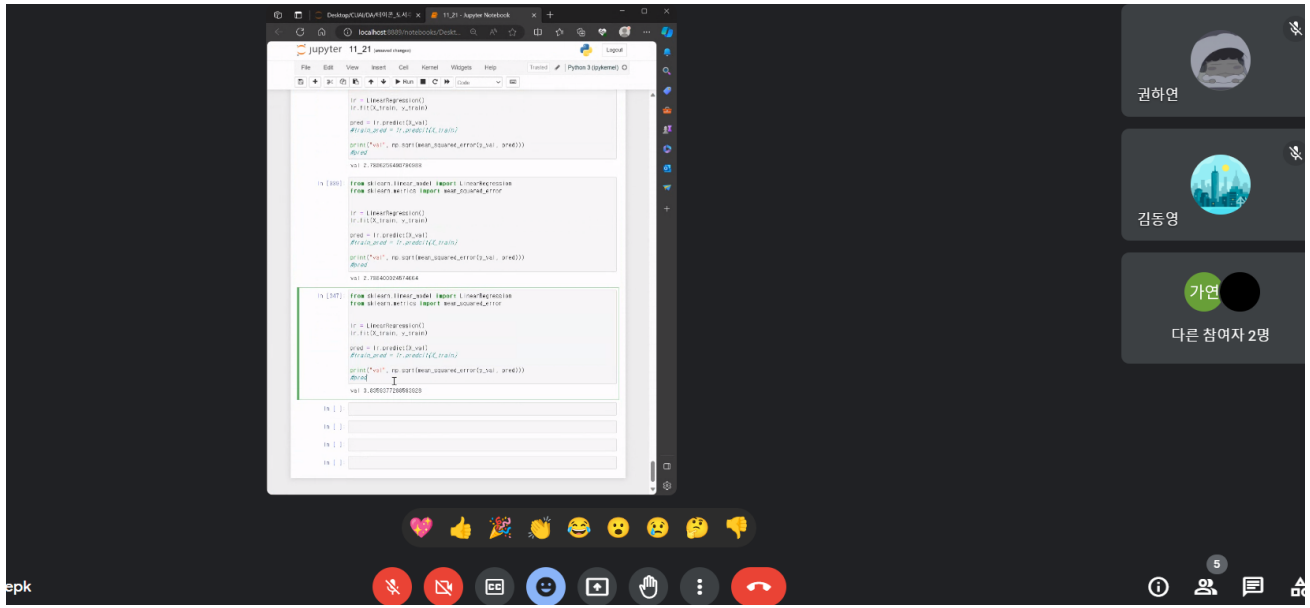
# DA 1팀

2023.11.28

발표자 김예원

# 팀원 소개



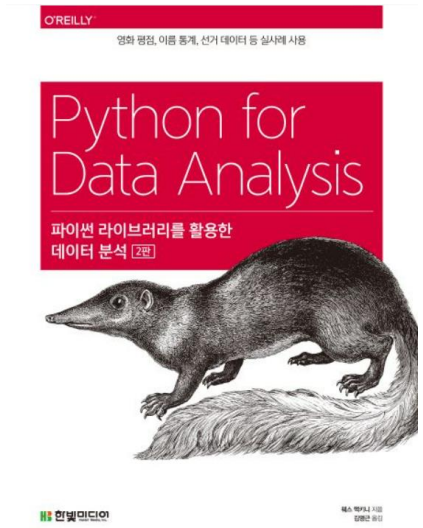

화학신소재공학부 고가연

소프트웨어학부 권하연

소프트웨어학부 김동영

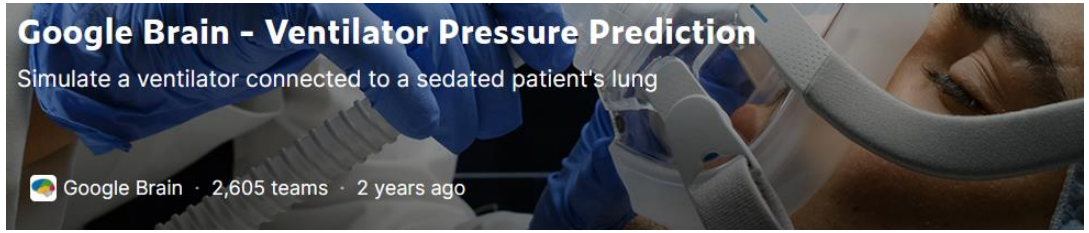
AI학과 김예원

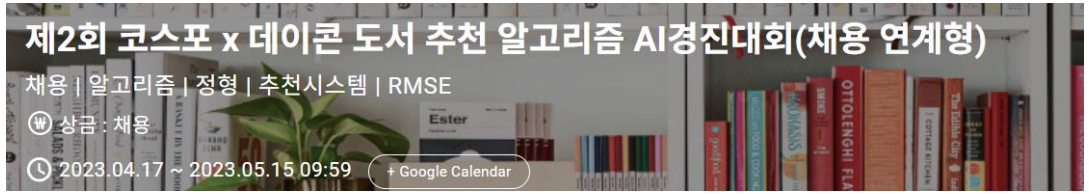
# 스터디 과정




~ 중간고사

- 〈파이썬 라이브러리를 활용한 데이터 분석〉 스터디
- 매주 3 챕터씩 진행




중간고사 이후

- 캐글 〈ventilator pressure prediction〉 EDA
- 데이콘 〈도서 추천 대회〉 프로젝트 진행

# 도서 추천 알고리즘 AI경진대회


=〉 사용자와 책에 대한 데이터가 주어지고,

사용자가 매긴 Book rating 점수를 예측하는 회귀 문제

목표:
EDA를 잘 해서, 성능 향상에 도움이 되는 파생변수를 잘 만들어보자!

# 데이터 소개

-ID : 샘플 고유 ID
-User_ID : 유저 고유 ID
-Book_ID : 도서 고유 ID

유저 정보
-Age : 나이
-Location : 지역

도서 정보
-Book_Title : 도서 명
-Book_Author : 도서 저자
-Year_Of_Publication : 도서 출판년도
-Publisher : 출판사
-Book_Rating : 유저가 도서에 부여한 평점 (0 ~ 10점)
• 단, 0점인 경우에는 유저가 해당 도서에 관심이 없고 관련이 없는 경우

**[제출양식] sample_submission.csv**

-ID : 샘플 고유 ID
-Book_Rating : 예측한 유저가 도서에 부여할 평점
• 단, 0점인 경우에는 유저가 해당 도서에 관심이 없고 관련이 없는 경우

ID | User-ID | Book-ID | Age | Location | Book-Title | Book-Author | Year-Of-Publication | Publisher | Book-Rating

타겟

# 도서 평점 데이터

예측해야하는 타겟값

| | ID | User-ID | Book-ID | Age | Location | Book-Title | Book-Author | Year-Of-Publication | Publisher | Book-Rating |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | TRAIN_000000 | USER_00000 | BOOK_044368 | 23.0 | sackville, new brunswick, canada | Road Taken | Rona Jaffe | 2001.0 | Mira | 8 |
