boostcamp ai tech

Book Rating Prediction

사용자의 책 평점 데이터를 바탕으로

Recsys 06 따봉도치 조 👍

김성연 김찬호 박문순 배성수 이지훈

목차

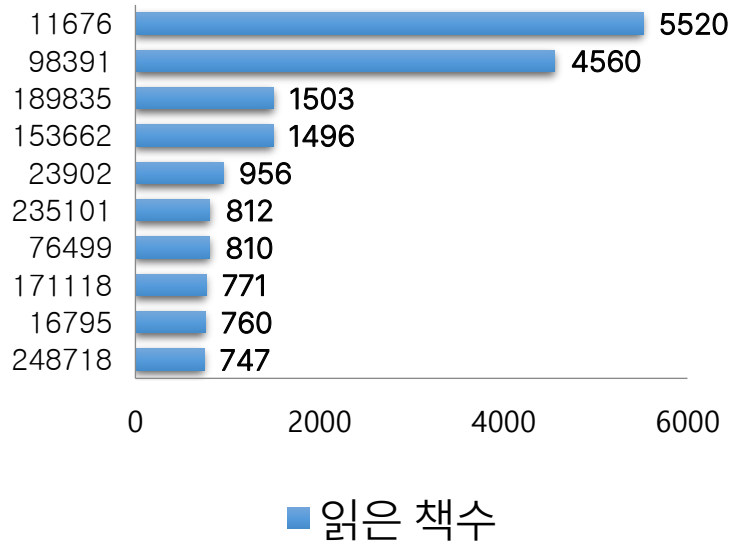
- EDA
  - Train, Test Data Set의 Top 10 독서 유저 리스트
  - 유저와 아이템(Books)의 개수와 Train Data Set의 Rating 개수(희소행렬)
- CatBoost 모델
  - 베이스라인 Code의 DL Model 실험
  - 평점의 평균 값을 예측에 사용
  - 데이터의 특성
  - 간략한 데이터 전처리
- 딥러닝 모델
  - 앙상블 효과를 높이기 위한 방법
  - FFM+DCN 모델
- 모델 고도화
  - wandb + sweep
  - Optuna
  - K-Fold
  - Ensemble
- Q & A

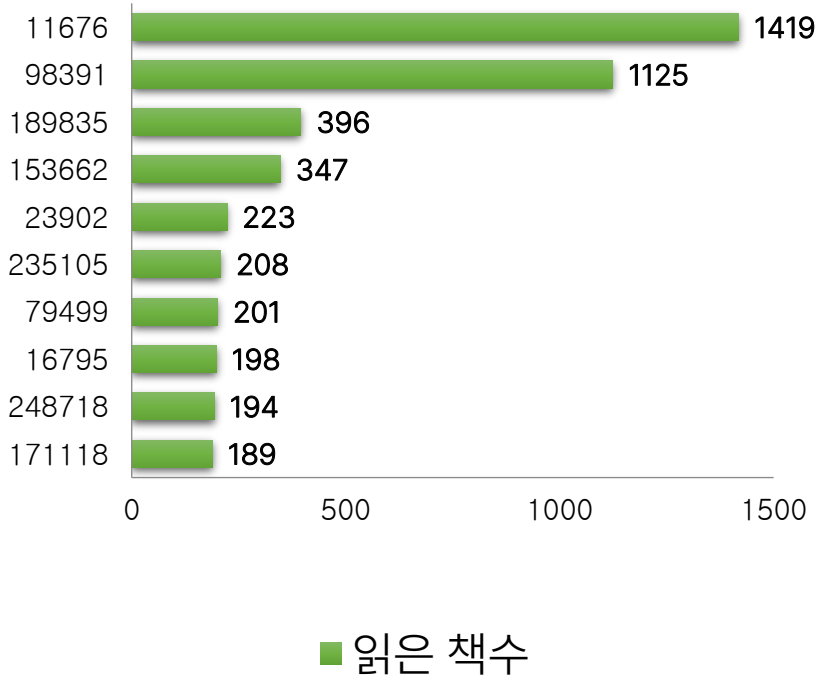
EDA

Top 10 User

Train Data Set의 Top 10

- Train Data Set에서 책을 많이 읽은 **Top 10** user





Test Data Set의 Top 10

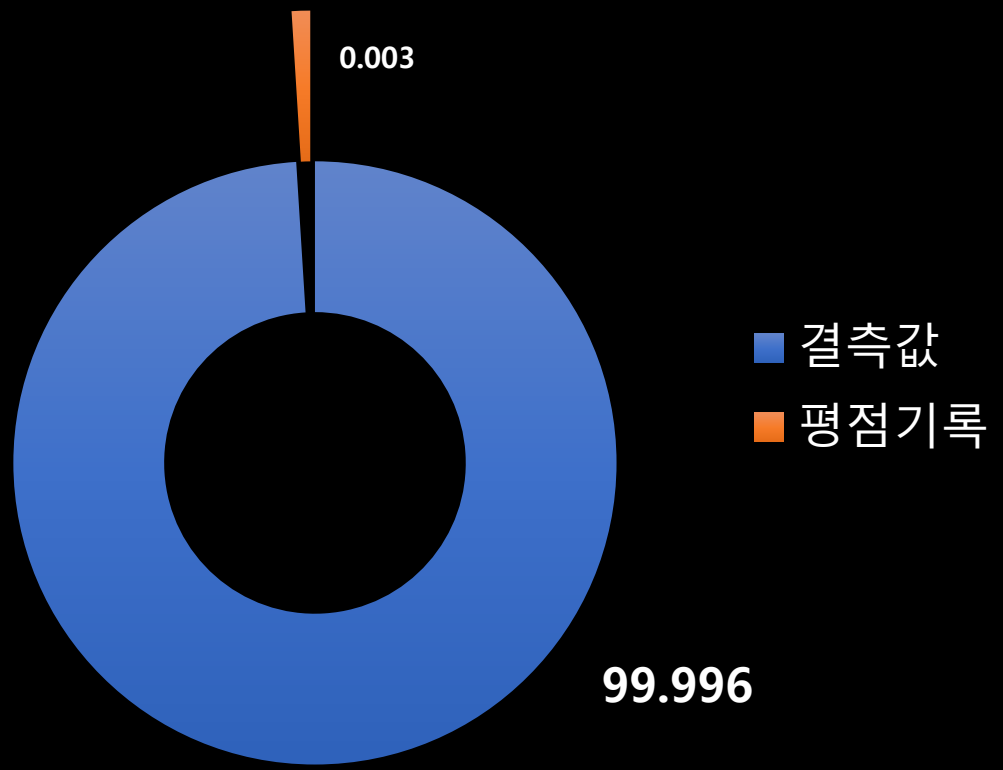
- Test Data Set에서 책을 많이 읽은 **Top 10** user

EDA

희소행렬

유저 수 : **68069**, 책 수 : **149570**
최대 평점 기록 개수 : 유저 수 * 책 수
=68069 * 149570 = **10181080330**.

