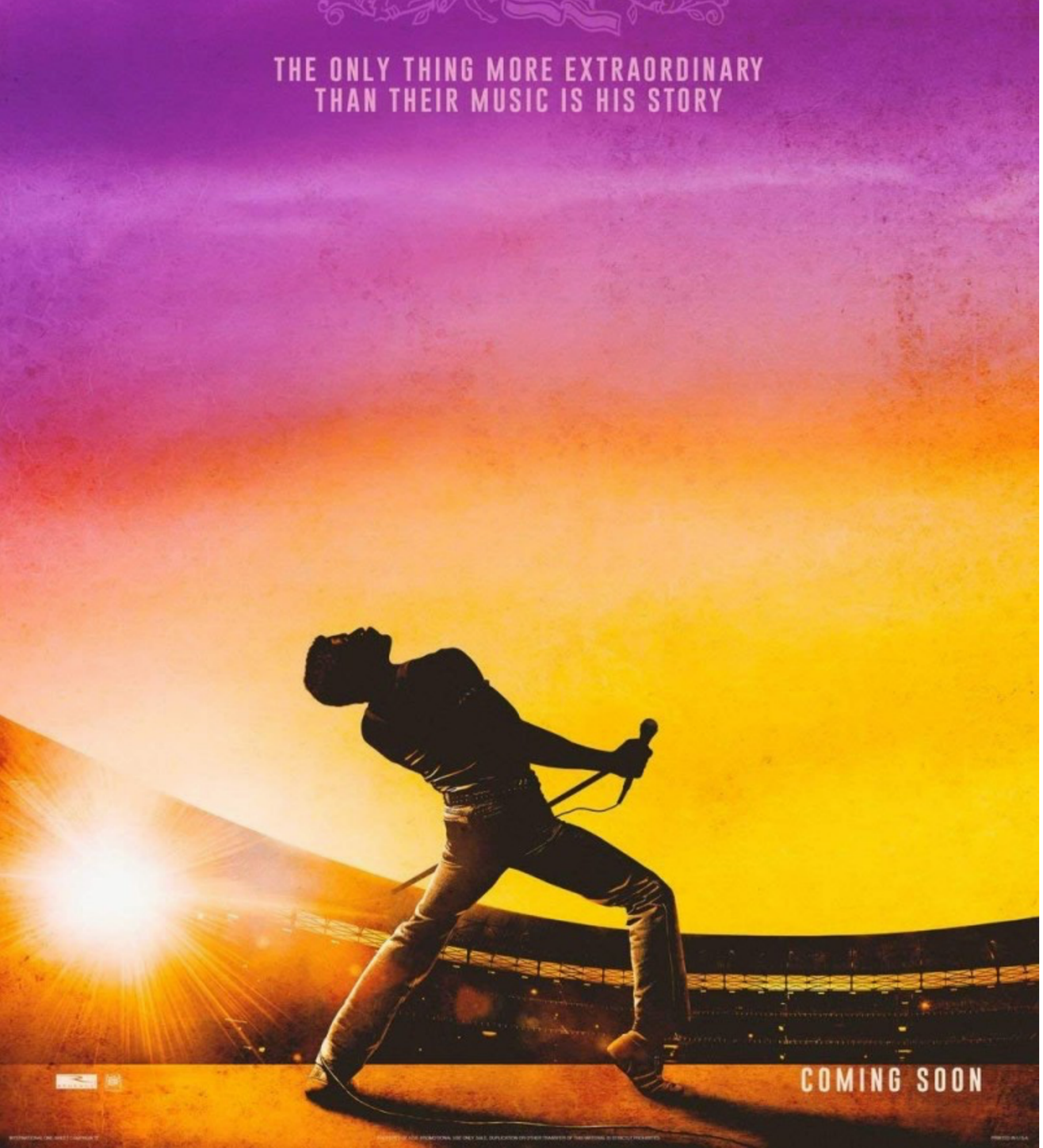


음원 순위 예측

중간보고

강동원 백원희 유건욱 이청파 오태환

Contents

- Introduction
- Literature review
- Method
- Results
- Plan

Introduction

- 음악 산업 시장은 이미 포화 되어 있음에도 불구하고 2016년부터 2018년까지 매출액이 꾸준하게 증가하고 있다 (문화체육관광부, 2019).
- 이에 따른 부가 가치 창출, 해외 진출 등으로 앞으로 발전 가능성이 더 클 것으로 기대 된다 (추승엽 외, 2016).
- 음원 시장은 시대의 특성이 반영되어 기존의 음반 산업과 다르게 다양한 변수에 의해 영향을 받는 시장으로 변화 하였다 (추승엽 외, 2016).
- 또한 음원 시장은 유투브, SNS, 인터넷 사이트 등 다양한 플랫폼의 영향이 커지는 양상을 보이고 있다 (김가연 외, 2018).
- 그러므로 쏟아져 나오는 음원의 홍수 속에서 대중들에게 기억되는 음원을 만들어 내기 위한 방법으로, 순위를 예측하는 시스템을 개발하여 상업적으로 연결 되어져야 한다.
- 더불어 이번 프로젝트의 음원 순위 예측을 통해 음원 시장의 사재기의 난타전에서 (곽아영, 2016) 음원 사재기 방지의 객관적인 지표를 마련해 볼수 있기를 기대해 본다.

Literature Review

음원 순위에 영향을 미치는 요인

- 음원 발매 시기, 조성, BPM, 음악 스타일, OST, 런닝 타임 (Chung el al., 2018)
- 가수 타입, 가수 성별, 운율, 코러스 (Chung el al., 2018; 김가연 외, 2018)
- 신문 기사 등의 홍보 (하정처 , 2016; 김가연 외, 2018)
- DC 인사이드 갤러리의 평균 게시글 수, SNS (김가연 외, 2018)

Method

Outcome variable

- 가온차트의 주간 차트의 순위(1위부터 100위 or 200위)를 rank 1 부터 rank 10 까지 카테코리화
- 기간: 2018년 4월 22일 ~ 2020년 4월 19일, 총 106 주간
- Observation: 17,363

Method

Independent variable

- 가수 관련 변수: 성별 (남, 여, 혼성), 활동 (솔로, 듀엣, 그룹, 프로젝트, 밴드)
- 노래 관련 변수: 런닝 타임, 장르 (댄스, 랩/힙합,R&B/소울, 락 등), 타입 (K-POP, POP, OST)
- 콘텐츠 관련 변수: Naver trend score, Google trend score, DC 갤러리 평균 수, You tube 조회수, OST의 프로그램 시청률
- 기타 변수: 계절, 소속사 trend score
- 파생변수: 주간 순위 안에서 가수의 이전 곡 랭킹, 계절에 따른 장르의 순위