| 1 | TRAIN_000001 | USER_00000 | BOOK_081205 | 23.0 | sackville, new brunswick, canada | Macbeth (New Penguin Shakespeare) | William Shakespeare | 1981.0 | Penguin Books | 8 |
| 2 | TRAIN_000002 | USER_00000 | BOOK_086781 | 23.0 | sackville, new brunswick, canada | Waverley (Penguin English Library) | Walter Scott | 1981.0 | Penguin Books | 0 |
| 3 | TRAIN_000003 | USER_00000 | BOOK_098622 | 23.0 | sackville, new brunswick, canada | Mother Earth Father Sky | Sue Harrison | 1991.0 | Avon | 0 |
| 4 | TRAIN_000004 | USER_00000 | BOOK_180810 | 23.0 | sackville, new brunswick, canada | She Who Remembers | Linda Lay Shuler | 1989.0 | Signet Book | 8 |

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 871393 entries, 0 to 871392
Data columns (total 10 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ID                   871393 non-null  object
 1   User-ID              871393 non-null  object
 2   Book-ID              871393 non-null  object
 3   Age                  871393 non-null  float64
 4   Location             871393 non-null  object
 5   Book-Title           871393 non-null  object
 6   Book-Author          871393 non-null  object
 7   Year-Of-Publication  871393 non-null  float64
 8   Publisher            871393 non-null  object
 9   Book-Rating          871393 non-null  int64
dtypes: float64(2), int64(1), object(7)
memory usage: 66.5+ MB
```

## ▶ 결측치 확인