평점 기록 : 306795
희소 행렬(결측값 비율) =
(1- 유저 수 * 책 수) / 평점기록 * 100 % =
99.99699 %



EDA

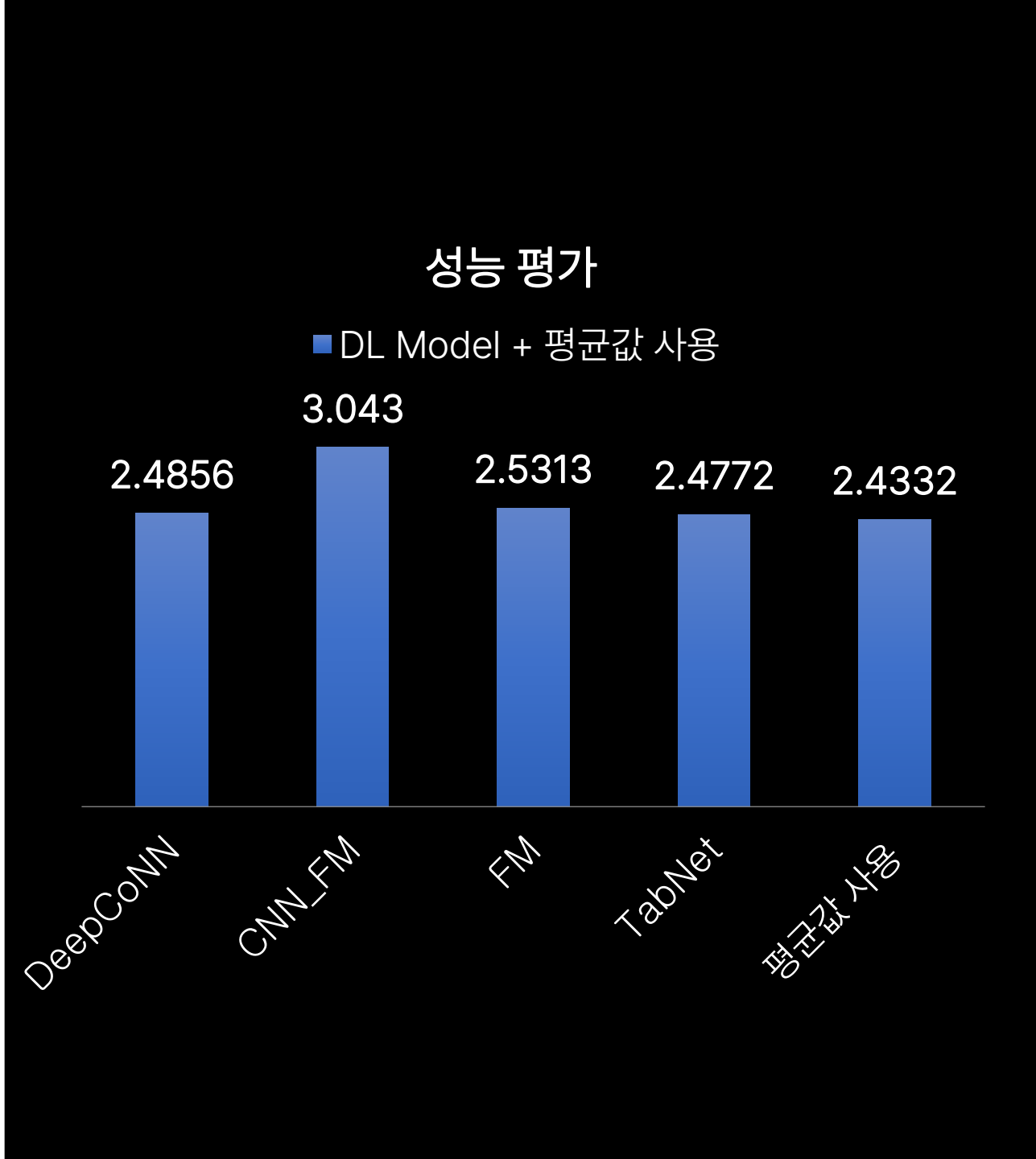
BassLine DL Model 실험

DeepCoNN : 베이스라인과 유저 summary 정보를 여러 형태로 변환해서 사용했습니다.

TabNet : 사이킷런 패키지를 이용했습니다.

CNN_FM, FM : 베이스라인 코드 입니다.

평균값 사용 : Rating 평균 값 7을 예측값으로 사용했을 때 RMSE 지표입니다.



모델 접근법 - CatBoost

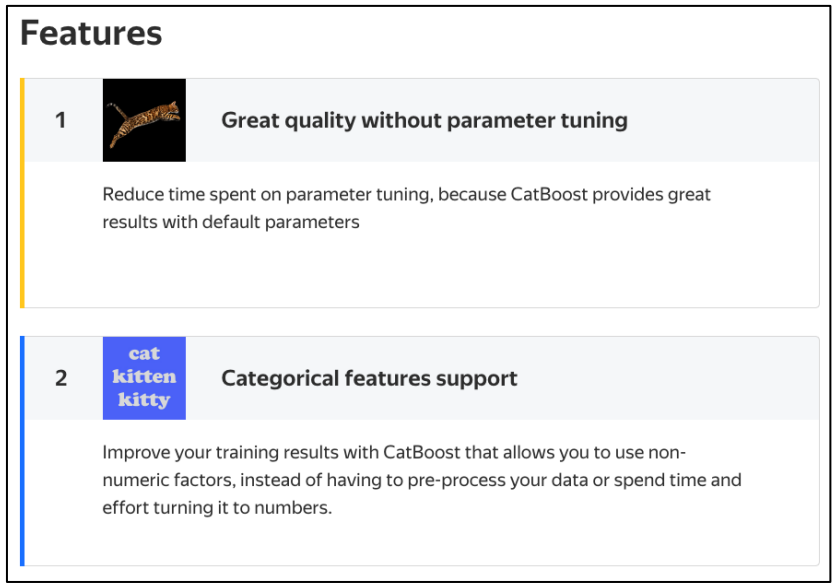
데이터 특성

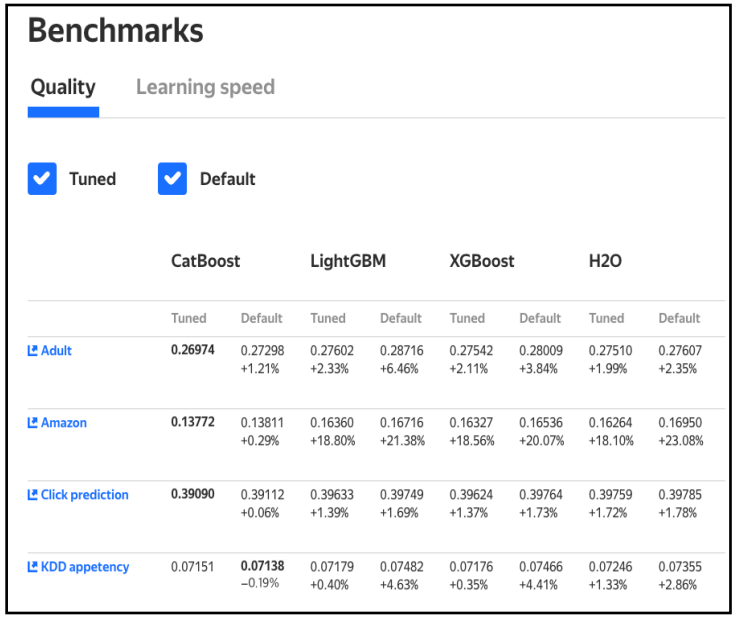
```
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   isbn                 149570 non-null  object
 1   book_title           149570 non-null  object
 2   book_author          149570 non-null  object
 3   year_of_publication  149570 non-null  float64
 4   publisher            149570 non-null  object
 5   img_url              149570 non-null  object
 6   language             82343 non-null   object
 7   category             80719 non-null   object
 8   summary              82343 non-null   object
 9   img_path             149570 non-null  object
dtypes: float64(1), object(9)
```

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   user_id   68092 non-null  int64
 1   location  68092 non-null  object
 2   age       40259 non-null  float64
dtypes: float64(1), int64(1), object(1)
```

범주형 변수

- 수치형 변수인 year_of publication, age 변수를 제외한 모든 변수는 **범주형 변수**





CatBoost 모델 선택

- 정형 데이터를 다루는 분류 및 회귀 문제에서 **트리 기반 부스팅 모델**이 **가장 좋은 성능**을 보여주고 있음 (in Kaggle...)
- 트리 기반 부스팅 모델 중 **범주형 변수 처리에 특화**된 CatBoost 선택

모델 접근법 - CatBoost

데이터 전처리

