2020 가을 프로젝트 1주차

DACON의 교육에서 '영화 관객수 예측 모델 개발'을 주제로 골랐다. 여름에는 이미지 분류를 해보았으니 이번에는 뭔가 흔하면서도 꼭 알아야 될 거 같은 예측에 대해서 골랐다. 일단 이번주는 공유코드를 보고 공부해보기로 했다. 공유코드는 모델을 lightGBM, k-fold lightGBM, feature engineering, grid search 이렇게 네 가지를 이용했는데 이를 모두 공부한 다음 LSTM을 모델로 사용하여 예측해보려고 한다. 일단 이번주는 lightGBM, k-fold light GBM으로 코드를 공부해보았다.

1. Library & Data 불러오기

```
[1]  import pandas as pd #판다스 패키지 불러오기
     import lightgbm as lgb
```

```
[2]  train = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/movies_train.csv',encoding = 'utf-8') # train data
     test = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/movies_test.csv',encoding = 'utf-8') # test data
     submission = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/submission.csv', encoding = 'utf-8') # submission data
```

2. 탐색적 자료분석

```
[6]  print(train.shape)
     print(test.shape)
     print(submission.shape)
```

```
(600, 12)
(243, 11)
(243, 2)
```

```
[7]  train.info
```

```
<bound method DataFrame.info of          title   distributor genre  ... num_staff  num_actor box_off_num
0       개들의 전쟁       롯데엔터테인먼트    액션  ...         91          2       23398
1         내부자들         (주)쇼박스   느와르  ...        387          3     7072501
2    은밀하게 위대하게         (주)쇼박스    액션  ...        343          4     6959083
3     나는 공무원이다         (주)NEW   코미디  ...         20          6      217866
4         불량남녀  쇼박스(주)미디어플렉스   코미디  ...        251          2      483387
..         ...          ...   ...  ...        ...        ...         ...
595         해무         (주)NEW   드라마  ...        510          7     1475091
596       파파로티         (주)쇼박스   드라마  ...        286          6     1716438
597      살인의 강      (주)마운틴픽쳐스    공포  ...        123          4        2475
598     악의 연대기       CJ 엔터테인먼트   느와르  ...        431          4     2192525
599        베틀린      CJ 엔터테인먼트    액션  ...        363          5     7166532

[600 rows x 12 columns]>
```

```
[8]  test.info
```

```
<bound method DataFrame.info of              title  distributor   genre  ... dir_prev_num  num_staff num_actor
0             용서는 없다       시네마서비스      느와르  ...            2        304         3
1        아빠가 여자를 좋아해        (주)쇼박스  멜로/로맨스  ...            4        275         3
2              하모니    CJ 엔터테인먼트      드라마  ...            3        419         7
3              의형제        (주)쇼박스      액션  ...            2        408         2
4            평행 이론    CJ 엔터테인먼트       공포  ...            1        380         1
..             ...          ...   ...  ...          ...        ...       ...
238        해에게서 소년에게          디씨드      드라마  ...            1          4         4
239           울보 권투부        인디스토리   다큐멘터리  ...            0         18         2
240            어떤살인   (주)컨텐츠온미디어      느와르  ...            0        224         4
241        말하지 못한 비밀   (주)씨타마운틴픽쳐스      드라마  ...            1         68         7
242  조선안방 스캔들-칠거지악 2    (주) 케이알씨지  멜로/로맨스  ...            0         10         4

[243 rows x 11 columns]>
```

[9] train.describe()

	time	dir_prev_bfnum	dir_prev_num	num_staff	num_actor	box_off_num
count	600.000000	2.700000e+02	600.000000	600.000000	600.000000	6.000000e+02
mean	100.863333	1.050443e+06	0.876667	151.118333	3.706667	7.081818e+05
std	18.097528	1.791408e+06	1.183409	165.654671	2.446889	1.828006e+06
min	45.000000	1.000000e+00	0.000000	0.000000	0.000000	1.000000e+00
25%	89.000000	2.038000e+04	0.000000	17.000000	2.000000	1.297250e+03
50%	100.000000	4.784236e+05	0.000000	82.500000	3.000000	1.259100e+04
75%	114.000000	1.286569e+06	2.000000	264.000000	4.000000	4.798868e+05
max	180.000000	1.761531e+07	5.000000	869.000000	25.000000	1.426277e+07