```
train.isnull().sum()
```

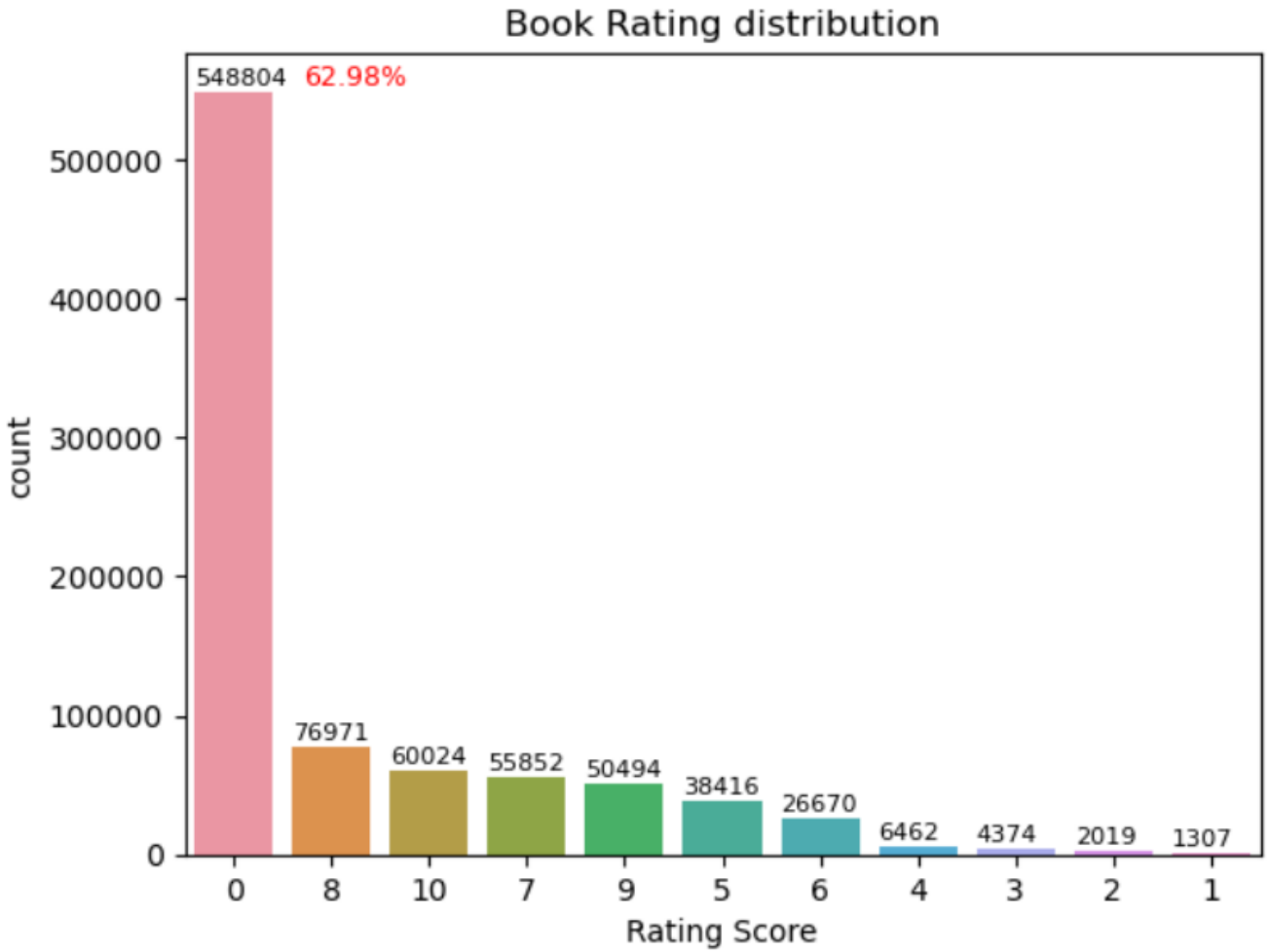
```
ID                     0
User-ID                0
Book-ID                0
Age                    0
Location               0
Book-Title             0
Book-Author            0
Year-Of-Publication    0
Publisher              0
Book-Rating            0
dtype: int64
```

모든 칼럼에 결측치 없음

도서 평점 데이터 ID User-ID Book-ID Age Location Book-Title Book-Author Year-Of-Publication Publisher Book-Rating

타겟

# Book-Rating(타겟)




평점 범위: 0~10점

전체 871393개의 평점 中

0점이 62.98%를
차지할 정도로 높았음

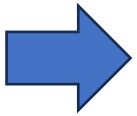
(0점: 유저가 해당 도서에 관심없는 경우)

도서 평점 데이터 ID User-ID Book-ID Age **Location** Book-Title Book-Author Year-Of-Publication Publisher Book-Rating

타겟

# Location

| Location |
|---|
| sackville, new brunswick, canada |
| sackville, new brunswick, canada |
| sackville, new brunswick, canada |
| sackville, new brunswick, canada |
| sackville, new brunswick, canada |



| City | State | Country |
|---|---|---|
| sackville | new brunswick | canada |
| sackville | new brunswick | canada |
| sackville | new brunswick | canada |
| sackville | new brunswick | canada |
| sackville | new brunswick | canada |

Location

=〉 [City, State, Country]

Location 칼럼은 삭제

도서 평점 데이터 ID User-ID Book-ID Age **Location** Book-Title Book-Author Year-Of-Publication Publisher Book-Rating

타겟

# Location

```
=============city============
n/a                          13732
toronto                      12516
chicago                       7661
seattle                       7144
ottawa                        7108
                              ...
remseck                          1
kardamili                        1
not sure                         1
sanur                            1
castiglion fiorentino            1
```

사실 판다스의 결측치표기(NaN)가 아닌
문자열 ‘n/a’로 표기된 결측치가 있었음

City 칼럼에 문자열 'n/a’ 결측치가 가장 많다

City, State, Country 中

<Country>만 사용하기로 결정

도서 평점 데이터 ID User-ID Book-ID Age **Location** Book-Title Book-Author Year-Of-Publication Publisher Book-Rating

타겟

# Location – Country