```
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   isbn                 149570 non-null  object
 1   book_title           149570 non-null  object
 2   book_author          149570 non-null  object
 3   year_of_publication  149570 non-null  float64
 4   publisher            149570 non-null  object
 5   img_url              149570 non-null  object
 6   language             82343 non-null   object
 7   category             80719 non-null   object
 8   summary              82343 non-null   object
 9   img_path             149570 non-null  object
dtypes: float64(1), object(9)
```

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   user_id   68092 non-null  int64
 1   location  68092 non-null  object
 2   age       40259 non-null  float64
dtypes: float64(1), int64(1), object(1)
```

Users – Age

- 나이 그룹 별 평점 평균에 유의미한 차이 존재
- NULL 값을 평균값으로 대체하는 대신, 범주형으로 군집화

```
0 ~ 10 까지 출판된 책 623 개의 평균 평점은 6.7335 입니다.
10 ~ 20 까지 출판된 책 12034 개의 평균 평점은 7.1113 입니다.
20 ~ 30 까지 출판된 책 53141 개의 평균 평점은 7.2486 입니다.
30 ~ 40 까지 출판된 책 59784 개의 평균 평점은 7.0632 입니다.
40 ~ 50 까지 출판된 책 38489 개의 평균 평점은 7.2613 입니다.
50 ~ 100 까지 출판된 책 33299 개의 평균 평점은 7.3867 입니다.
NULL 나이 책은 92662개로 평균 평점은 6.7354 입니다.
```

Books – Language

- 언어 그룹 별 평점 평균에 유의미한 차이 존재

```
'en'로 작성된 책 182282개의 평점 평균은 7.089 입니다.
'de'로 작성된 책 2226개의 평점 평균은 6.663 입니다.
'es'로 작성된 책 1486개의 평점 평균은 6.918 입니다.
'fr'로 작성된 책 1175의 평점 평균은 7.12 입니다.
'it'로 작성된 책 296개의 평점 평균은 7.399 입니다.
책 작성 언어가 NULL인 책 119084개의 평점 평균은 7.049 입니다.
```

모델 접근법 - CatBoost

데이터 전처리

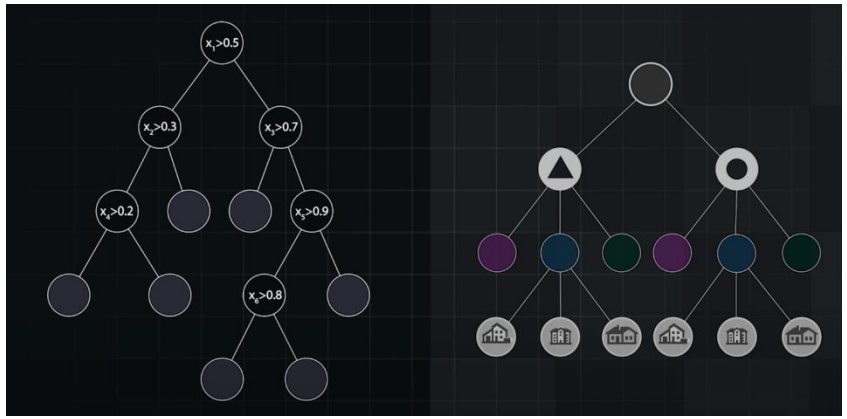
```
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   isbn                 149570 non-null  object
 1   book_title           149570 non-null  object
 2   book_author          149570 non-null  object
 3   year_of_publication  149570 non-null  float64
 4   publisher            149570 non-null  object
 5   img_url              149570 non-null  object
 6   language             82343 non-null   object
 7   category             80719 non-null   object
 8   summary              82343 non-null   object
 9   img_path             149570 non-null  object
dtypes: float64(1), object(9)
```

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   user_id   68092 non-null  int64
 1   location  68092 non-null  object
 2   age       40259 non-null  float64
dtypes: float64(1), int64(1), object(1)
```

Books – year_of_publication

- 책 출간 연도 그룹 별 평점 평균에 유의미한 차이 존재
- 책 출간 연도와 rating간 상관계수 -0.004 고려해 범주형으로 군집화

