

The logo for The Movie Database (TMDB) is centered in the upper half of the image. It consists of the words "THE", "MOVIE", and "DB" stacked vertically in a bold, green, sans-serif font. The text is enclosed within a white rounded square with a thick green border. The background of the entire slide is a light beige color with a faint, stylized illustration of film reels and film strips winding across it.

**THE  
MOVIE  
DB**

**Box Office Prediction**



# Content

## 1. Data Dictionary

## 2. Pre-Processing

## 3. Modeling

- 선형회귀
- 다항회귀
- SVR
- KNN
- 랜덤포레스트

## 4. Result



# 1 Data Dictionary y



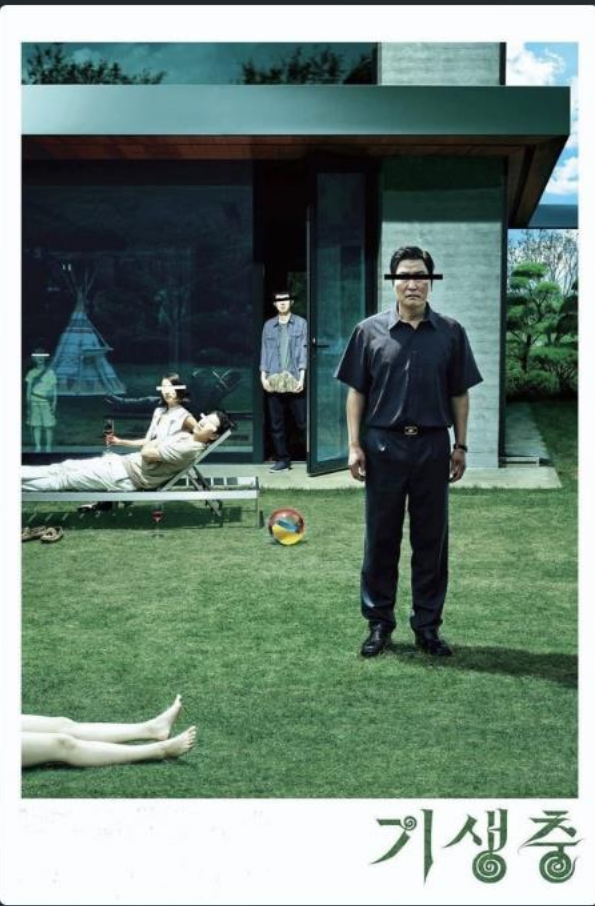


최신 영화 TV 프로그램 인물

앱 포럼 기여 랭킹 참여하기 API 지원

+ KO 로그인 회원가입

영화, TV 프로그램, 인물 검색



# 기생충 (2019)

86% 회원점수

▶ 트레일러 재생

## 개요

전원 백수로 살 길 막막하지만 사이는 좋은 기택(송강호) 가족. 장남 기우(최우식)에게 명문대생 친구가 연결시켜 준 고액 과외 자리는 모처럼 싹튼 고정수입의 희망이다. 온 가족의 도움과 기대 속에 박 사장(이선균) 집으로 향하는 기우. 글로벌 IT기업의 CEO인 박 사장의 저택에 도착하자 젊고 아름다운 사모님 연교(조여정)와 가정부 충숙(장혜진)이 기우를 맞이한다. 큰 문제 없이 박 사장의 딸 다혜(정지소)의 과외를 시작한 기우. 그러나 이렇게 시작된 두 가족의 만남 뒤로, 건잡을 수 없는 사건이 기다리고 있는데...

## 주요 제작진

Bong Joon-ho  
Screenplay, Director, Story

Han Jin-won  
Screenplay



## 주요 출연진



**Song Kang-ho**  
Kim Ki-taek



**Lee Sun-kyun**  
Park Dong-ik



**Cho Yeo-jeong**  
Yeon-kyo



**Choi Woo-shik**  
Ki-woo



**Park So-dam**  
Ki-jung

## 총 출연진 &amp; 제작진

사용자의  
견

리뷰 0

토론 내역 15

Haters make yourselves known

토론  
중

3

2월 15, 2020 at 7:28 AM  
작성자: Celluloid Fan

Undeserving of Oscar Best Picture

토론  
중

2

2월 13, 2020 at 1:53 PM  
작성자: znexyish

사항

상태  
개봉됨

## 개봉 정보

8월 30, 2019

프리미어

10월 11, 2019

극장 (제한)