```
cnt = 0
def process_country(country:str):
    global cnt
    if country in ['jersey','new jersey','united staes','united state','united sat
        cnt +=1
        return 'usa'

    elif country in ['united kindgonm','england','unitedkingdom','unitedkindgonm'
        cnt +=1
        return 'uk'

    elif country in ['unitedarabemirates']:
        cnt +=1
        return 'uae'

    elif country in ['pender','cananda'] :
        cnt +=1
        return 'canada'

    elif country in ['italia', 'litalia'] :
        cnt +=1
        return 'italia'

    elif country in ["" , "na","k1c7b1","the","ysa","everywhereandanywhere", 'c',
        cnt +=1
        return 'unknown'
    elif country in ['espaa','madrid','catalonia','catalunyaspain' ,'catalunya']
        cnt +=1
        return 'spain'

    elif country in ['lafrance', 'bergued']:
        cnt +=1
        return 'france'

    elif country in ['newzealand']:
        cnt +=1
        return 'nz'
    return country

train['Country'] = train['Country'].apply(lambda x : process_country(x))
```

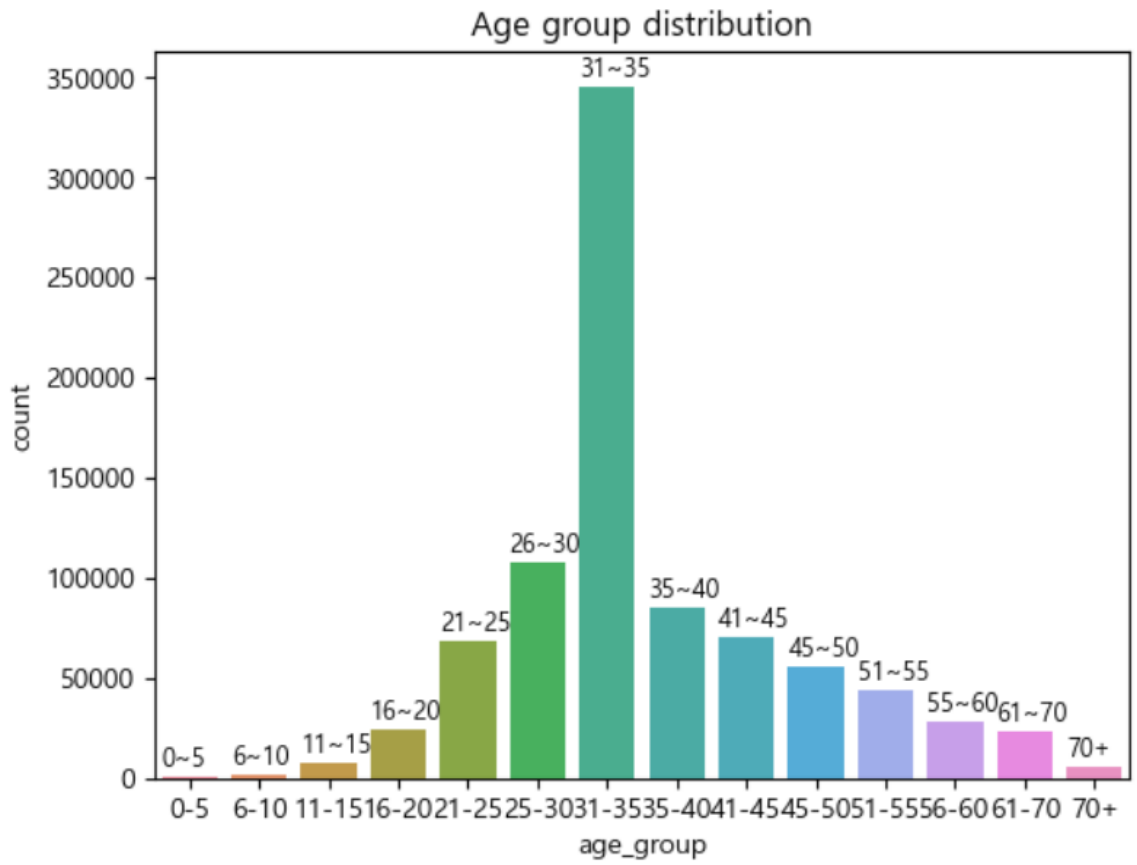
‘u.s.a’
‘united kindgonm’
‘cananda’,
‘new jersey’ ,,,

Country에서 오탈자, 지역 이름 등
잘못 작성된 국가명이 많았음

87만 1393개의 데이터 中

22563개의 country명(전체의 2.6%)를 교정

# Age




20대~40대 초반까지 rating 평점을 매긴 사람들이 몰려있다

특히 30대 초중반의 유저들의 평이 월등히 많았다

도서 평점 데이터 | ID | User-ID | Book-ID | Age | Location | Book-Title | Book-Author | Year-Of-Publication | Publisher | Book-Rating

타겟

# Age