```
1900 ~ 1970 까지 출판된 책 2469 개의 평균 평점은 7.5055 입니다.
1970 ~ 1980 까지 출판된 책 7208 개의 평균 평점은 7.2278 입니다.
1980 ~ 1990 까지 출판된 책 36409 개의 평균 평점은 7.0839 입니다.
1990 ~ 2000 까지 출판된 책 140308 개의 평균 평점은 7.0004 입니다.
2000 ~ 2020 까지 출판된 책 87570 개의 평균 평점은 7.1678 입니다.
```



CatBoost 모델에 적용

- CatBoost 모델이 범주형 변수에 강하기 때문에, 수치형 변수인 year_of publication, age 변수를 범주형으로 군집화
- 이러한 간단한 전처리 기반으로 별 다른 튜닝 없이 CatBoost 모델 Test RMSE 2.14 달성!

모델 접근법 - 딥러닝

TABULAR DATA:
DEEP LEARNING IS NOT ALL YOU NEED

- 정형 데이터 분류 및 회귀 문제에서 트리 기반 모델이 강세
- 딥러닝을 정형 데이터에 활용하기 위해 제안된 모델을 분석

- 트리 기반 부스팅 모델 XGBoost의 단일 모델 성능이 가장 좋음
- 딥러닝 기반 모델은 모델이 제안된 논문 데이터셋에서만 좋은 성능을 기록

- 저자는 딥러닝 모델의 성능이 데이터 셋에 따라 크게 변하는 이유로 2가지 가능성을 제시
  - 선택 편향 (모델에 잘 맞는 데이터에 대해 시연을 진행)
  - 하이퍼 파라미터 최적화의 차이

TABULAR DATA: DEEP LEARNING IS NOT ALL YOU NEED

Ravid Shwartz-Ziv
ravid.ziv@intel.com
IT AI Group, Intel

Amitai Armon
amitai.armon@intel.com
IT AI Group, Intel

Model Name	Rossman	CoverType	Higgs	Gas	Eye	Gesture
XGBoost	490.18 ± 1.19	3.13 ± 0.09	21.62 ± 0.33	2.18 ± 0.20	**56.07**±0.65	80.64 ± 0.80
NODE	488.59 ± 1.24	4.15 ± 0.13	21.19 ± 0.69	2.17 ± 0.18	68.35 ± 0.66	92.12 ± 0.82
DNF-Net	503.83 ± 1.41	3.96 ± 0.11	23.68 ± 0.83	**1.44** ±0.09	68.38 ± 0.65	86.98 ± 0.74
TabNet	**485.12**±1.93	3.01 ± 0.08	**21.14**±0.20	1.92 ± 0.14	67.13 ± 0.69	96.42 ± 0.87
1D-CNN	493.81 ± 2.23	3.51 ± 0.13	22.33 ± 0.73	1.79 ± 0.19	67.9 ± 0.64	97.89 ± 0.82
Simple Ensemble	488.57 ± 2.14	3.19 ± 0.18	22.46 ± 0.38	2.36 ± 0.13	58.72 ± 0.67	89.45 ± 0.89
Deep Ensemble w/o XGBoost	489.94 ± 2.09	3.52 ± 0.10	22.41 ± 0.54	1.98 ± 0.13	69.28 ± 0.62	93.50 ± 0.75
Deep Ensemble w XGBoost	485.33 ± 1.29	**2.99** ±0.08	22.34 ± 0.81	1.69 ± 0.10	59.43 ± 0.60	**78.93** ±0.73

TabNet DNF-Net

Model Name	YearPrediction	MSLR	Epsilon	Shrutime	Blastchar
XGBoost	77.98 ± 0.11	55.43±2e-2	11.12±3e-2	13.82 ± 0.19	20.39 ± 0.21
NODE	76.39 ± 0.13	55.72±3e-2	**10.39**±1e-2	14.61 ± 0.10	21.40 ± 0.25
DNF-Net	81.21 ± 0.18	56.83±3e-2	12.23±4e-2	16.8 ± 0.09	27.91 ± 0.17
TabNet	83.19 ± 0.19	56.04±1e-2	11.92±3e-2	14.94±, 0.13	23.72 ± 0.19
1D-CNN	78.94 ± 0.14	55.97±4e-2	11.08±6e-2	15.31 ± 0.16	24.68 ± 0.22
Simple Ensemble	78.01 ± 0.17	55.46±4e-2	11.07±4e-2	13.61±, 0.14	21.18 ± 0.17