Method

Data crawling

- 가온차트 음원 순위
- 지니 곡 정보
- 네이버 트렌드
- 구글 트렌드
- DC 갤러리 평균 게시글 수
- You tube 조회수
- OST 시청률

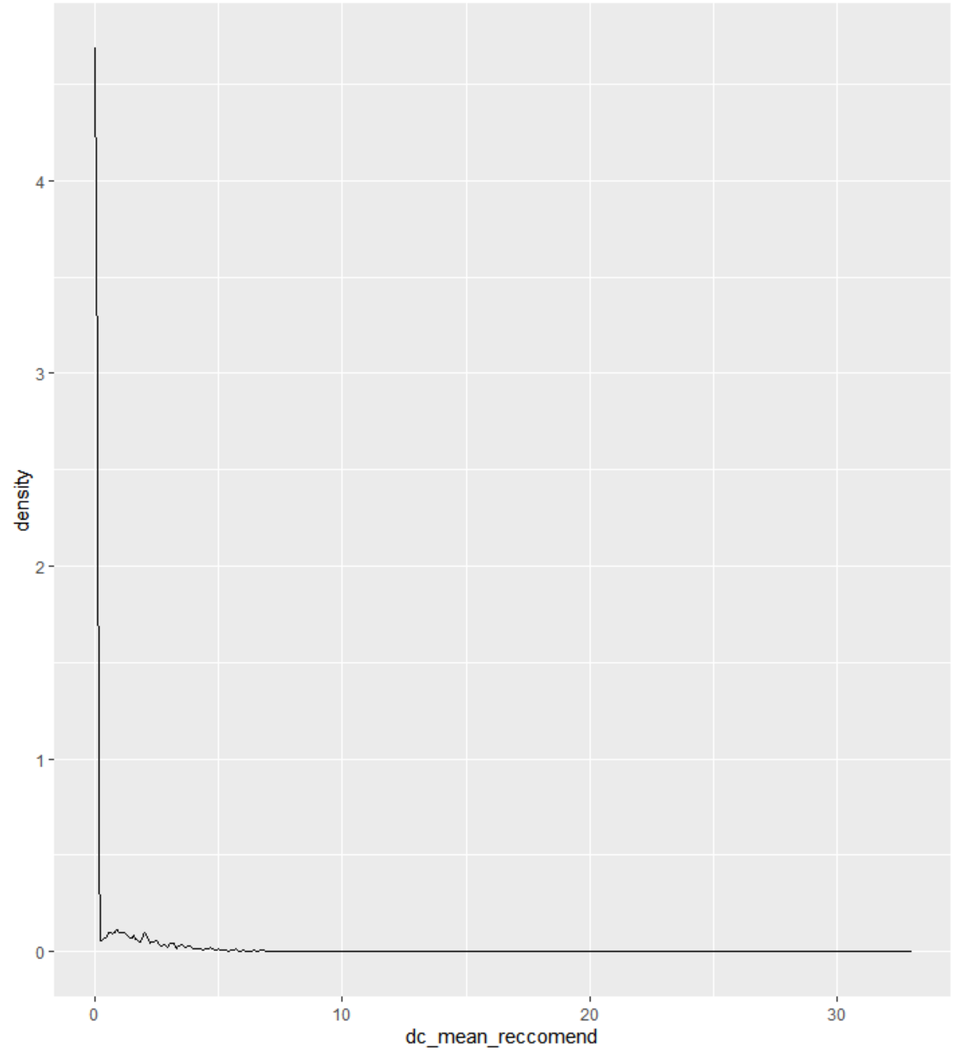
Method

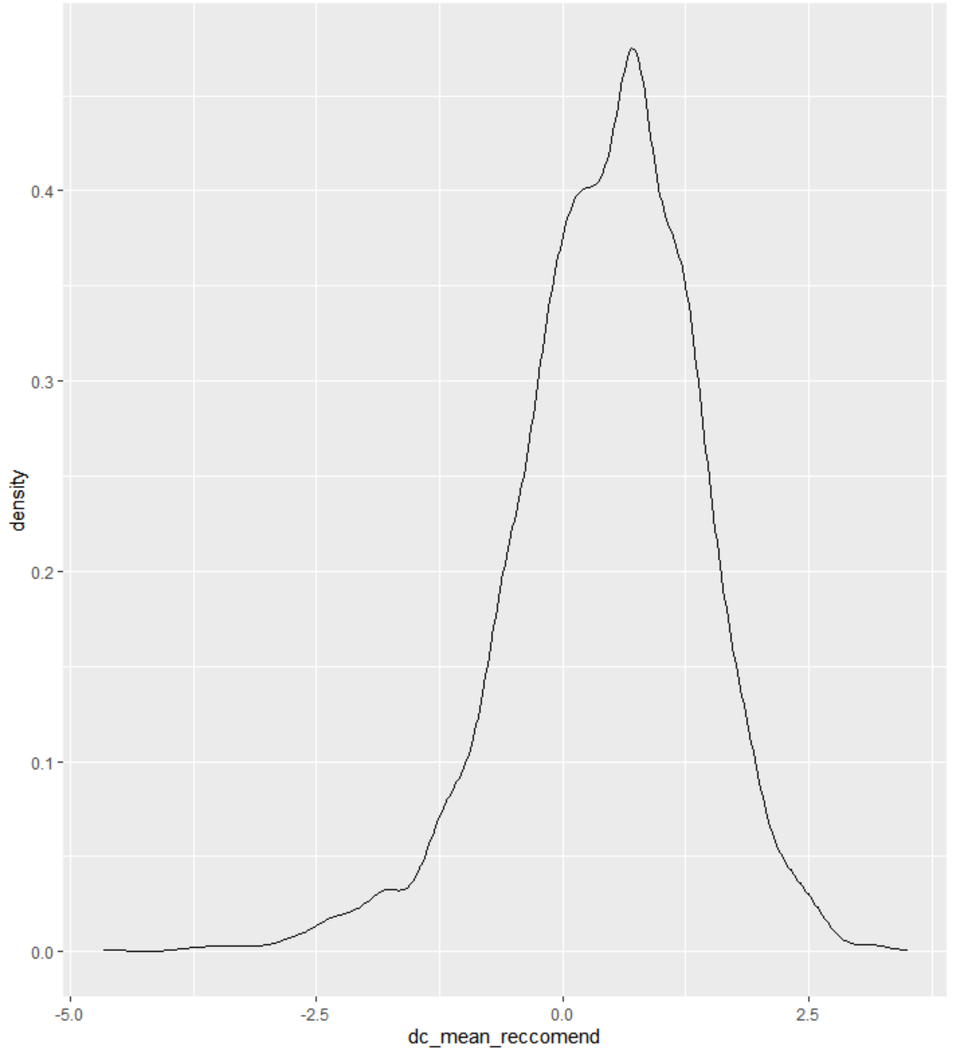
Statistical Method

- EDA
- Mean max scaling
- Log transformation
- Logistic regression
- Random forest
- XG boost
- Cat boost