[11] train[['genre', 'box_off_num']].groupby('genre').mean().sort_values('box_off_num')

genre	box_off_num
뮤지컬	6627.0
다큐멘터리	67172.3
서스펜스	82611.0
애니메이션	181926.7
멜로/로맨스	425968.0
미스터리	527548.2
공포	590832.5
드라마	625689.8
코미디	1193914.0
SF	1788345.7
액션	2203974.1
느와르	2263695.1

[12] pd.reset_option('display.float_format')

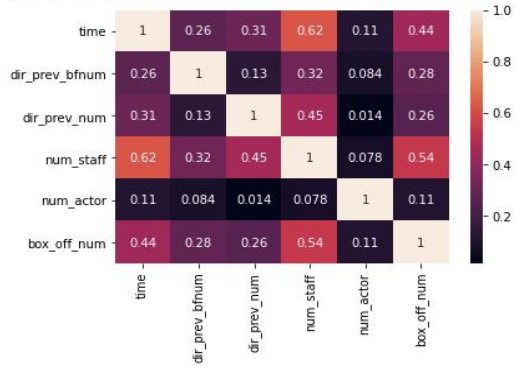
[13] train.corr() #correlation coefficient

	time	dir_prev_bfnum	dir_prev_num	num_staff	num_actor	box_off_num
time	1.000000	0.264675	0.306727	0.623205	0.114153	0.441452
dir_prev_bfnum	0.264675	1.000000	0.131822	0.323521	0.083818	0.283184
dir_prev_num	0.306727	0.131822	1.000000	0.450706	0.014006	0.259674
num_staff	0.623205	0.323521	0.450706	1.000000	0.077871	0.544265
num_actor	0.114153	0.083818	0.014006	0.077871	1.000000	0.111179
box_off_num	0.441452	0.283184	0.259674	0.544265	0.111179	1.000000

[14] import seaborn as sns

[15] sns.heatmap(train.corr(), annot = True)

<matplotlib.axes._subplots.AxesSubplot at 0x7f0f96aff5c0>



3. 데이터 전처리

Data Cleansing & Pre-processing

```
[16] test.isna().sum()/600    #결측치 여부 확인

     title            0.000000
     distributor      0.000000
     genre            0.000000
     release_time     0.000000
     time             0.000000
     screening_rat    0.000000
     director         0.000000
     dir_prev_bfnum   0.226667
     dir_prev_num     0.000000
     num_staff        0.000000
     num_actor        0.000000
     dtype: float64
```

```
[17] train[train['dir_prev_bfnum'].isna()]['dir_prev_num'].sum()

     0
```

```
[18] train['dir_prev_bfnum'].fillna(0, inplace = True)
```

```
[19] test[test['dir_prev_bfnum'].isna()]['dir_prev_num'].sum()

     0
```

```
[20] test['dir_prev_bfnum'].fillna(0, inplace = True)
```

4. 변수 선택 및 데이터 구축 Feature engineering & initial modeling

```
[21] model = lgb.LGBMRegressor(random_state=777, n_estimators=1000)
```

```
[22] features = ['time', 'dir_prev_num', 'num_staff', 'num_actor']
     target = ['box_off_num']
```

```
[23] X_train, X_test, y_train = train[features], test[features], train[target]
```

여기까지는 모두 같다. 이제부터 모델에 따라 코드가 달라진다.

<lightGBM>

lightGBM은 부스팅 모델로 기존 모델이 못 맞춘 문제에 가중치를 두어 지속적으로 학습을 이어나간다. 다른 부스팅 모델에 비해 속도가 빠르다.