Deep Ensemble w/o XGBoost	78.99 ± 0.11	55.59±3e-2	10.95±1e-2	14.69 ± 0.11	24.25 ± 0.22
Deep Ensemble w XGBoost	**76.19** ±0.21	**55.38**±1e-2	11.18±1e-2	**13.10**±0.15	**20.18**±0.16

NODE New datasets

Name	Average Relative Performance (%)
XGBoost	3.34
NODE	14.21
DNF-Net	11.96
TabNet	10.51
1D-CNN	7.56
Simple Ensemble	3.15
Deep Ensemble w/o XGBoost	6.91
Deep Ensemble w XGBoost	**2.32**

모델 접근법 - 딥러닝

TABULAR DATA:
DEEP LEARNING IS NOT ALL YOU NEED

- 딥러닝 모델과 트리 기반 부스팅 모델을 앙상블 했을 때, 일반적으로 가장 좋은 성능을 보여줌

- 앙상블 조합 비교군: 2가지 앙상블 테스트 추가 진행
  - Simple Ensemble: XGBoost + SVM + CatBoost
  - Deep Ensemble w/o XGBoost: without XGBoost

- 딥러닝 모델과 트리 기반 부스팅 모델의 앙상블 성능이 좋은 이유 추측
  - 딥러닝 모델의 비선형성
  - 데이터 간의 깊은 상호 작용

TABULAR DATA: DEEP LEARNING IS NOT ALL YOU NEED

Ravid Shwartz-Ziv
ravid.ziv@intel.com
IT AI Group, Intel

Amitai Armon
amitai.armon@intel.com
IT AI Group, Intel

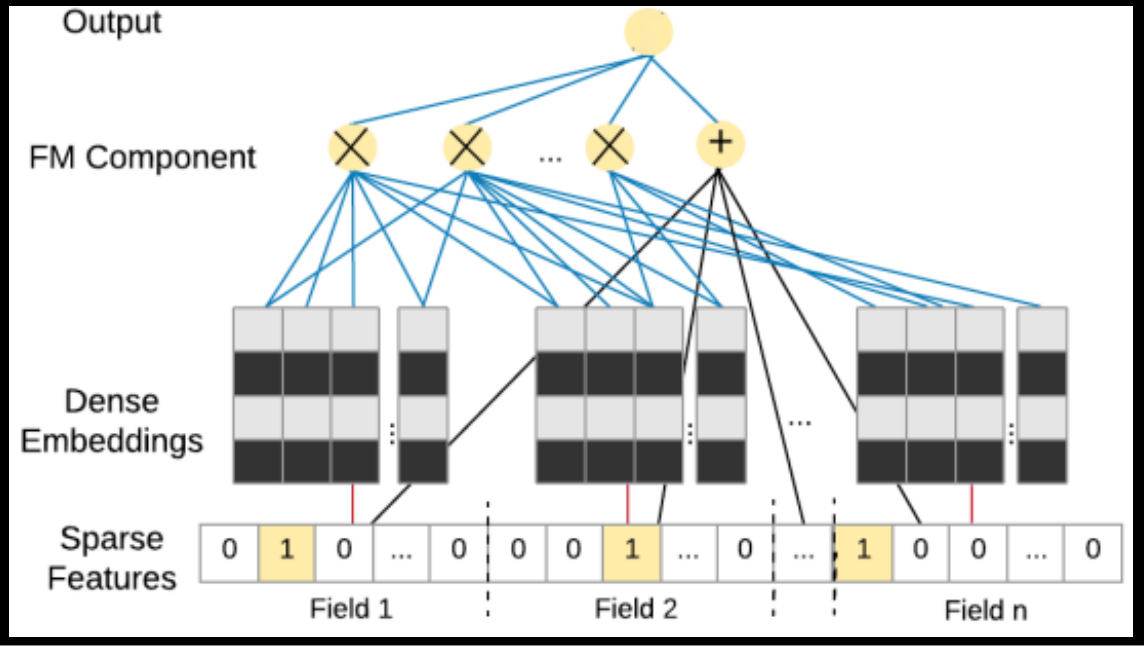
Model Name	Rossman	CoverType	Higgs	Gas	Eye	Gesture
XGBoost	490.18 ± 1.19	3.13 ± 0.09	21.62 ± 0.33	2.18 ± 0.20	**56.07**±0.65	80.64 ± 0.80
NODE	488.59 ± 1.24	4.15 ± 0.13	21.19 ± 0.69	2.17 ± 0.18	68.35 ± 0.66	92.12 ± 0.82
DNF-Net	503.83 ± 1.41	3.96 ± 0.11	23.68 ± 0.83	**1.44** ±0.09	68.38 ± 0.65	86.98 ± 0.74
TabNet	**485.12**±1.93	3.01 ± 0.08	**21.14**±0.20	1.92 ± 0.14	67.13 ± 0.69	96.42 ± 0.87
1D-CNN	493.81 ± 2.23	3.51 ± 0.13	22.33 ± 0.73	1.79 ± 0.19	67.9 ± 0.64	97.89 ± 0.82
Simple Ensemble	488.57 ± 2.14	3.19 ± 0.18	22.46 ± 0.38	2.36 ± 0.13	58.72 ± 0.67	89.45 ± 0.89
Deep Ensemble w/o XGBoost	489.94 ± 2.09	3.52 ± 0.10	22.41 ± 0.54	1.98 ± 0.13	69.28 ± 0.62	93.50 ± 0.75
Deep Ensemble w XGBoost	485.33 ± 1.29	**2.99** ±0.08	22.34 ± 0.81	1.69 ± 0.10	59.43 ± 0.60	**78.93** ±0.73

TabNet DNF-Net

Model Name	YearPrediction	MSLR	Epsilon	Shrutime	Blastchar
XGBoost	77.98 ± 0.11	55.43±2e-2	11.12±3e-2	13.82 ± 0.19	20.39 ± 0.21
NODE	76.39 ± 0.13	55.72±3e-2	**10.39**±1e-2	14.61 ± 0.10	21.40 ± 0.25
DNF-Net	81.21 ± 0.18	56.83±3e-2	12.23±4e-2	16.8 ± 0.09	27.91 ± 0.17
TabNet	83.19 ± 0.19	56.04±1e-2	11.92±3e-2	14.94±, 0.13	23.72 ± 0.19
1D-CNN	78.94 ± 0.14	55.97±4e-2	11.08±6e-2	15.31 ± 0.16	24.68 ± 0.22