Results_*EDA*

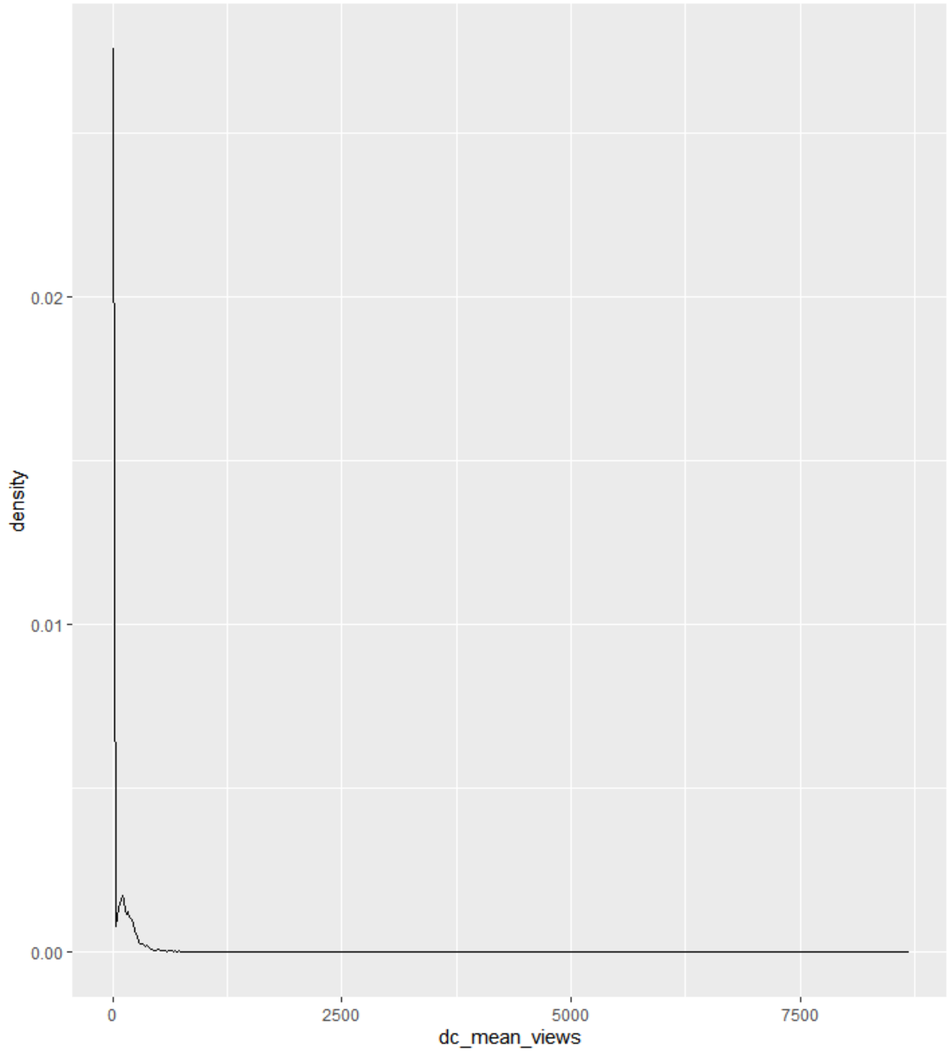
DC mean recommend 변환 전 후

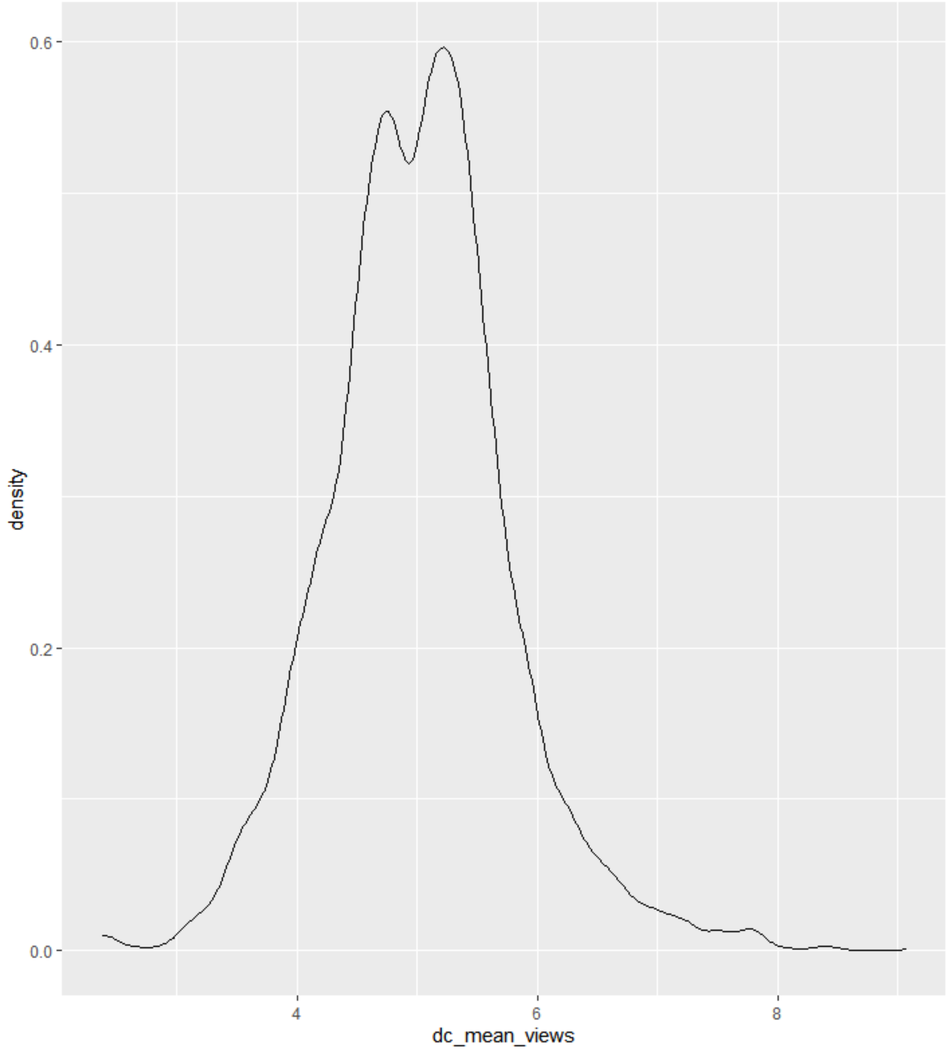




Results_*EDA*

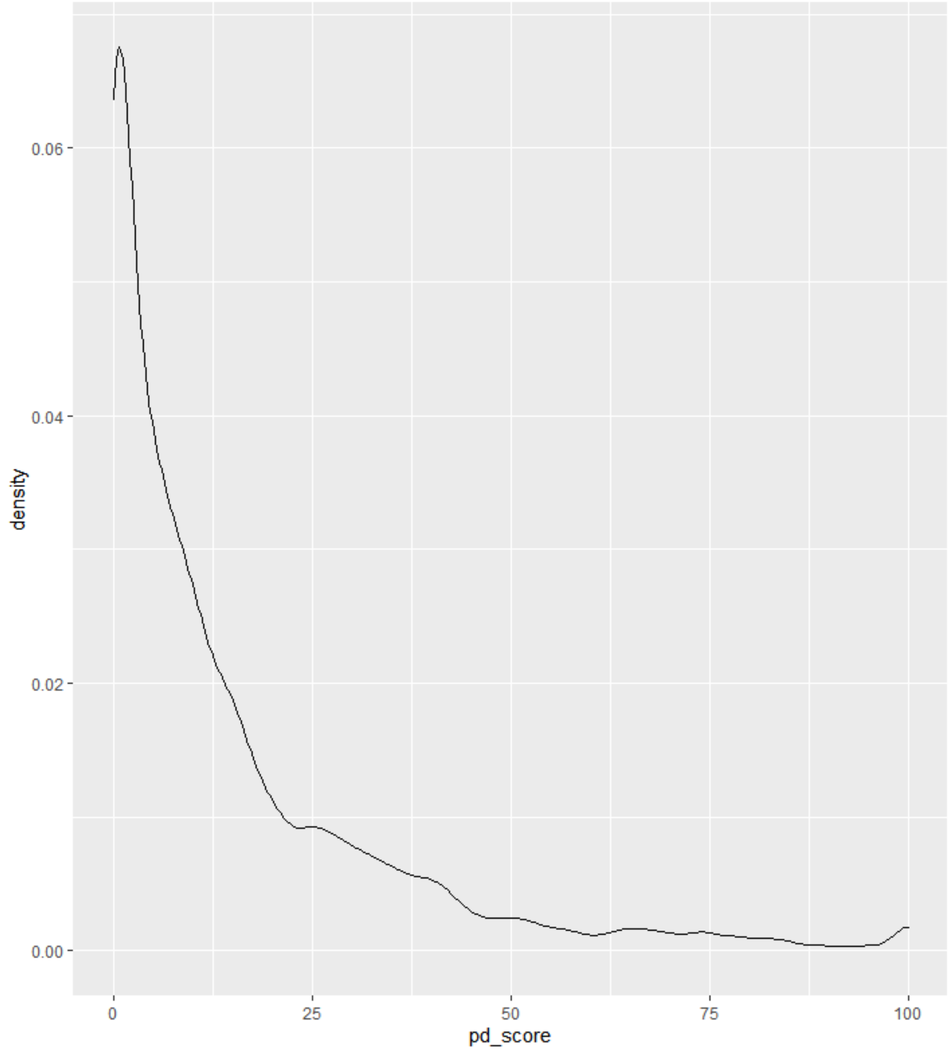
DC total 변환 전 후

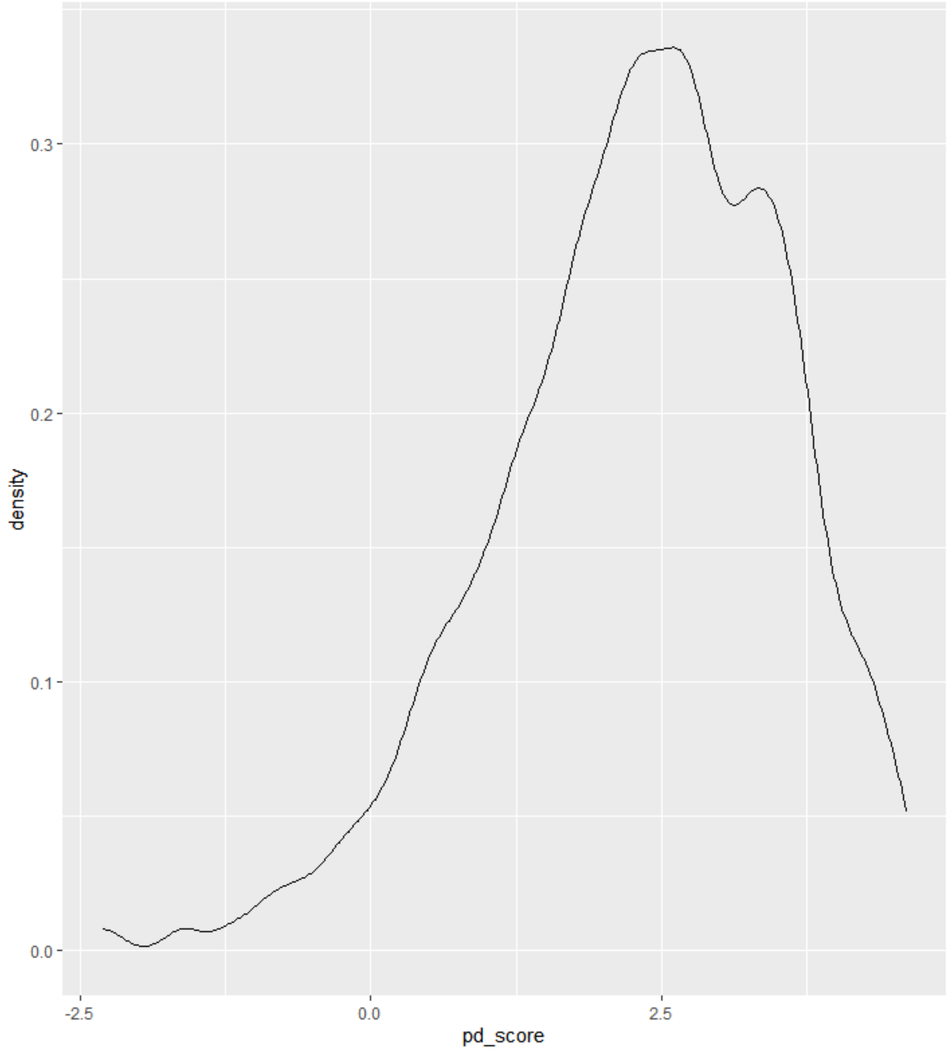




Results_*EDA*

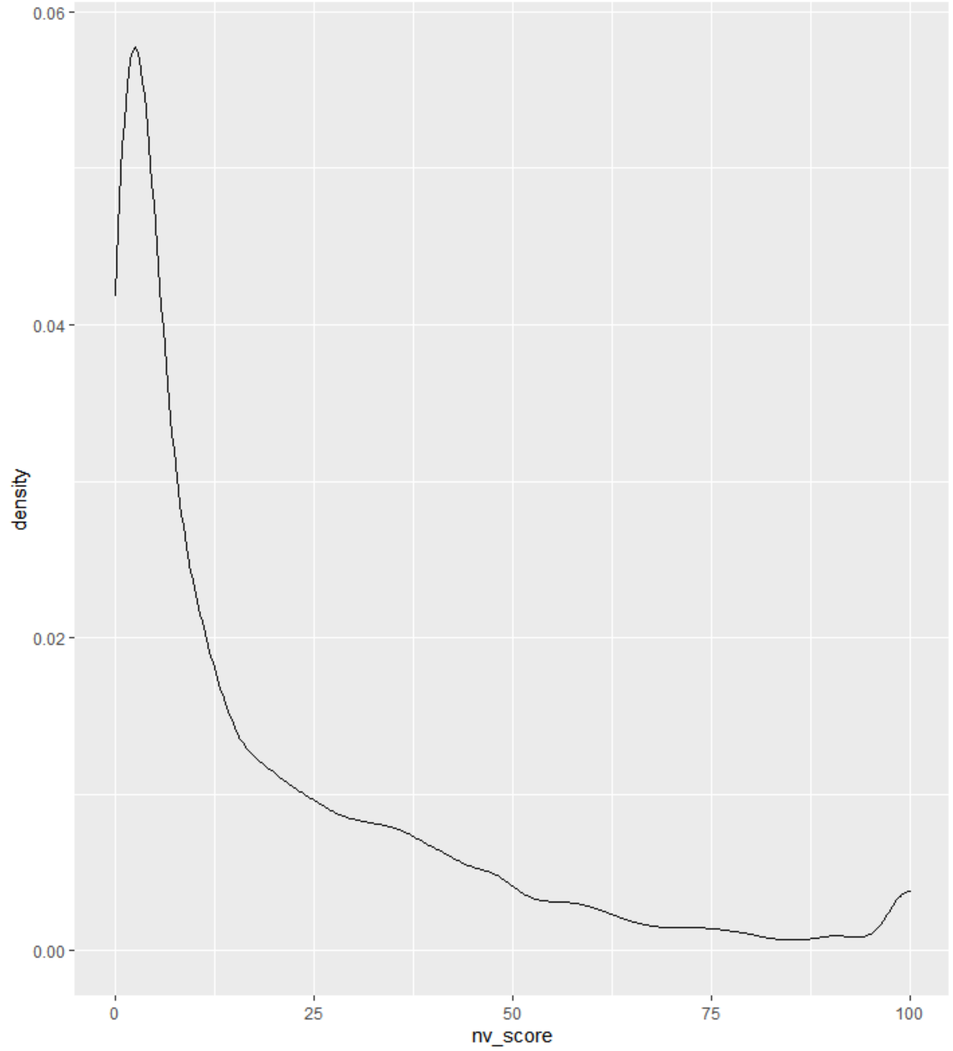
소속사 trend score 변환 전 후

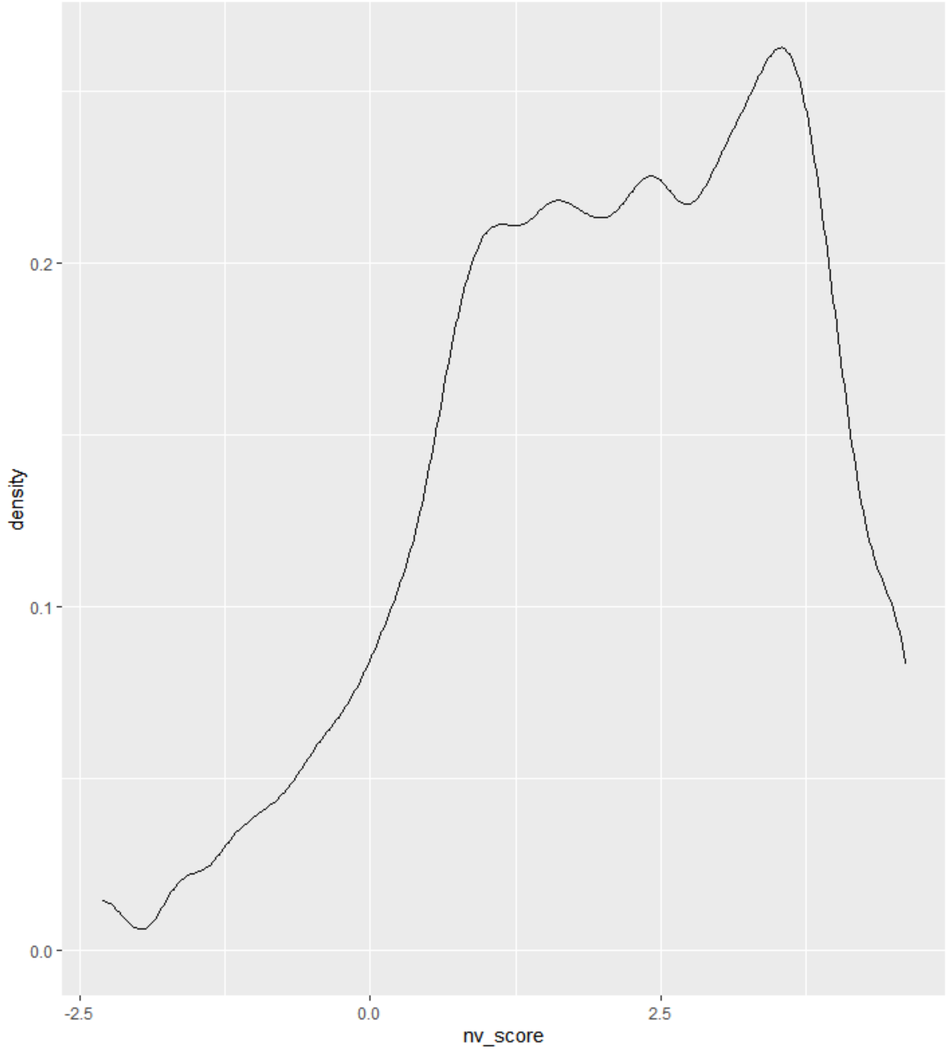




Results_*EDA*

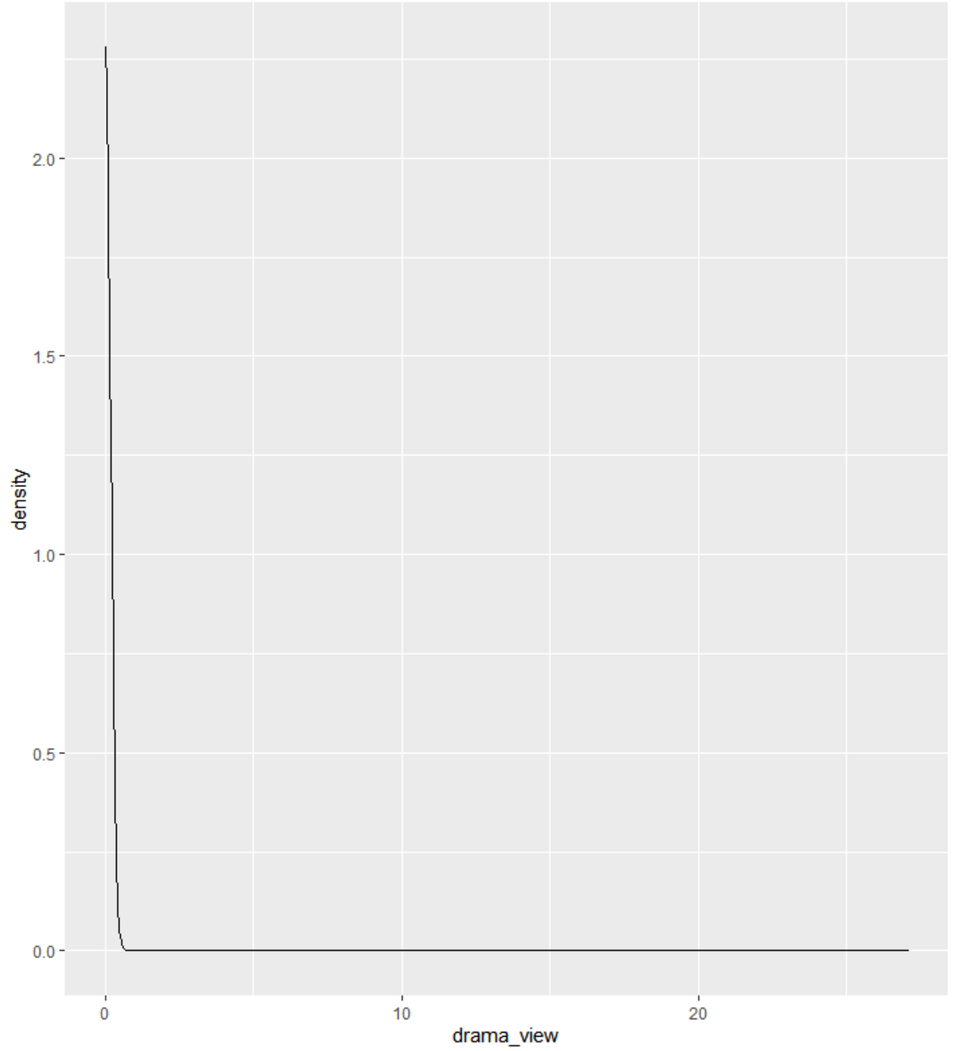
네이버 trend score 변환 전 후

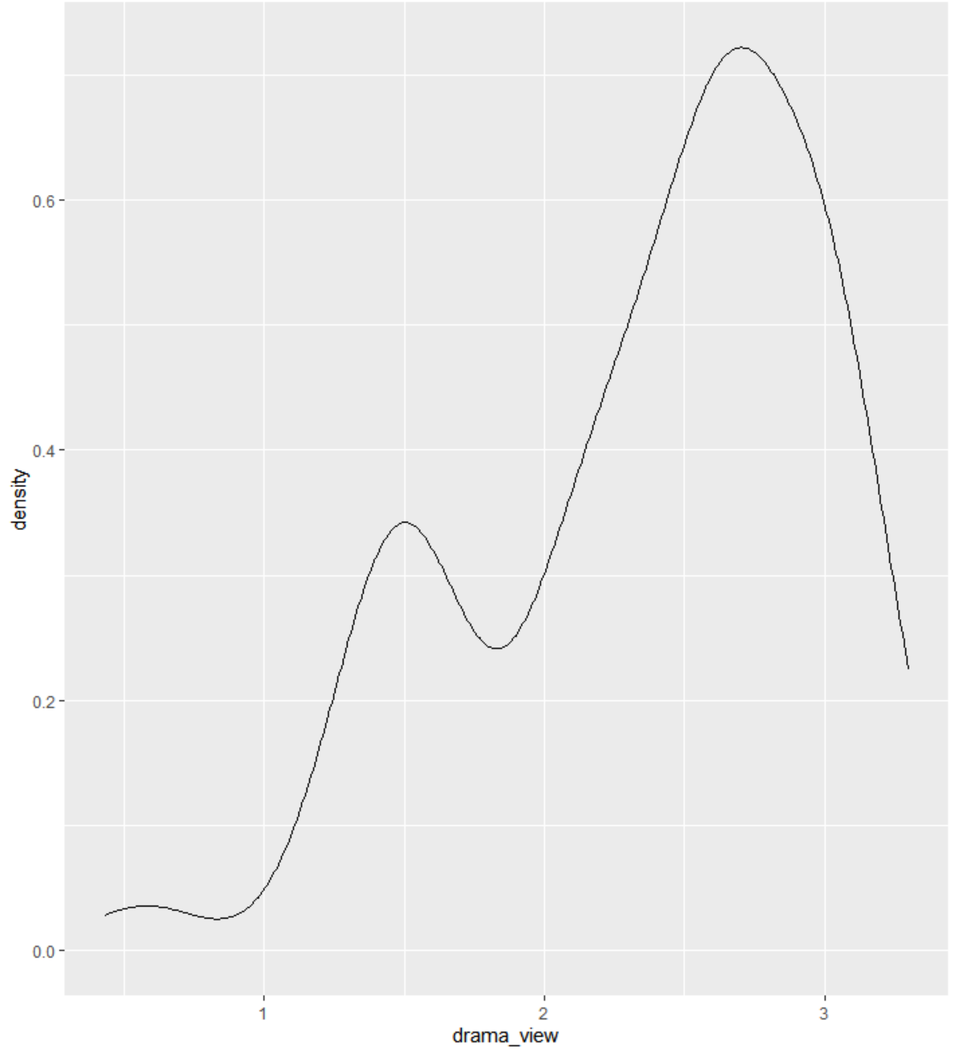




Results_*EDA*

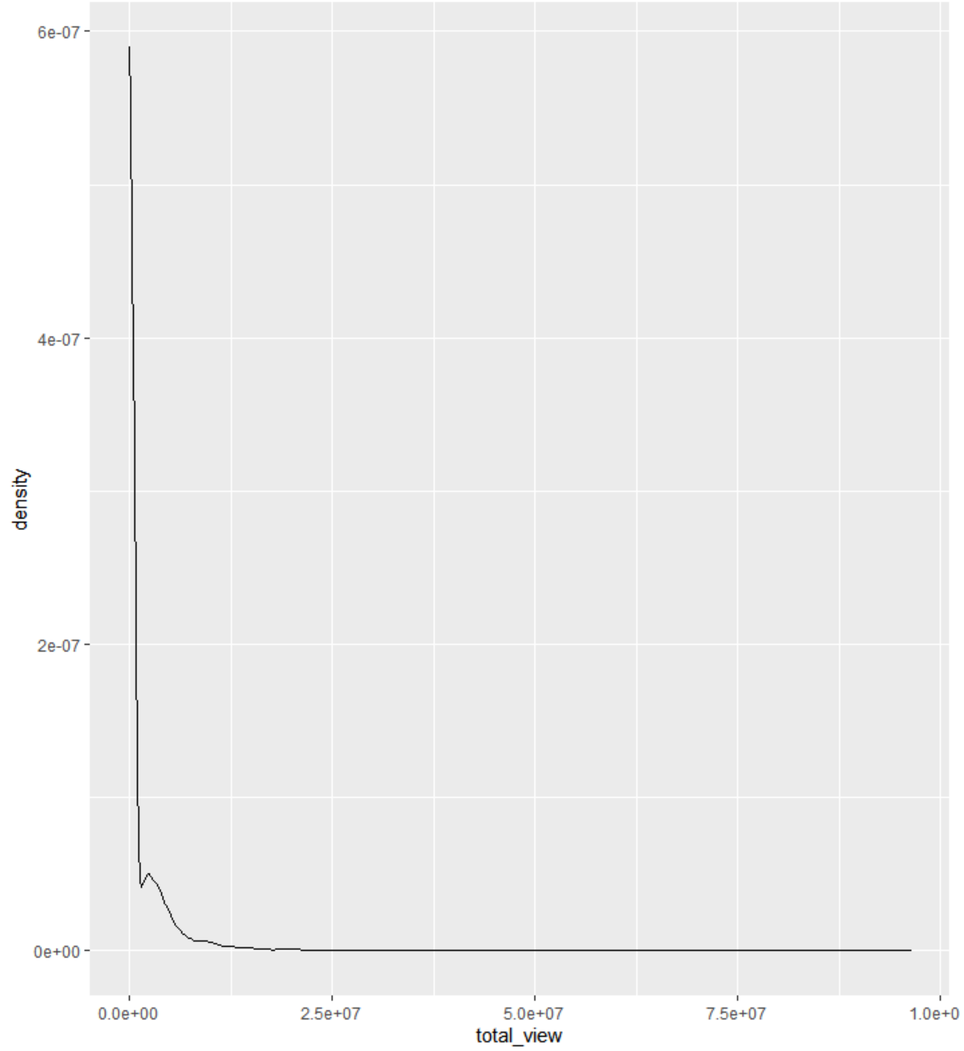
드라마 시청률 변환 전 후

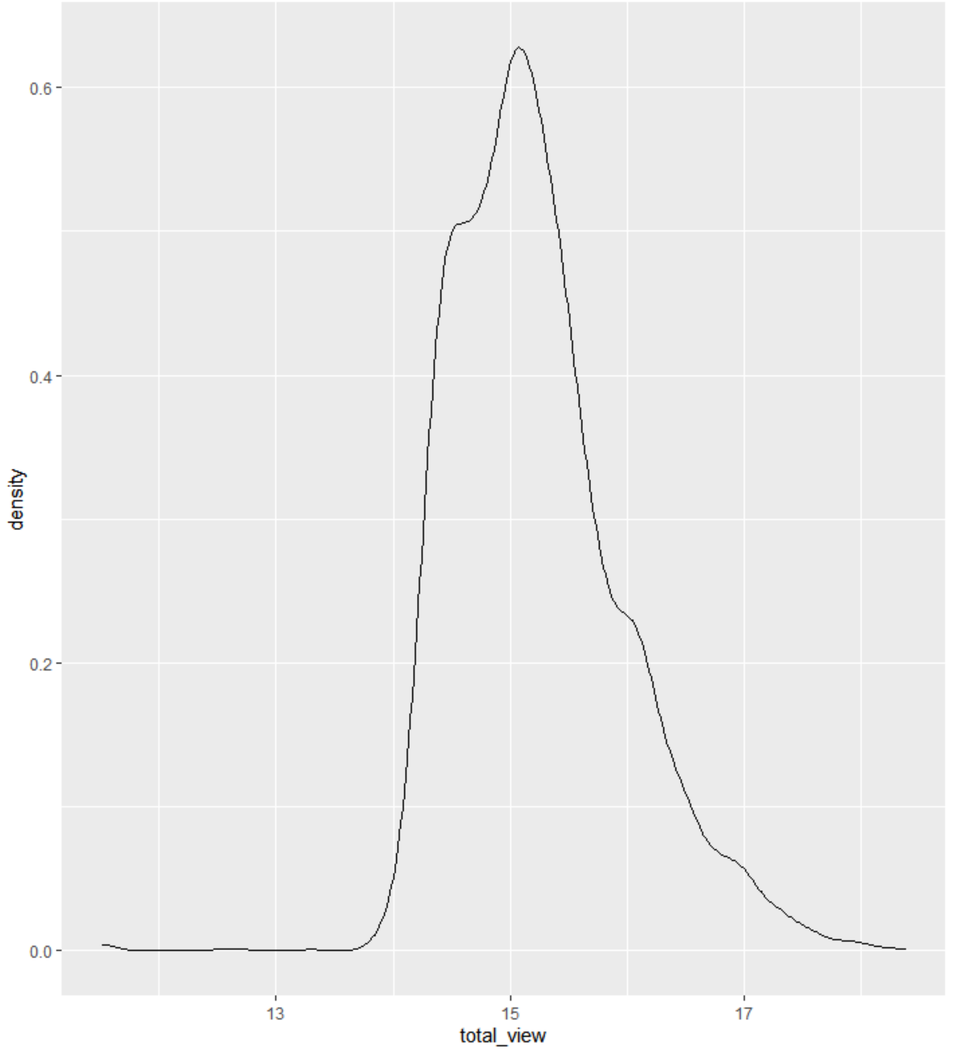




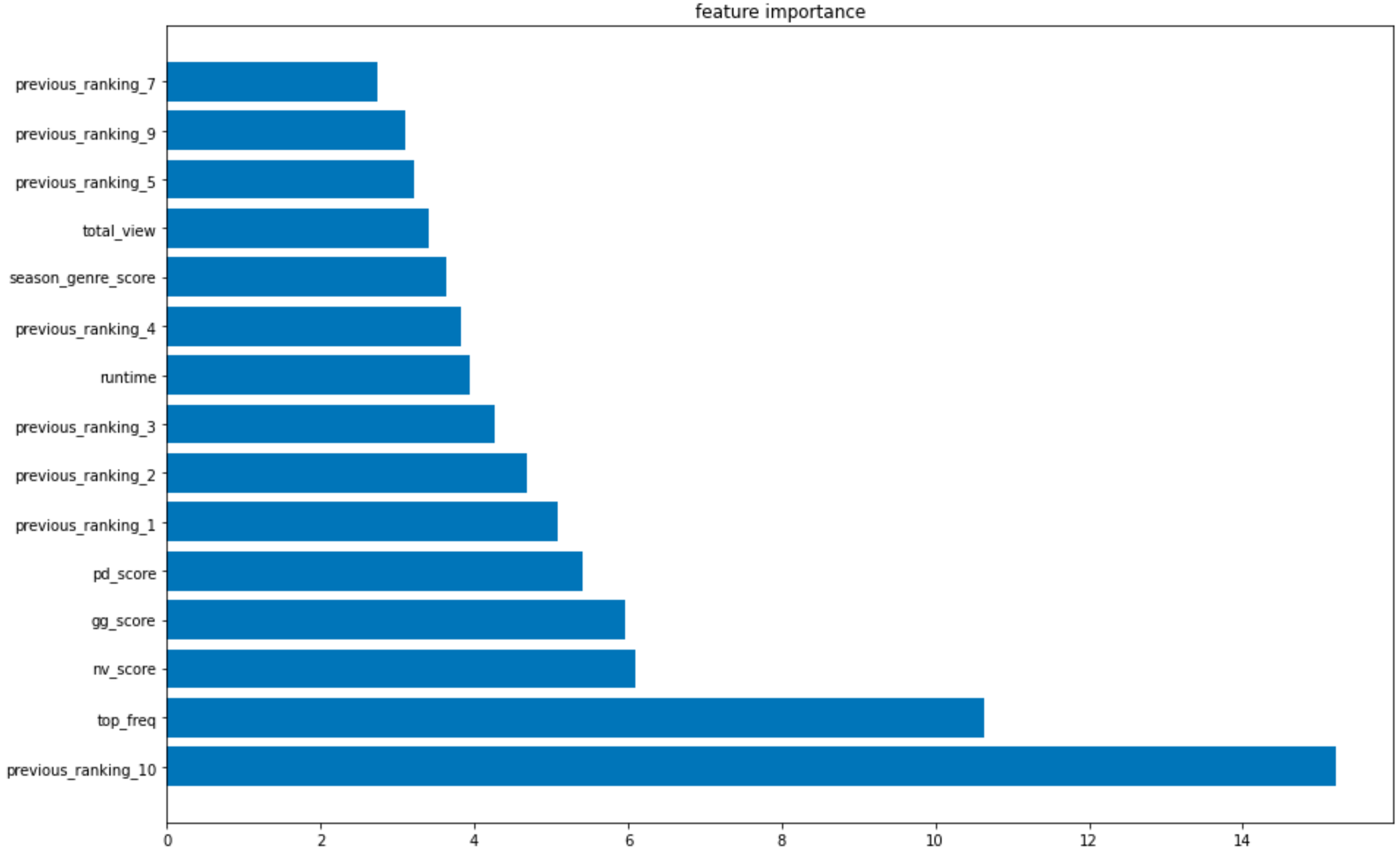
Results_*EDA*

total view 변환 전 후





Results_*Feature importance*