```
def age_(age):
    if (age < 10):
        return 1
    elif(age < 20):
        return 2
    elif(age < 30):
        return 3
    elif(age < 40):
        return 4
    elif(age < 50):
        return 5
    elif(age < 60):
        return 6
    elif(age < 70):
        return 7
    elif(age < 80):
        return 8
    elif(age < 90):
        return 9
    else: # abnormal
        return 10

train['Age_cat'] = train.a
train.drop(['Age'], axis =
```

| Age_cat |
|---|
| 3 |
| 3 |
| 3 |
| 3 |
| 3 |

Age를 10대, 20대, 30대,,, 별로 카테고리화

-원본 데이터의 Age 칼럼은 삭제
-Age_cat을 추가

도서 평점 데이터 ID User-ID Book-ID Age Location **Book-Title** Book-Author Year-Of-Publication Publisher Book-Rating

타겟

# Book-Title

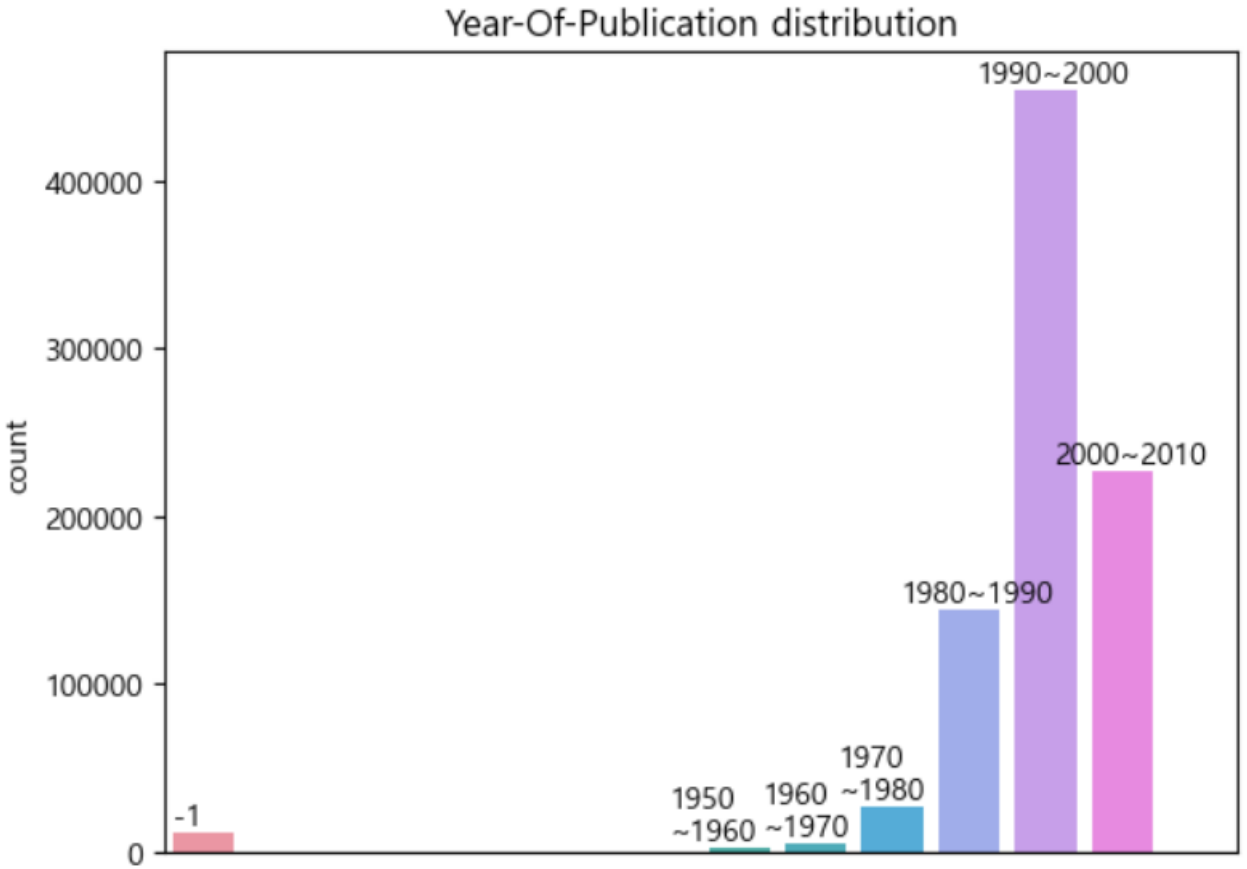
| Book-ID | Book-Title |
|---|---|
| BOOK_044368 | Road Taken |
| BOOK_081205 | Macbeth (New Penguin Shakespeare) |
| BOOK_086781 | Waverley (Penguin English Library) |
| BOOK_098622 | Mother Earth Father Sky |
| BOOK_180810 | She Who Remembers |

Book-ID를 기준으로 Book-Title은 고윳값일 것이라고 예상

➔ 확인해보니 Book-ID에 따른 Book-Title은 고윳값이었음

➔ Book-Title을 칼럼에서 삭제

# Year-Of-Publication




1990~2000년대가 52%로 가장 많았고

출판연도가 '-1'로 기록된 이상치도
11515개로 적지 않았음

도서 평점 데이터 ID User-ID Book-ID Age Location Book-Title Book-Author Year-Of-Publication Publisher Book-Rating

타겟

# Year-Of-Publication

