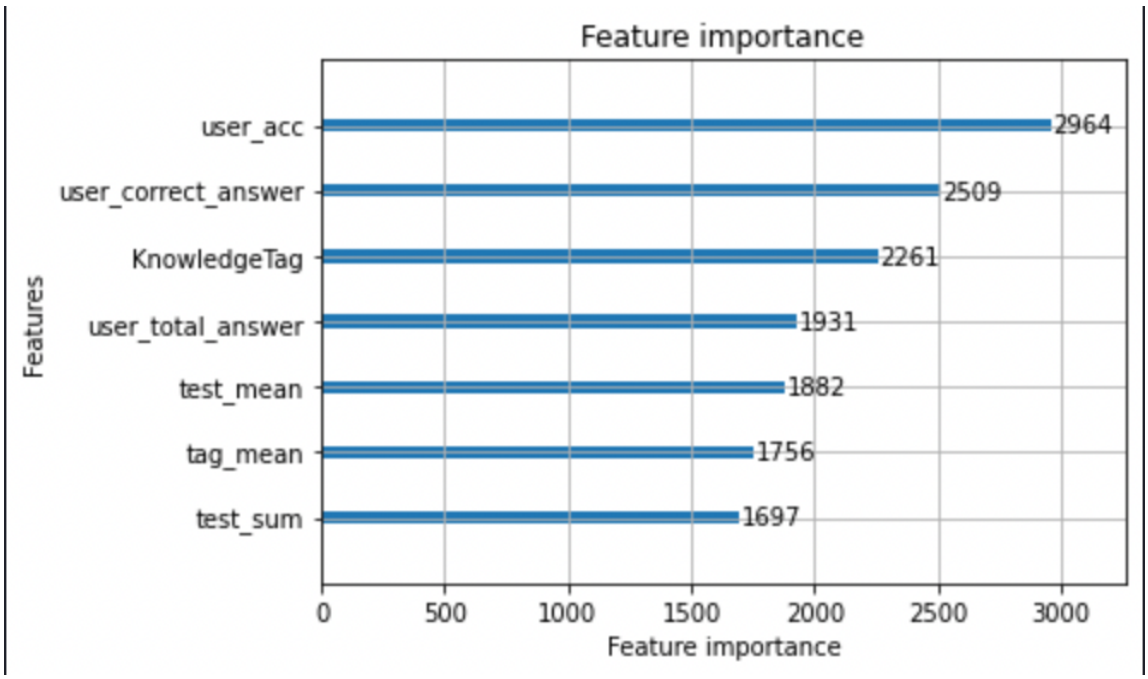
orig+timestamp(int):



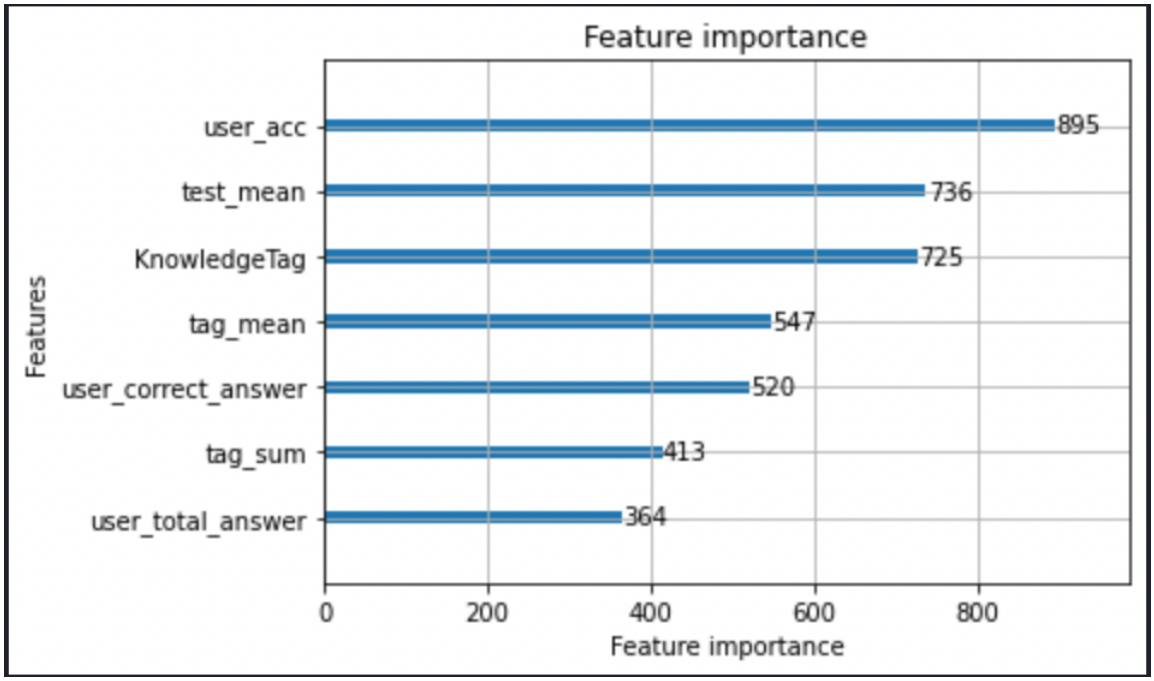
orig+timestamp(log)