11월 1, 2019

극장

1월 14, 2020

디지털

1월 28, 2020

물리매체

원어  
한국어상영시간  
2h 11m제작비  
\$11,363,000.00



Oscars

토론  
중

12

2월 11, 2020 at 3:42 PM  
작성자: znexyish

토론으로 이동

미디어

인기 있는

동영상 18

배경 26

포스터 73



추천

수익

\$165,923,763.00

장르

코미디

스릴러

드라마

키워드

underground

seoul

birthday party

private lessons

basement

con artist

working class

limousine driver

class differences

rich poor

housekeeper

tutor

family

crime family

flood

smell

unemployed

wealthy family

korean

# 1. Data Dictionary

총 23개의 칼럼

1단계 : Collecting

- revenue와 관계 확인→Collecting

2단계 : 전처리

- null 처리
- 클래스 구조 분리
- 가중치 없애기 (Onehot Encoding)

```
[ ] df.info()

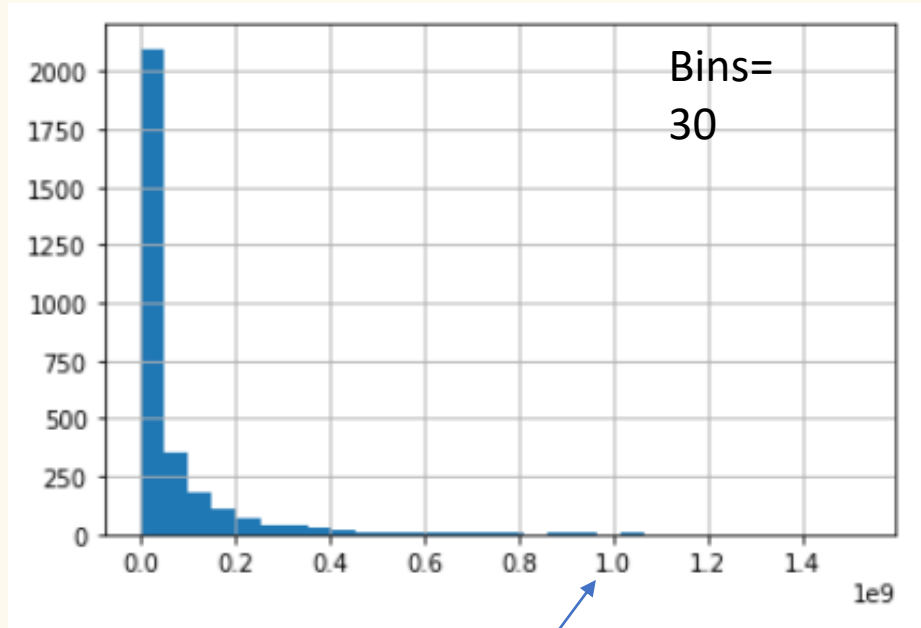
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 23 columns):
 id                3000 non-null int64
 belongs_to_collection  604 non-null object
 budget            3000 non-null int64
 genres            2993 non-null object
 homepage          946 non-null object
 imdb_id           3000 non-null object
 original_language  3000 non-null object
 original_title     3000 non-null object
 overview          2992 non-null object
 popularity         3000 non-null float64
 poster_path       2999 non-null object
 production_companies  2844 non-null object
 production_countries  2945 non-null object
 release_date       3000 non-null object
 runtime           2998 non-null float64
 spoken_languages   2980 non-null object
 status            3000 non-null object
 tagline           2403 non-null object
 title             3000 non-null object
 Keywords           2724 non-null object
 cast              2987 non-null object
 crew              2984 non-null object
 revenue           3000 non-null int64
 dtypes: float64(2), int64(3), object(18)
memory usage: 539.2+ KB
```

```
[ ] df.shape

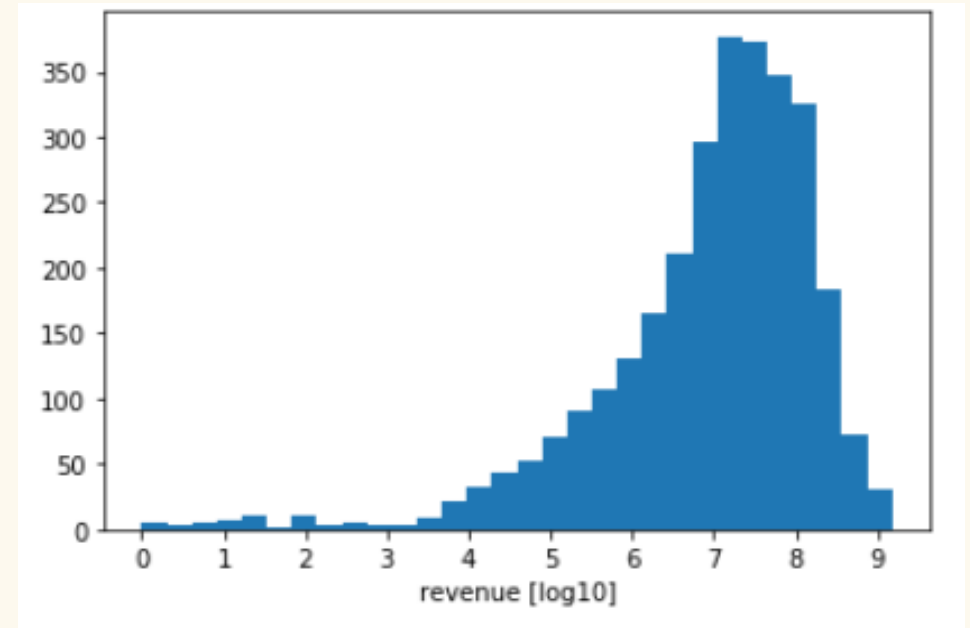
(3000, 23)
```