5. 모델 학습 및 검증

(1)LightGBM

```
[24] model.fit(X_train, y_train)

     LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
                   importance_type='split', learning_rate=0.1, max_depth=-1,
                   min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
                   n_estimators=1000, n_jobs=-1, num_leaves=31, objective=None,
                   random_state=777, reg_alpha=0.0, reg_lambda=0.0, silent=True,
                   subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

```
[25] singleLGBM = submission.copy()
```

```
singleLGBM.head()
```

	title	box_off_num
0	용서는 없다	0
1	아빠가 여자를 좋아해	0
2	하모니	0
3	의형제	0
4	평행 이론	0

```
[27] singleLGBM['box_off_num'] = model.predict(X_test)
```

```
singleLGBM.to_csv('singleLGBM.csv', index = False)
```

결과는 이렇게 나온다. (윗부분만)

singleLGBM.csv

총 243개 항목 중 1~100 필터

title	box_off_num
용서는 없다	2817995.2423639353
아빠가 여자를 좋아해	375377.2309689026
하모니	-569324.328434587
의형제	1581189.003443361
평행 이론	-527780.6357232291
회오리 바람	-292772.6981921616
경계도시 2	-21290.51915854855
이웃집 남자	3411.7397152532
아마존의 눈물 극장판	-120173.6380219186
반가운 살인자	3266359.409243073
집 나온 남자들	-346034.9561255791
베스트셀러	1174444.5993783975
작은 연못	2074626.2123943283
경	-48659.23011264903
대한민국 1%	813362.2441990551
포화 속으로	3243106.8157236245
나쁜 놈이 더 잘 잔다	-110172.9988931485
맨발의 꿈	3817956.8788898704
요술	-749394.5792219058
마법천자문: 대마왕의 부활을 막아라	186604.7951356381
인플루언스	112440.8723393296
미스터 좀비	53503.9502453996
김복남 살인사건의 전말	473808.3904602472
엄지아빠	242974.4435939
탈주	89374.88704163235
해결사	2324788.88366019
할	712307.8929964601
된장	1106991.361848784
조금만 더 가까이	683798.1931952764
우리 만난 적 있나요	-255207.09080207944
쩨쩨한 로맨스	4295695.421669217
용서	-107542.89038208067
브라보! 재즈 라이프	-20789.585249419015
카페 느와르	2773196.50270051
오래된 인력거	16229.925418033019
창피해	374246.05371719
양 한 마리, 양 두 마리	96976.44089384539

<k-fold cross validation>

과대적합(Overfitting)을 방지한다. Dataset을 training dataset과 validation dataset으로 반복적으로 나눈다. 해당 모델의 성능을 측정한다.

(2) K-fold cross validation

```
[29] from sklearn.model_selection import KFold
```

```
[30] k_fold = KFold(n_splits=5, shuffle=True, random_state=777)
```

```
[31] for train_idx, val_idx in k_fold.split(X_train):
         print(len(train_idx), len(val_idx))
         break
```

```
480 120
```

```
[32] model = lgb.LGBMRegressor(random_state=777, n_estimators=1000)

     models = []

     for train_idx, val_idx in k_fold.split(X_train):
         x_t = X_train.iloc[train_idx]
         y_t = y_train.iloc[train_idx]
         x_val = X_train.iloc[val_idx]
         y_val = y_train.iloc[val_idx]

         models.append(model.fit(x_t, y_t, eval_set=(x_val, y_val), early_stopping_rounds=100, verbose = 100))
```

```
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 2.70572e+12
Early stopping, best iteration is:
[6]     valid_0's l2: 2.45438e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 3.90847e+12
Early stopping, best iteration is:
[33]    valid_0's l2: 3.72825e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 3.50344e+12
Early stopping, best iteration is:
[8]     valid_0's l2: 2.58737e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 1.45977e+12
Early stopping, best iteration is:
[11]    valid_0's l2: 1.26226e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 1.77214e+12
Early stopping, best iteration is:
[22]    valid_0's l2: 1.57631e+12
```