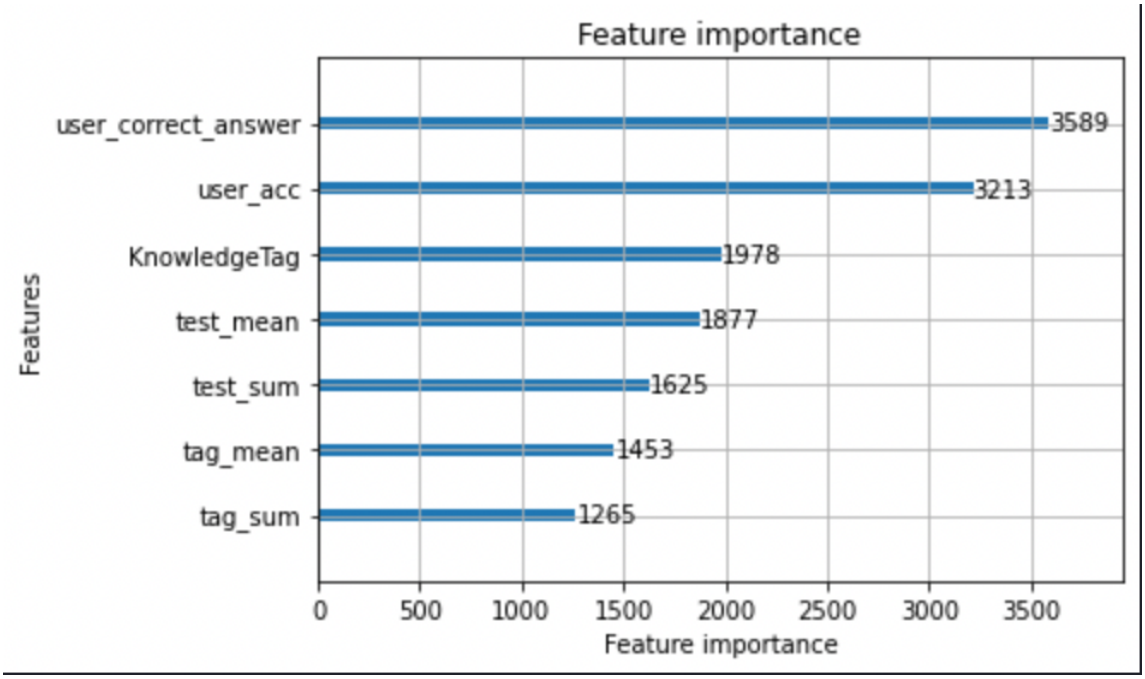
orig-tagsum



orig-testsum

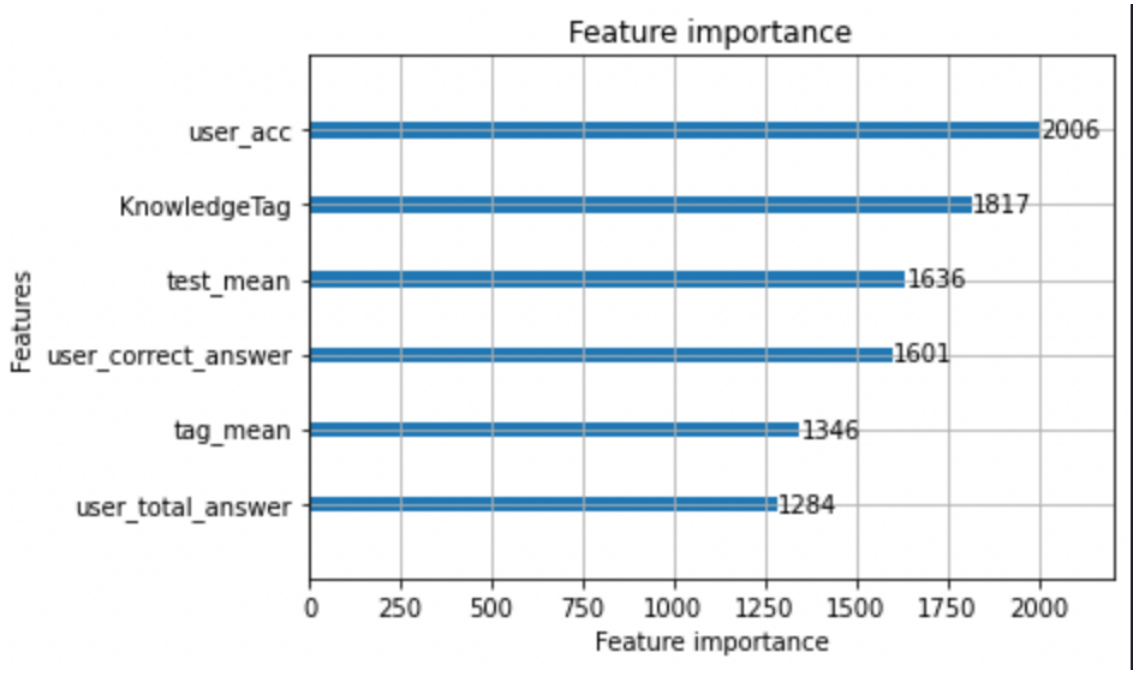


orig-usertotalanswer