# Revenue

## 1. Data Dictionary



1억\$



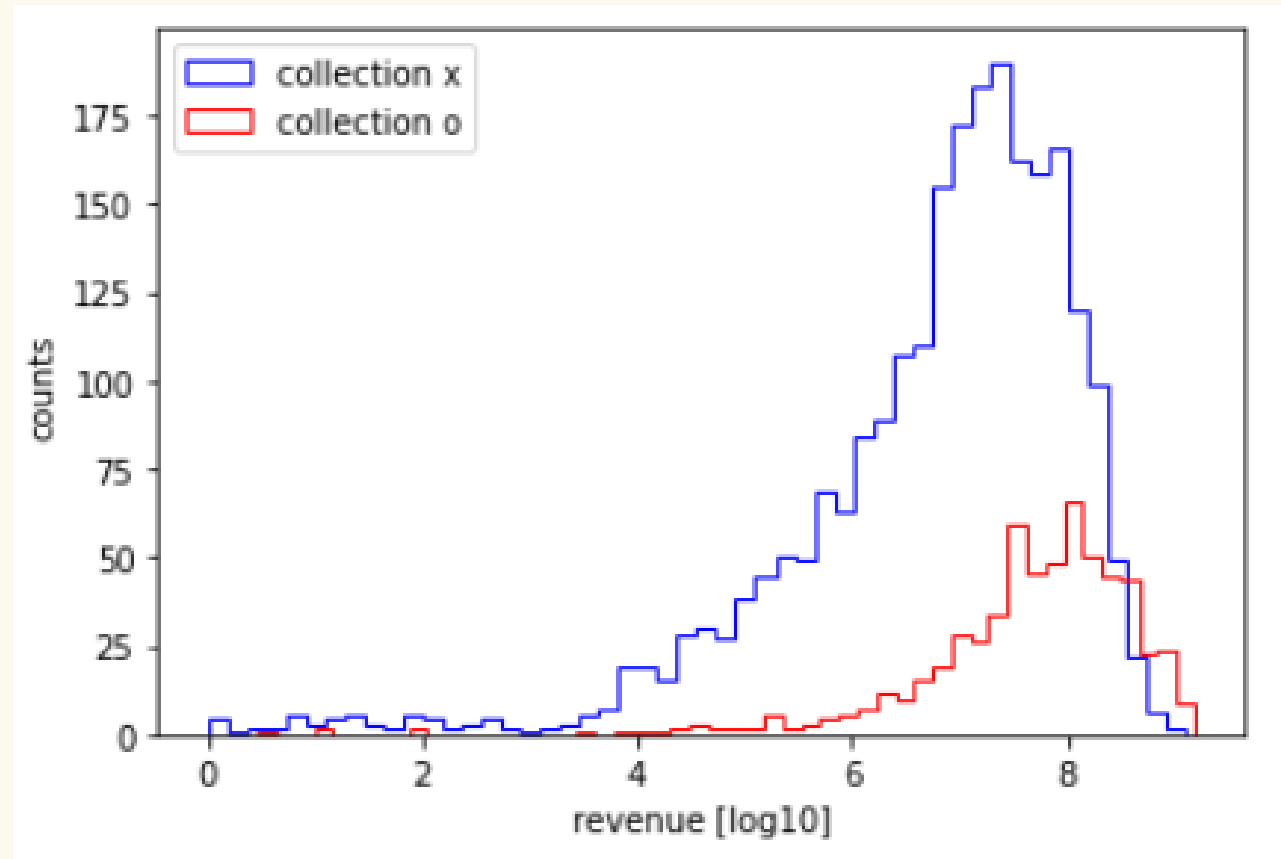
Log



# Belong\_to\_collection

## 1. Data Dictionary

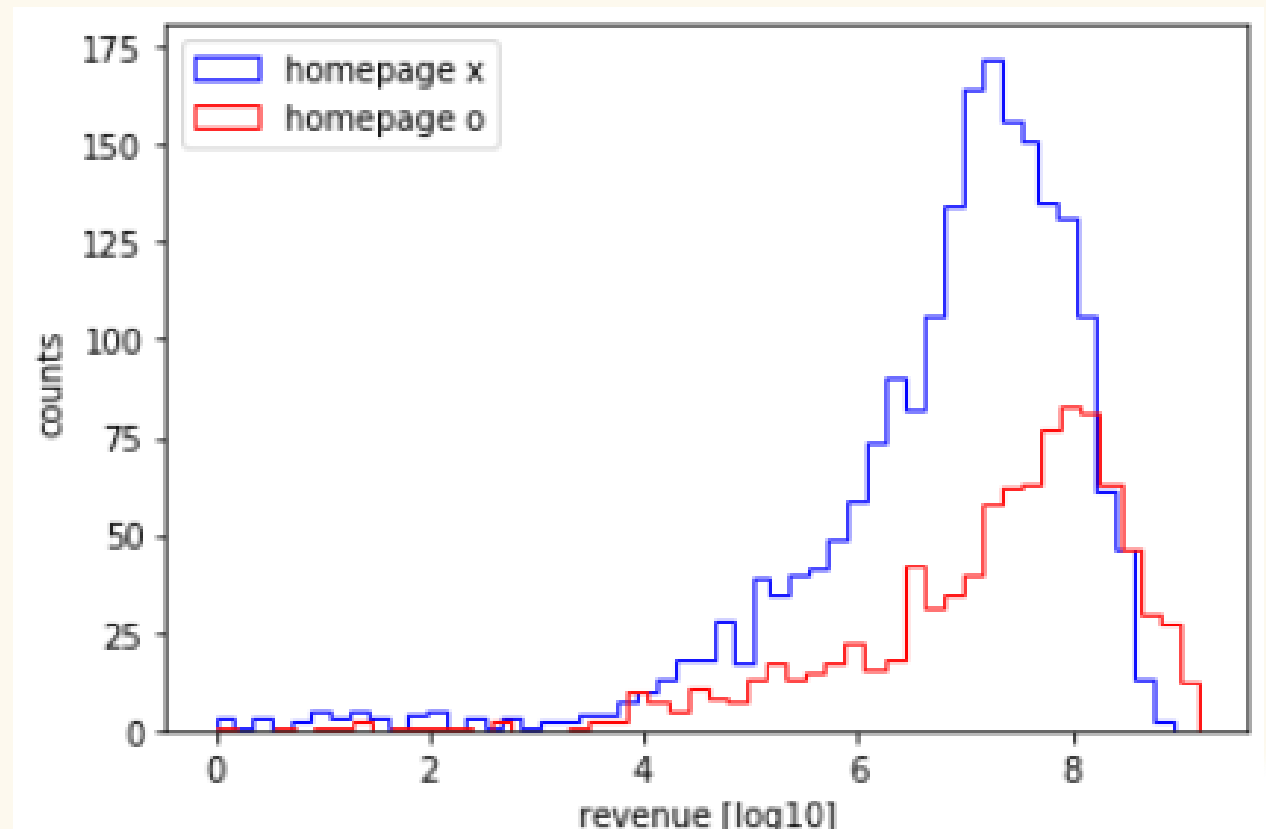
- Collection이 있는 경우 604개, 없는 경우 2396개.
- Collection이 있는 경우의 평균이 더 크다 (t-test 통과).
- Collection이 있을 때, 평균= $43(x10^6)$
- Collection이 없을 때, 평균= $160(x10^6)$