```
def yop_(yop):
    if(yop < 1910):
        return 1
    elif(yop < 1920):
        return 2
    elif(yop < 1930):
        return 3
    elif(yop < 1940):
        return 4
    elif(yop < 1950):
        return 5
    elif(yop < 1960):
        return 6
    elif(yop < 1970):
        return 7
    elif(yop < 1980):
        return 8
    elif(yop < 1990):
        return 9
    elif(yop < 2000):
        return 10
    elif(yop>= 2000):
        return 11
    else: #abnormal
        print(yop)
        return 12
```

| YOP_cat |
|---|
| 11 |
| 9 |
| 9 |
| 10 |
| 9 |

1900년 이전 연도(-1, 1300,,,) 등은 카테고리 1로 변환

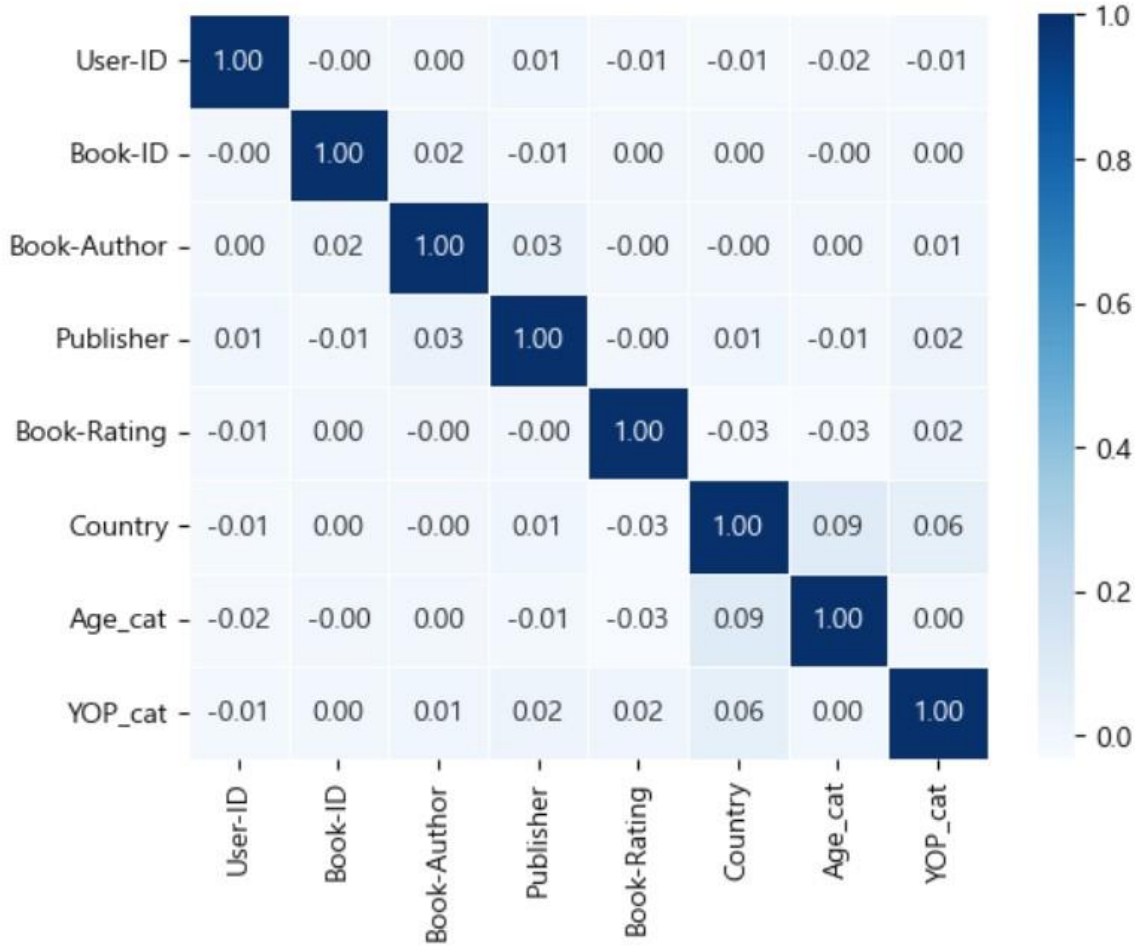
-원본 데이터의 YOP 칼럼은 삭제
-YOP_cat을 추가

# 레이블 인코딩