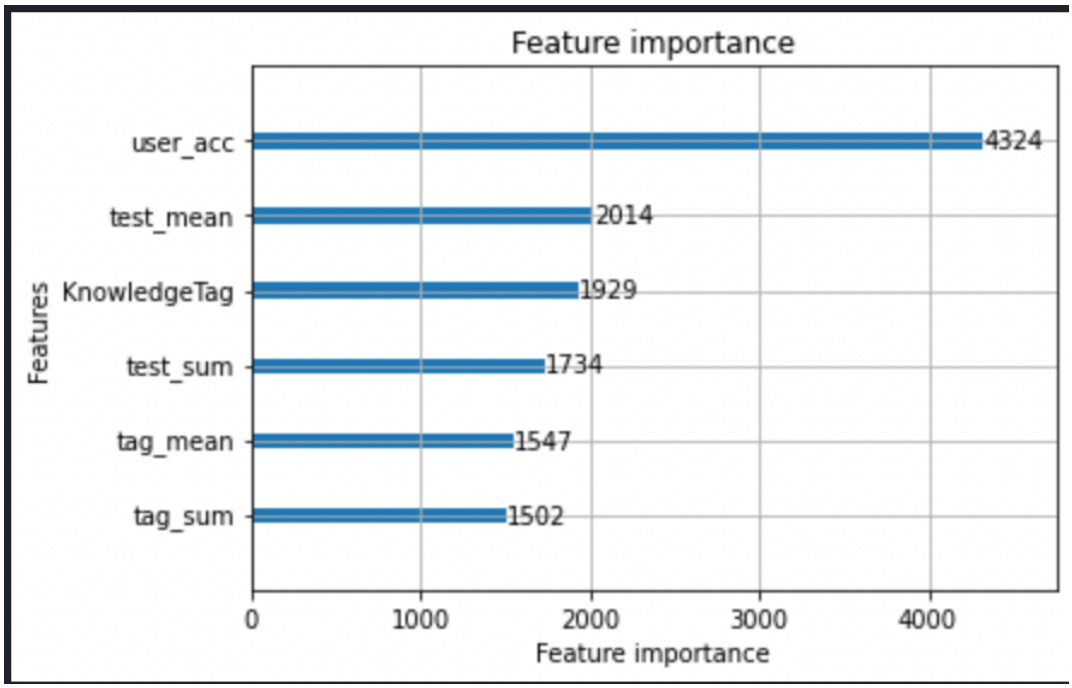


orig-testsum-usertotalanswer

orig-tagsum



orig-usertotal-usercorrect



제목 없음

제목 없음

제목 없음