```
[33] models
```

```
[LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=-1, num_leaves=31, objective=None,
               random_state=777, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0),
 LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=-1, num_leaves=31, objective=None,
               random_state=777, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0),
 LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=-1, num_leaves=31, objective=None,
               random_state=777, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0),
 LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=-1, num_leaves=31, objective=None,
               random_state=777, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0),
 LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=-1, num_leaves=31, objective=None,
               random_state=777, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)]
```

```
[34] preds = []
     for model in models:
         preds.append(model.predict(X_test))
     len(preds)
```

```
5
```

```
[35] kfoldLightGBM = submission.copy()
```

```
[36] import numpy as np
```

```
[37] kfoldLightGBM['box_off_num'] = np.mean(preds, axis = 0)
```

```
kfoldLightGBM.head()
```

	title	box_off_num
0	용서는 없다	3.367422e+06
1	아빠가 여자를 좋아해	9.611389e+05
2	하모니	1.097930e+06
3	의형제	2.097271e+06
4	평행 이론	7.814763e+05

```
kfoldLightGBM.to_csv('kfoldLightGBM.csv', index = False)
```

결과, 이렇게 예측되었다.

kfoldLightGBM.csv ✕

총 243개 항목 중 1~100

title	box_off_num
용서는 없다	3367422.0821102364
아빠가 여자를 좋아해	961138.8833701599
하모니	1097929.6785131267
의형제	2097270.8199992133
평행 이론	781476.3113104681
회오리 바람	123133.2339697665
경계도시 2	84085.0189624762
이웃집 남자	199222.9267030342
아마존의 눈물 극장판	124854.97973097453
반가운 살인자	1072684.7882064746
집 나온 남자들	657040.5576998391
베스트셀러	1644701.471607788
작은 연못	924735.7208061942
경	191699.58415836154
대한민국 1%	723218.3994875526
포화 속으로	2395613.428884621
나쁜 놈이 더 잘 잔다	113180.54906592457
맨발의 꿈	1882765.9281229642
요술	97286.47282982926
마법천자문: 대마왕의 부활을 막아라	417270.4211343128
인플루언스	161482.33675769935
미스터 좀비	63495.733631151365

프로젝트 2주차

5. 모델 학습 및 검증

(3) feature engineering

Feature engineering 은 머신러닝 알고리즘을 작동하기 위해 데이터에 대한 도메인 지식을 활용하여 특징(Feature)를 만들어내는 과정이다. 즉, 머신러닝 모델을 위한 데이터 테이블의 컬럼(특징)을 생성하거나 선택하는 작업을 의미한다. 가장 이상적인 입력 데이터는 부족하지도 과하지도 않은 정확한 정보만 포함될 때이다. 그러므로 충분한 데이터를 먼저 모으고 어떤 feature 가 유용한지 아닌지 확인하는 과정을 거친다(특징 선택(feature selection) 또는 특징 추출(feature extraction), 차원감소(dimension reduction)). 해당 과정은 기존 입력을 토대로 새로운 입력 데이터를 만들기 때문에 보통 learning 과정 전에 실행된다.

```
[69]  features

      ['time', 'dir_prev_num', 'num_staff', 'num_actor']
```

```
[70]  train.columns

      Index(['title', 'distributor', 'genre', 'release_time', 'time',
             'screening_rat', 'director', 'dir_prev_bfnum', 'dir_prev_num',
             'num_staff', 'num_actor', 'box_off_num'],
            dtype='object')
```

```
[71]  train.genre

      0        액션
      1       느와르
      2        액션
      3       코미디
      4       코미디
             ...
      595     드라마
      596     드라마
      597      공포
      598     느와르
      599      액션
      Name: genre, Length: 600, dtype: object
```

```
[73]  from sklearn import preprocessing
      le=preprocessing.LabelEncoder()
      train['genre']=le.fit_transform(train['genre'])
```

[38]
```
train['genre']
```