```
from sklearn.preprocessing import LabelEncoder

cat_cols = train.select_dtypes(include = [object]).columns

for col in cat_cols:
    encoder = LabelEncoder()
    encoder.fit(train[col])
    train[col] = encoder.transform(train[col])
```

| | User-ID | Book-ID | Book-Author | Publisher | Country | Age_cat | YOP_cat |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 39921 | 77475 | 9093 | 54 | 3 | 11 |
| **1** | 0 | 73045 | 91647 | 10515 | 54 | 3 | 9 |
| **2** | 0 | 78014 | 90102 | 10515 | 54 | 3 | 9 |
| **3** | 0 | 88719 | 83754 | 1166 | 54 | 3 | 10 |
| **4** | 0 | 162881 | 54167 | 12609 | 54 | 3 | 9 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **871388** | 83251 | 72982 | 52004 | 6162 | 298 | 4 | 10 |
| **871389** | 83252 | 232668 | 16876 | 6091 | 298 | 4 | 11 |
| **871390** | 83253 | 64665 | 36822 | 13698 | 54 | 5 | 11 |
| **871391** | 83254 | 227657 | 87010 | 12989 | 298 | 5 | 10 |
| **871392** | 83255 | 117702 | 4221 | 6037 | 137 | 4 | 10 |

# 피쳐 상관관계 히트맵




결측치 제거
이상치 처리
카테고리화
레이블 인코딩만 한 데이터에서는

모든 피쳐가 Book-Rating과
상관관계가 거의 없었음

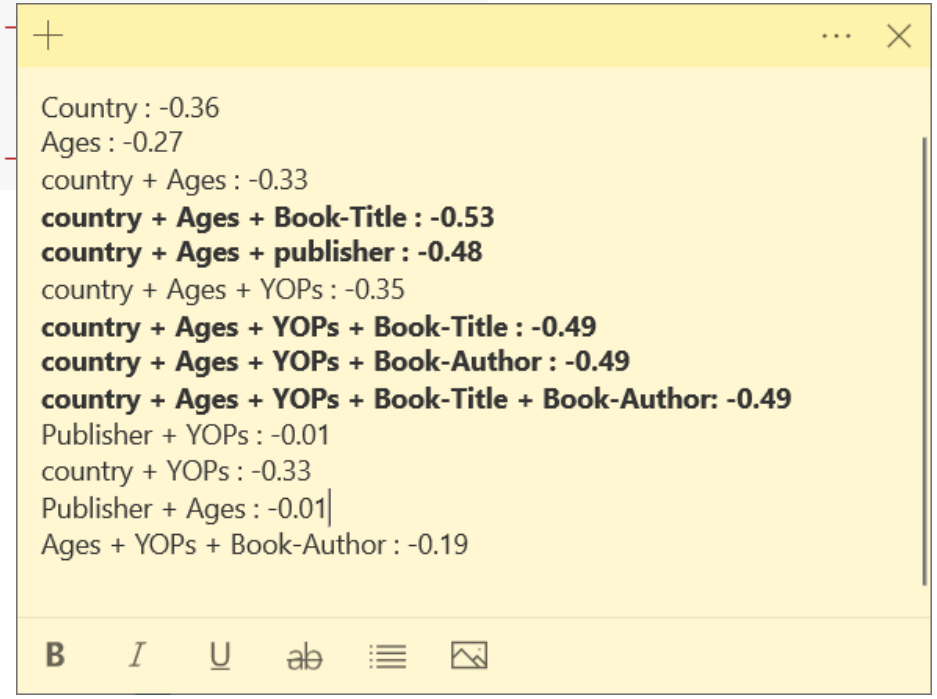
모델: catboost_regressor
스코어 : 3.651851755

# 파생변수 추가

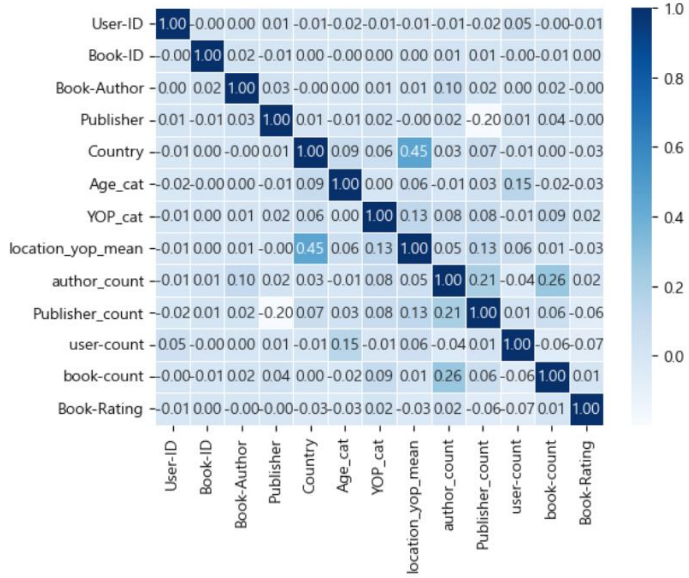
EDA로 다른 변수들 간의 상관관계 파악을 하지 못해,
기존 피쳐들을 조합해 새로운 정보라고 생각되는 것들을 추가하였음
〈다작 작가인지, 다독 사용자인지, 유명한 책인지 등..〉