Results

cat boost

Catboost

In [386]:
```
from catboost import CatBoostClassifier
```

In [387]:
```
model = CatBoostClassifier(n_estimators= 1000,learning_rate = 0.05)
```

In [388]:
```
score = cross_val_score(model, X_train, y_train, cv = stf, scoring = 'f1_micro')
```

...

In [389]:
```
score
```

```
array([0.71990172, 0.71123872, 0.70935961, 0.70336894, 0.69983553,
       0.70263591, 0.72300082, 0.69884488, 0.71617162, 0.7169967 ])
```

In [390]:
```
score.mean()
```

```
0.7101354452463009
```

In [391]:
```
model.fit(X_train,y_train)
```

...

In [393]:
```
pred_y = model.predict(X_test)
print("F1 : %.3f" % f1_score(y_test, pred_y, average = 'micro'))
```

```
F1 : 0.714
```

Results

xg boost

In [345]:
```
from xgboost import XGBClassifier
```

In [346]:
```
# XGB는 생각보다 성능이 낮다... 아마 파라미터 조정을 해야될듯...
model = XGBClassifier(n_estimators = 500, learning_rate = 0.05)
```

In [347]:
```
score = cross_val_score(model, X_train, y_train, cv = stf, scoring = 'f1_micro')
```

In [348]:
```
score.mean()
```

```
0.6941777404628138
```

In [349]:
```
score
```

```
array([0.7018837 , 0.69155045, 0.6863711 , 0.69926048, 0.68256579,
       0.67710049, 0.70898599, 0.6790429 , 0.71039604, 0.70462046])
```

In [350]:
```
model.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.05, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=500, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [351]:
```
pred_y = model.predict(X_test)