```
0      10
1       2
2      10
3      11
4      11
       ..
595     4
596     4
597     1
598     2
599    10
Name: genre, Length: 600, dtype: int64
```

[39]
```
test['genre']=le.transform(test['genre'])
```

[40]
```
features=['time', 'dir_prev_num', 'num_staff', 'num_actor', 'dir_prev_bfnum', 'genre']
```

[41]
```
X_train, X_test, y_train = train[features], test[features], train[target]
```

[42]
```
model = lgb.LGBMRegressor(random_state=777, n_estimators=1000)

models = []

for train_idx, val_idx in k_fold.split(X_train):
    x_t = X_train.iloc[train_idx]
    y_t = y_train.iloc[train_idx]
    x_val = X_train.iloc[val_idx]
    y_val = y_train.iloc[val_idx]

    models.append(model.fit(x_t, y_t, eval_set=(x_val, y_val), early_stopping_rounds=100, verbose = 100))
```

```
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 2.62067e+12
Early stopping, best iteration is:
[9]     valid_0's l2: 2.42668e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 4.39227e+12
Early stopping, best iteration is:
[23]    valid_0's l2: 3.97173e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 3.29841e+12
Early stopping, best iteration is:
[10]    valid_0's l2: 2.53643e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 1.56499e+12
Early stopping, best iteration is:
[16]    valid_0's l2: 1.21201e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 1.60118e+12
Early stopping, best iteration is:
[29]    valid_0's l2: 1.47528e+12
```

[48]
```
X_test.head()
```

	time	dir_prev_num	num_staff	num_actor	dir_prev_bfnum	genre
0	125	2	304	3	3.005290e+05	2
1	113	4	275	3	3.427002e+05	5
2	115	3	419	7	4.206611e+06	4
3	116	2	408	2	6.913420e+05	10
4	110	1	380	1	3.173800e+04	1

```
preds = []
for model in models:
    preds.append(model.predict(X_test))
len(preds)
```

```
5
```

```
[50] feLightGBM = submission.copy()
```

```
[51] feLightGBM['box_off_num'] = np.mean(preds, axis = 0)
```

```
[53] feLightGBM.to_csv('feLightGBM.csv', index = False)
```

결과는 아래와 같다(윗부분만 첨부)

feLightGBM.csv

총 243개 항목 중 1~100 필터

title	box_off_num
용서는 없다	3395492.6833621534
아빠가 여자를 좋아해	823543.8619063471
하모니	1162055.3950630578
의형제	2184689.060704529
평행 이론	809328.7816518641
회오리 바람	74073.45796993117
경계도시 2	142544.45134581588
이웃집 남자	96480.98505297104
아마존의 눈물 극장판	5744.8780552960925
반가운 살인자	732617.6089669269
집 나온 남자들	608605.7200069077
베스트셀러	1765804.1699847379
작은 연못	1053202.9832343853
경	25749.97479361035
대한민국 1%	539060.3271928253
포화 속으로	2308436.750526924
나쁜 놈이 더 잘 잔다	35221.96031735892
맨발의 꿈	2096111.3914733878
요술	71737.90451829384
마법천자문: 대마왕의 부활을 막아라	302655.1306580895
인플루언스	-12212.64945828141
미스터 좀비	117094.31198743661
김복남 살인사건의 전말	746727.7801060962
엄지아빠	183796.99826904727
탈주	163983.82345340913
해결사	1600657.11945412
할	79968.25679136766
된장	1358571.9408876954
조금만 더 가까이	280178.51227549027
우리 만난 적 있나요	126540.46264248571
쩨쩨한 로맨스	1951921.6424078573

(4) Grid Search

모델 하이퍼 파라미터에 넣을 수 있는 값들을 순차적으로 입력한 뒤에 가장 높은 성능을 보이는 하이퍼 파라미터들을 찾는 탐색 방법. (하이퍼 파라미터 : 모델을 생성할 때 사용자가 직접 설정하는 변수, 파라미터 : 학습 과정에서 생성되는 변수) 그리드 서치를 하는 이유는 가장 우수한 성능을 보이는 모델의 하이퍼 파라미터를 찾기 위해서이다. best_estimator_를 사용하면 학습 과정에서 하이퍼 파라미터 조정으로 가장 좋은 성능을 보인 모델이 반환된다. 단점으로 시간이 오래 걸린다. min_child_sampels : 최종 결정 클래스인 Leaf node가 되기 위해서 필요한 최소 데이터 개체의 수. neg_mean_squred_err : 오차가 얼마나 나는지.