Simple Ensemble	78.01 ± 0.17	55.46±4e-2	11.07±4e-2	13.61±, 0.14	21.18 ± 0.17
Deep Ensemble w/o XGBoost	78.99 ± 0.11	55.59±3e-2	10.95±1e-2	14.69 ± 0.11	24.25 ± 0.22
Deep Ensemble w XGBoost	**76.19** ±0.21	**55.38**±1e-2	11.18±1e-2	**13.10**±0.15	**20.18**±0.16

NODE New datasets

Name	Average Relative Performance (%)
XGBoost	3.34
NODE	14.21
DNF-Net	11.96
TabNet	10.51
1D-CNN	7.56
Simple Ensemble	3.15
Deep Ensemble w/o XGBoost	6.91
Deep Ensemble w XGBoost	**2.32**

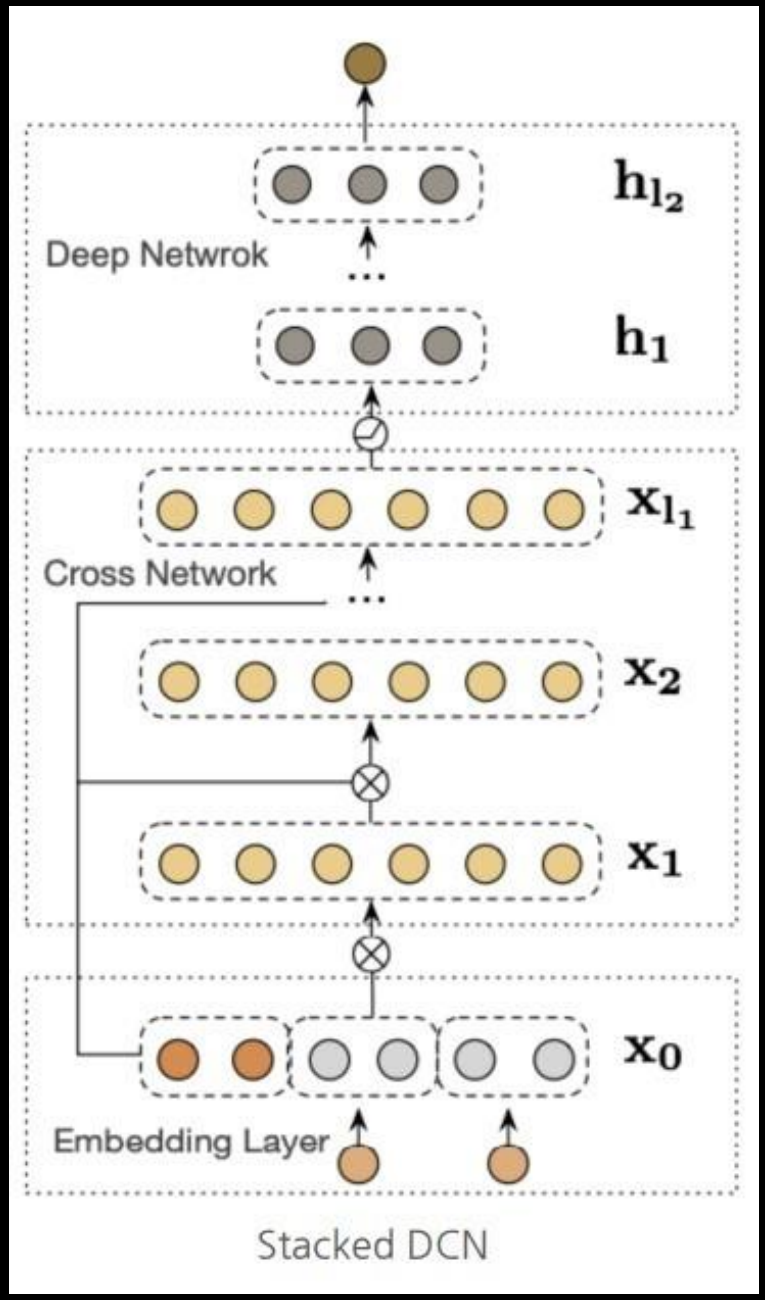


Field-aware Factorization Machine

- Cold Start 문제에 맞춰 데이터를 최대한 활용할 수 있는 모델
- User와 Books의 모든 Field를 입력으로 사용 가능
- 대응되는 Field마다 다른 Latent Vector를 사용해 각각의 Field와의 관계를 적절하게 반영
- Cold Start 유저에 대해 괜찮은 성능을 보여주나, 그렇지 않은 유저에 대해 효과적이지 않음

Deep & Cross Network

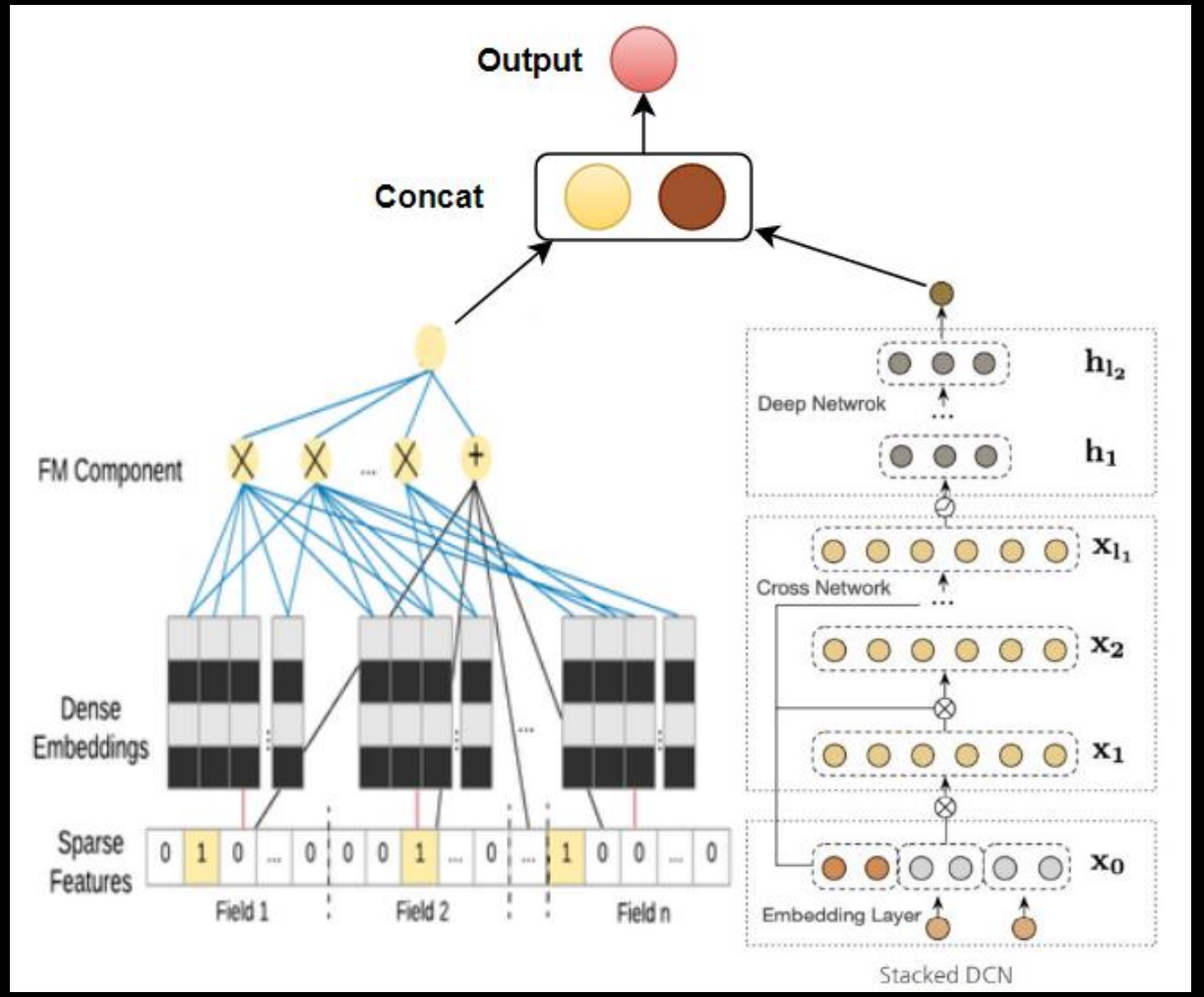
- Sparse하고 Feature가 많은 데이터를 Dense한 Embedding을 만들어 사용
- DCN 모델은 Cross Network를 사용해 효과적인 Feature 교차를 명시적으로 적용
- 유저와 책의 모든 Field들을 넣어 사용하기엔 네트워크의 크기에 비해 데이터 개수가 부족하다고 판단됨



FFM+DCN

1 Epoch Is All You Need To Do

- 서로의 단점을 보완할 수 있도록 FFM과 DCN을 병렬로 합친 모델
- DCN에서 학습하기 힘든 user_id와 isbn 외의 Field들의 암시적인 상호작용은 FFM에서 학습
- FFM에서 효과적이지 못한 유저와 책의 암시적이고 비 선형적인 상호작용은 DCN에서 학습
- FM과 DCN의 base line 출력을 concatenate한 뒤 Linear Layer를 사용하여 출력



모델 고도화

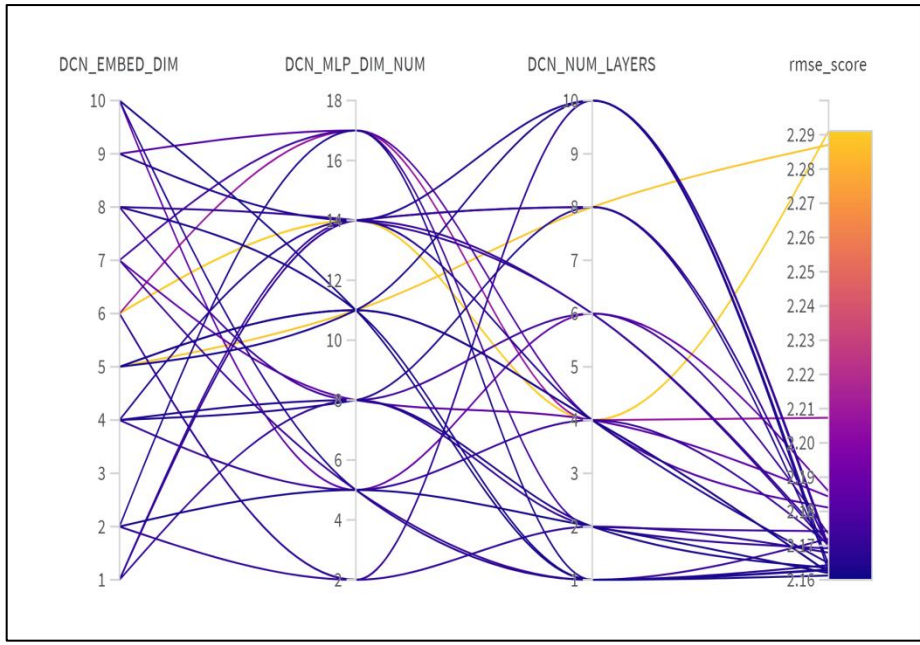
Hyper parameter Tuning

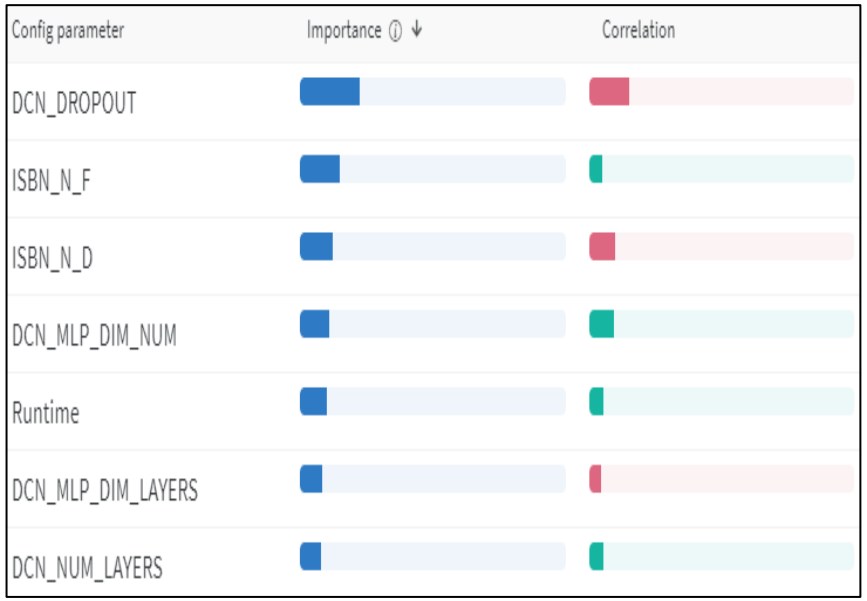
WandB & Sweep

Weights & Biases

Parallel coordinates

- sweep에 지정한 Hyper parameter에 대해서 rmse score와의 연관성을 평행 좌표 그래프를 통해서 시각적으로 관찰 하기 위함





Parameter importance

- Importance(변수중요도) : 어떤 hyper parameter가 최고의 예측 변수였는지 확인할 수 있음
- Correlation(상관관계) : RMSE score에 대한 이상적인 값과 어떤 상관관계가 있는지 파악할 수 있음

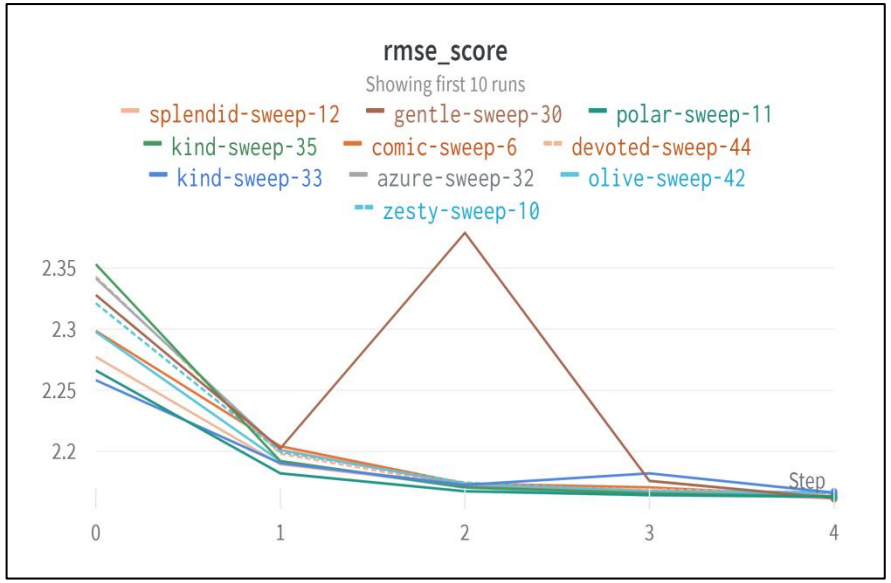
모델 고도화

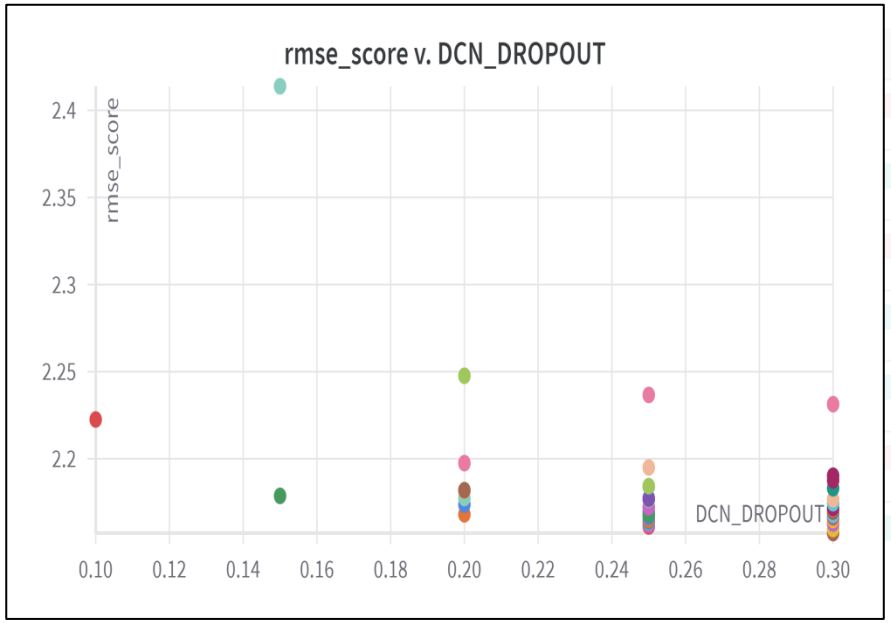
Hyper parameter Tuning
WandB & Sweep

Weights & Biases

Epochs와 RMSE

각 모델이 에포크 별로 RMSE 변화하는 것을 시각적으로 변해줍니다.





하이퍼파라미터와 RMSE

하이퍼파라미터 별로 RMSE를 구분하는 scatter plot 입니다.

모델 고도화

Hyper parameter Tuning

Optuna

Optuna 함수 정의

- learning_rate, n_estimators 비율을 일정하게 지키면서 범위의 변화를 주며 최적화를 진행하였음
- n_trials 의 비율도 조절하면서 진행했지만, 성능의 변화가 거의 없었음
- 시간적 차원 때문에 위 세 가지 hyper parameter에 대해서만 변경하면서 최적화를 진행함