print("F1 : %.3f" % f1_score(y_test, pred_y, average = 'micro'))
```

```
F1 : 0.700
```

Results

Logistic regression

Logistic Regression

In [363]:
```
model = LogisticRegression(random_state=1)
```

In [368]:
```
score = cross_val_score(model, X_train, y_train, cv = stf, scoring = 'f1_micro')
```

...

In [370]:
```
score.mean()
```

```
0.7001788141248058
```

In [364]:
```
model.fit(X_train,y_train)
```

```
/Users/gunwook/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver w
ll be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/gunwook/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default multi_c
ss will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)


LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=1, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [365]:
```
pred_y = model.predict(X_test)
```

In [366]:
```
from sklearn.metrics import precision_score, recall_score, f1_score
```

In [367]:
```
print("F1 : %.3f" % f1_score(y_test, pred_y, average = 'micro'))
```

```
F1 : 0.700
```

Results

Random forest

RandomForest

In [74]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [336]:
```
model = RandomForestClassifier(random_state =1, n_estimators = 500)
```

In [161]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
```

In [337]:
```
stf = StratifiedKFold(n_splits = 10, shuffle = True, random_state = 1)
```

In [338]:
```
model = RandomForestClassifier(random_state =1, n_estimators = 500)
score = cross_val_score(model, X_train, y_train, cv = stf, scoring = 'f1_micro')
```

In [340]:
```
score
```

```
array([0.6994267 , 0.68744873, 0.67898194, 0.68364832, 0.67269737,
       0.68121911, 0.69579555, 0.65676568, 0.69636964, 0.68729373])
```

In [341]:
```
score.mean()
```

```
0.6839646750957066
```

In [342]:
```
model.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=500,
                       n_jobs=None, oob_score=False, random_state=1, verbose=0,
                       warm_start=False)
```

In [343]:
```
pred_y = model.predict(X_test)
```

In [344]:
```
print("F1 : %.3f" % f1_score(y_test, pred_y, average = 'micro'))
```

```
F1 : 0.691
```

Plan

- Shiny 웹 개발


Thank you!