```
[37] from sklearn.model_selection import GridSearchCV
```

```
[38] model = lgb.LGBMRegressor(random_state=777, n_estimators=1000)


     params = {
         'learning_rate': [0.1, 0.01, 0.003],
         'min_child_samples': [20, 30]}

     gs = GridSearchCV(estimator=model,
                param_grid=params,
                scoring='neg_mean_squared_error',
                cv = k_fold)
```

```
[39] gs.fit(X_train, y_train)
```

```
     GridSearchCV(cv=KFold(n_splits=5, random_state=777, shuffle=True),
                  error_score=nan,
                  estimator=LGBMRegressor(boosting_type='gbdt', class_weight=None,
                                          colsample_bytree=1.0,
                                          importance_type='split', learning_rate=0.1,
                                          max_depth=-1, min_child_samples=20,
                                          min_child_weight=0.001, min_split_gain=0.0,
                                          n_estimators=1000, n_jobs=-1,
                                          num_leaves=31, objective=None,
                                          random_state=777, reg_alpha=0.0,
                                          reg_lambda=0.0, silent=True, subsample=1.0,
                                          subsample_for_bin=200000,
                                          subsample_freq=0),
                  iid='deprecated', n_jobs=None,
                  param_grid={'learning_rate': [0.1, 0.01, 0.003],
                              'min_child_samples': [20, 30]},
                  pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                  scoring='neg_mean_squared_error', verbose=0)
```

```
[40] gs.best_params_
```

```
     {'learning_rate': 0.003, 'min_child_samples': 30}
```

```
model = lgb.LGBMRegressor(random_state=777, n_estimators=1000, learning_rate= 0.003, min_child_samples=30)

models = []

for train_idx, val_idx in k_fold.split(X_train):
    x_t = X_train.iloc[train_idx]
    y_t = y_train.iloc[train_idx]
    x_val = X_train.iloc[val_idx]
    y_val = y_train.iloc[val_idx]

    models.append(model.fit(x_t, y_t, eval_set=(x_val, y_val), early_stopping_rounds=100, verbose = 100))
```

```
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 2.56673e+12
[200]   valid_0's l2: 2.45583e+12
[300]   valid_0's l2: 2.42575e+12
[400]   valid_0's l2: 2.43392e+12
Early stopping, best iteration is:
[335]   valid_0's l2: 2.42348e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 4.89194e+12
[200]   valid_0's l2: 4.40922e+12
[300]   valid_0's l2: 4.19146e+12
[400]   valid_0's l2: 4.05951e+12
[500]   valid_0's l2: 3.96931e+12
[600]   valid_0's l2: 3.91727e+12
[700]   valid_0's l2: 3.88462e+12
[800]   valid_0's l2: 3.87695e+12
[900]   valid_0's l2: 3.87088e+12
Early stopping, best iteration is:
[876]   valid_0's l2: 3.87035e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 3.14361e+12
[200]   valid_0's l2: 2.79286e+12
[300]   valid_0's l2: 2.59302e+12
[400]   valid_0's l2: 2.47608e+12
[500]   valid_0's l2: 2.40386e+12
[600]   valid_0's l2: 2.36407e+12
[700]   valid_0's l2: 2.38505e+12
Early stopping, best iteration is:
[647]   valid_0's l2: 2.35685e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 1.60592e+12
[200]   valid_0's l2: 1.40227e+12
[300]   valid_0's l2: 1.30053e+12
[400]   valid_0's l2: 1.25184e+12
[500]   valid_0's l2: 1.23543e+12
[600]   valid_0's l2: 1.23595e+12
Early stopping, best iteration is:
[536]   valid_0's l2: 1.23368e+12
Training until validation scores don't improve for 100 rounds.
[100]   valid_0's l2: 1.96107e+12
[200]   valid_0's l2: 1.75478e+12
[300]   valid_0's l2: 1.64513e+12
[400]   valid_0's l2: 1.58132e+12
[500]   valid_0's l2: 1.54801e+12
[600]   valid_0's l2: 1.52159e+12
[700]   valid_0's l2: 1.50655e+12
[800]   valid_0's l2: 1.49834e+12
[900]   valid_0's l2: 1.50018e+12
Early stopping, best iteration is:
[873]   valid_0's l2: 1.49621e+12
```