```
def objective(trial):
    param = {
        "random_state":42,
        "objective" : "RMSE",
        "cat_features" : list(train_ratings.drop(['rating'],axis = 1).columns),
        'learning_rate' : trial.suggest_loguniform('learning_rate', 0.01, 0.5),
        'bagging_temperature' :trial.suggest_loguniform('bagging_temperature', 0.01, 100.00),
        "n_estimators":trial.suggest_int("n_estimators", 1000, 10000),
        "max_depth":trial.suggest_int("max_depth", 4, 16),
        'random_strength' :trial.suggest_int('random_strength', 0, 100),
        "l2_leaf_reg":trial.suggest_float("l2_leaf_reg",1e-8,3e-5),
        "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),
        "max_bin": trial.suggest_int("max_bin", 200, 500),
        'od_type': trial.suggest_categorical('od_type', ['IncToDec', 'Iter']),
    }
```

```
test_ratings['rating'] = (test_ratings['pred_0'] + test_ratings['pred_1']
                          test_ratings['pred_6'] + test_ratings['pred
test = test_ratings[['user_id', 'isbn', 'rating']]
test.to_csv('../submit/CAT_10Fold.csv', index = False)
test_ratings.to_csv('../data/CAT_10Fold.csv', index = False)
```

Optuna 활용

- 최적의 hyper paramete를 적용한 각 fold에 담긴 예측 값들을 평균을 냄
- Submit에 저장 후 제출 !

모델 고도화

Hyper parameter Tuning

Optuna & Feature Importance

OPTUNA

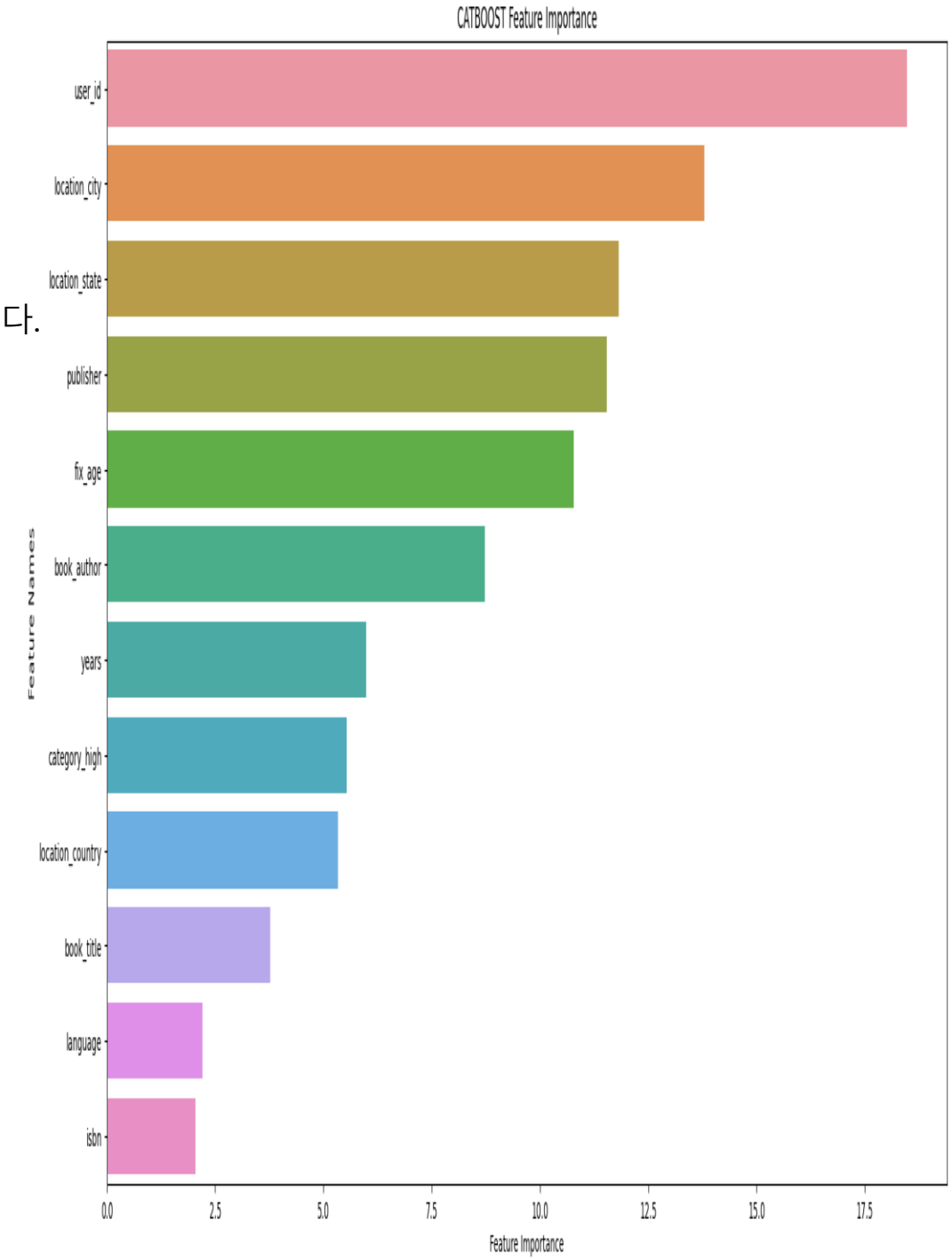
Feature Importance

CatBoost 모델의 변수 중요도를 시각화 했습니다.

변수 중요도 순서는 다음과 같습니다.

1. User ID
2. Location_city
3. Location_state
4. Publisher

...

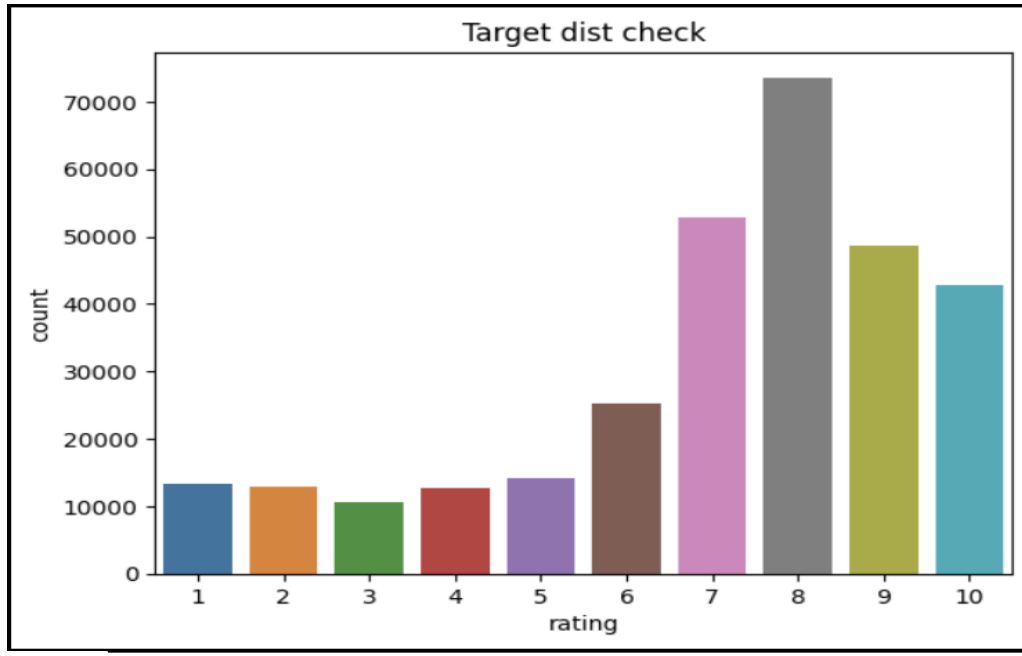


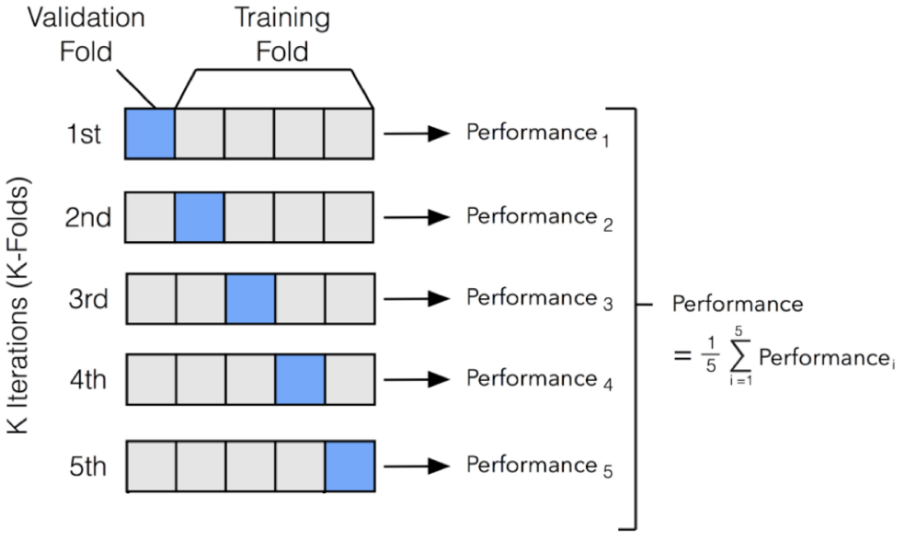
모델 고도화

Stratified K-FOLD

Class UnBalance

- train data의 rating(y값) 평점 분포
- 7~ 10점이 약 65% 이상 차지
  - class 불균형
    - Stratified K-fold 통해 해결





Stratified K-Fold

- 기존 K-Fold 사용할 경우 하나 또는 그 이상의 Fold에 Class가 치우칠 수 있는 문제가 발생 할 수 있음
- Each Fold – 1~10점의 비율이 동일하게 구성 되서 불균형 문제를 해결할 수 있음

모델 고도화

Optuna & Stratified K-FOLD

Each fold Applying Optuna

- 각 폴드에 대해서 Optuna를 진행해줌
- 장점 : 기존의 traindata에 hold-out 기법을 적용하여 Optuna를 1 번 진행한 것 보다 성능이 향상됨
- 단점 : 시간적 Resorce 소모가 커짐