```
train['location_yop_mean'] = train.groupby("Country")['YOP_cat'].transform('mean')

temp = dict(train['Book-Author'].value_counts())
train['author_count'] = train.apply(lambda x: temp[x['Book-Author']], axis=1)

temp = dict(train['Publisher'].value_counts())
train['Publisher_count'] = train.apply(lambda x: temp[x['Publisher']], axis=1)

temp = dict(train['User-ID'].value_counts())
train['user-count'] = train.apply(lambda x: temp[x['User-

temp = dict(train['Book-ID'].value_counts())
train['book-count'] = train.apply(lambda x: temp[x['Book-
```

- Author_count
- Publisher_count
- User_count
- Book_count



# 파생변수 추가

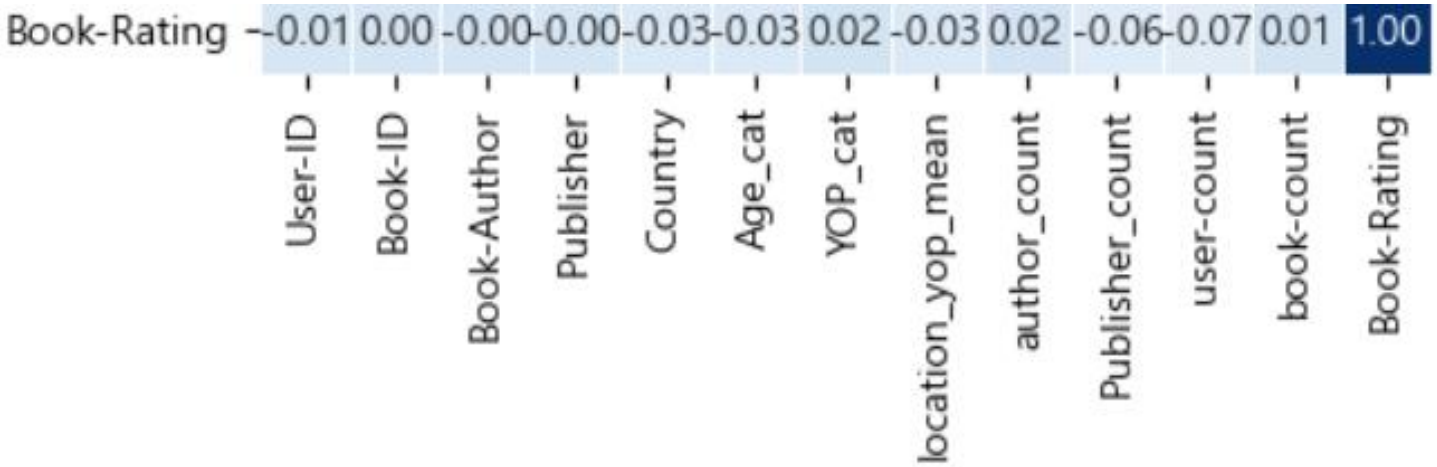



- Author_count
- Publisher_count
- User_count
- Book_count

모델: catboost_regressor

author-count, publisher-count 추가: 3.632075473

user-count, book-count 추가: 3.471357467



도서 평점 데이터 | ID | User-ID | Book-ID | Age | Location | Book-Title | Book-Author | Year-Of-Publication | Publisher | Book-Rating

타겟

# 파생변수 추가

Catboost_regressor 모델에
Cat_features 에 카테고리형 피쳐를 지정하니 성능이 크게 향상되었음

- Author_count
- Publisher_count
- User_count
- Book_count

모델: catboost_regressor

author-count, publisher-count 추가: 3.632075473

user-count, book-count 추가: 3.471357467

k-fold 전 3.309

k-fold 후 3.286

파생변수 추가 후 3.283

private 점수도 같이 오른 것으로 보아

유의미한 성능의 향상이 있다고 평가할 수 있지만(랭킹 변동이 있었음)

성능 향상의 폭이 크지 않음

-> 추가적인 EDA 필요

=〉 4% 순위권

# 결론

파생변수 추가를 많이 하지 않았음에도 catboost_regressor(cat_features 추가) 와 k-fold 만 했을 때 나쁘지 않은 결과가 나왔음

파생변수를 이용한 성능 향상을 통해 EDA와 파생변수 추가의 중요성을 알게 됨.

catboost 만 써도 좋은 결과가 나오긴 하지만, 모델링의 관점에서 다양한 방법들을 사용해보았음에도 일정 수준 이상으로는 점수가 오르지 않는 것으로 보아

결국 순위권(1%) 에 들기 위해서는 데이터 분석과

- 파생변수 추가

- 이상치 처리

- 결측치 처리 방법이 가장 중요할 것으로 분석됨

- 중간고사 이전 책을 통해 익혔던 pandas 사용법이 EDA 시에 도움이 되었음

감사합니다