```
[42]  preds = []
      for model in models:
          preds.append(model.predict(X_test))
```

```
[43]  gs.best_score_
```

```
-2334525343085.6494
```

```
[44]  gslgbm=submission.copy()
```

```
[45]  import numpy as np
```

```
[47]  gslgbm['box of num']=np.mean(preds, axis=0)
```

```
[30]  X_test.head()
```

	time	dir_prev_num	num_staff	num_actor	dir_prev_bfnum	genre
0	125	2	304	3	3.005290e+05	2
1	113	4	275	3	3.427002e+05	5
2	115	3	419	7	4.206611e+06	4
3	116	2	408	2	6.913420e+05	10
4	110	1	380	1	3.173800e+04	1

```
[48]  gslgbm.to_csv('gslgbm.csv', index = False)
```

결과는 아래와 같다.(윗부분만 첨부)

gslgbm.csv

총 243개 항목 중 1~100 필터

title	box_off_num	box of num
용서는 없다	0	2974959.748625541
아빠가 여자를 좋아해	0	982313.148006434
하모니	0	1283210.4005761403
의형제	0	1681758.4928494357
평행 이론	0	909584.4964290311
회오리 바람	0	80883.66589124226
경계도시 2	0	76793.30780519867
이웃집 남자	0	69919.0727616737
아마존의 눈물 극장판	0	28379.757955412126
반가운 살인자	0	824868.6575766597
집 나온 남자들	0	546360.0476170096
베스트셀러	0	1709895.552216495
작은 연못	0	1201415.9101280058
경	0	132466.2992459081
대한민국 1%	0	1032903.3405147765
포화 속으로	0	1999595.684219319
나쁜 놈이 더 잘 잔다	0	44694.587598126695
맨발의 꿈	0	2187616.8180050296
요술	0	61286.94684276744
마법전자문: 대마왕의 부활을 막아라	0	242944.04391478616
인플루언스	0	128618.08860033753
미스터 좀비	0	236832.86490207948
김복남 살인사건의 전말	0	882223.501722675
엄지아빠	0	150256.41018946373
탈주	0	170638.80179603127
해결사	0	1421937.163529749
할	0	79456.02587312998
된장	0	1169584.705865378
조금만 더 가까이	0	272819.112055928
우리 만난 적 있나요	0	109980.53674776231
쩨쩨한 로맨스	0	1773325.772992485
용서	0	82701.49998398831
브라보! 재즈 라이프	0	202486.76010937413
카페 느와르	0	1614252.3080419085
오래된 인력거	0	74266.42055487669
창피해	0	389457.89406999084
양 한 마리, 양 두 마리	0	83855.1888024103

6. 결과 정리

정확도는 lightGBM<kfold cross validation lightGBM<feature engineering<Grid Search라고 생각했으나 feature engineering과 lightGBM에서 마이너스의 관객 수가 예측되었다. Kfold cross validation과 Grid Search는 마이너스의 관객 수가 예측되지는 않았지만 전체적으로 Grid Search에서 더 극단적인 값이 예측되었다. 다음주에 LSTM을 이용해서 관객 수를 예측해보고 위의 4가지와 비교해 볼 생각이다.