```
for fold in range(0,10):
    print(f'======================================{fold+1}=========
    train_idx, valid_idx = folds[fold]
    X_train = train_ratings.drop(['rating'],axis = 1).iloc[train_idx]
    X_valid = train_ratings.drop(['rating'],axis = 1).iloc[valid_idx]
    y_train = train_ratings['rating'].iloc[train_idx]
    y_valid = train_ratings['rating'].iloc[valid_idx]

    sampler = optuna.samplers.TPESampler(seed=42)
    study = optuna.create_study(
        study_name = 'cat_parameter_opt',
        direction = 'minimize',
        sampler = sampler,
    )
    study.optimize(objective, n_trials=10)


    model = CatBoostRegressor(**study.best_params, task_type = 'GPU',
                              devices = '0', random_state = SEED, objective = 'RMSE',
                              cat_features = list(train_ratings.drop(['rating'],axis = 1).columns))
    model.fit(X_train, y_train)

    pred = model.predict(test_ratings.drop(['rating'],axis = 1))
    test_ratings[f'pred_{fold}'] = pred
    print(f'=================================================
```

모델 고도화에 따른 성능 변화

CATBOOST
FFDCN
ENSEMBLE

고도화에 따른 모델 성능 진척도

1. CatBoost

- X / 2.14
- Optuna / 2.1352
- Optuna + 5K-fold / 2.1263
- Optuna + 10K-fold / **2.1220**

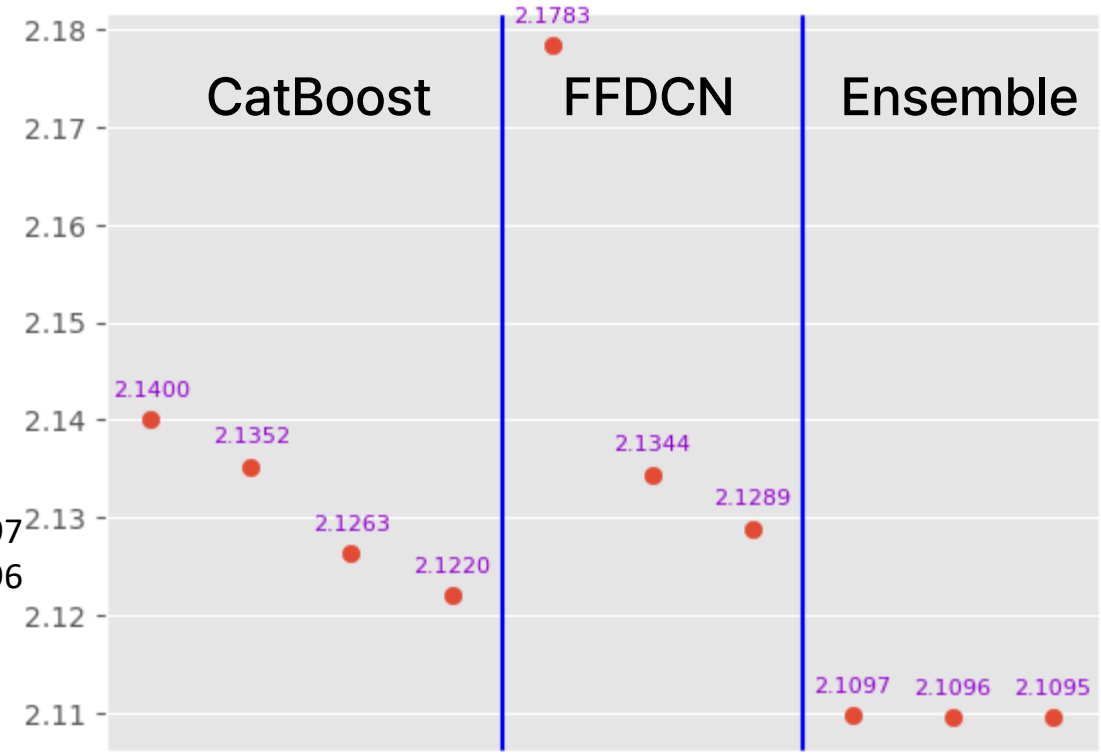
2. FFDCN

- X / 2.1783
- Optuna / 2.1344
- Optuna + 5K-fold / **2.1289**

3. Ensemble

- 0.5 * CatBoost + 0.5 * FFDCN / 2.1097
- 0.6 * CatBoost + 0.4 * FFDCN / 2.1096
- 0.55 * CatBoost + 0.45 * FFDCN

=> **2.1095 (최고 성능)**



모델 결과

기간 별 모델 성능 진척도

- 10/26: DeepCoNN 2.4856 (Start)
- 10/27: CatBoost 2.1732
  CNN_FM 3.0430
  FM(5-fold) 2.5313
- 10/28: TabNet 2.4772
  CatBoost(전처리) 2.14
  XGBoost 2.3978
- 10/29: CatBoost(Optuna, StK-Fold)
  2.1352
- 11/01: CatBoost(Opt in Fold) 2.1263
  FFM+DCN 2.4152
  FM(sweep) 2.2413
- 11/02: FFM+DCN(Optuna) 2.1783
- 11/03: CatBoost(10-fold) 2.12
  Enssemble(Cat + FFDCN) 2.11