# Homepage

## 1. Data Dictionary

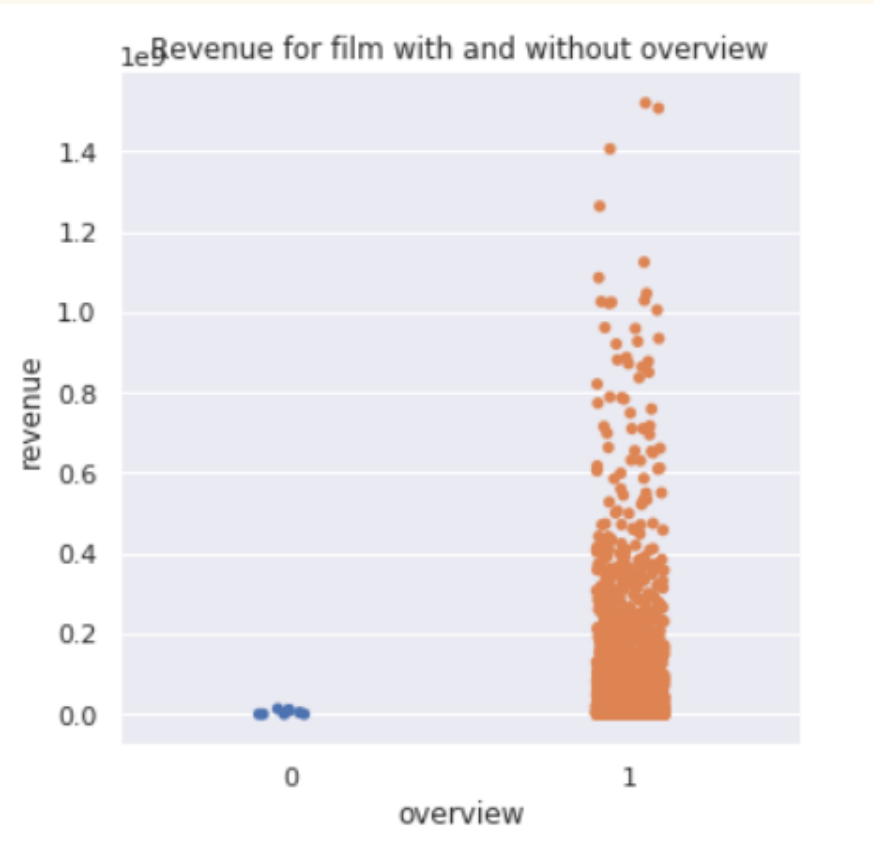
- Homepage가 있는 경우 946개, 없는 경우 2054개.
- Homepage가 있는 경우의 평균이 더 크다. (t-test 통과)
- Homepage가 있을 때, 평균=42( $\times 10^6$ )
- Homepage가 없을 때, 평균=120( $\times 10^6$ )



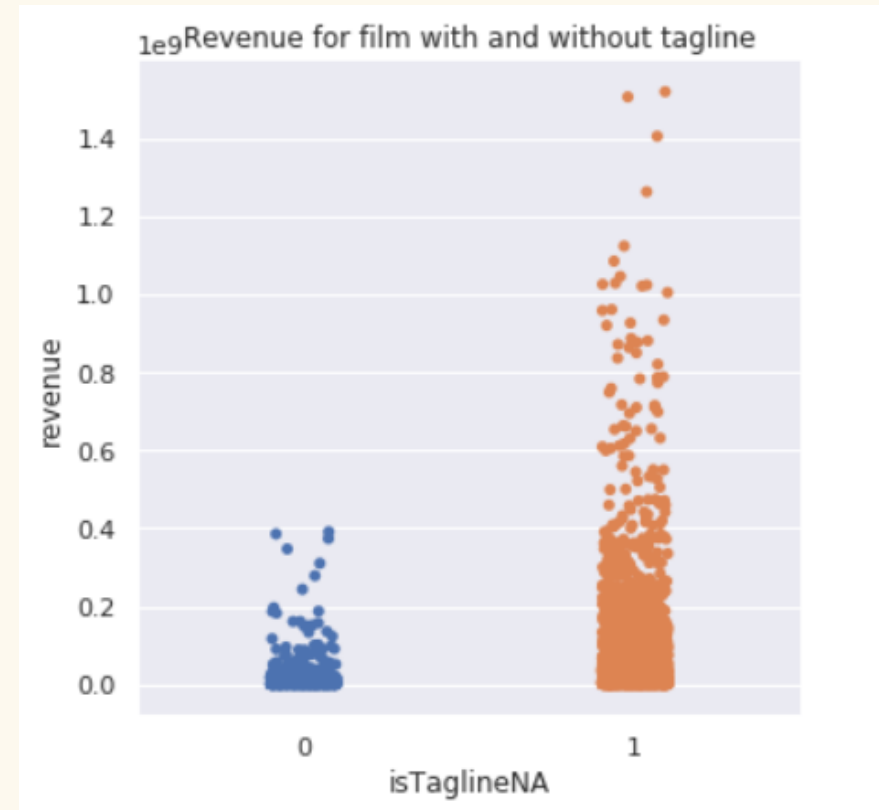
# Overview & Tagline

## 1. Data Dictionary

Overview가 있을 때 수익이 높다.



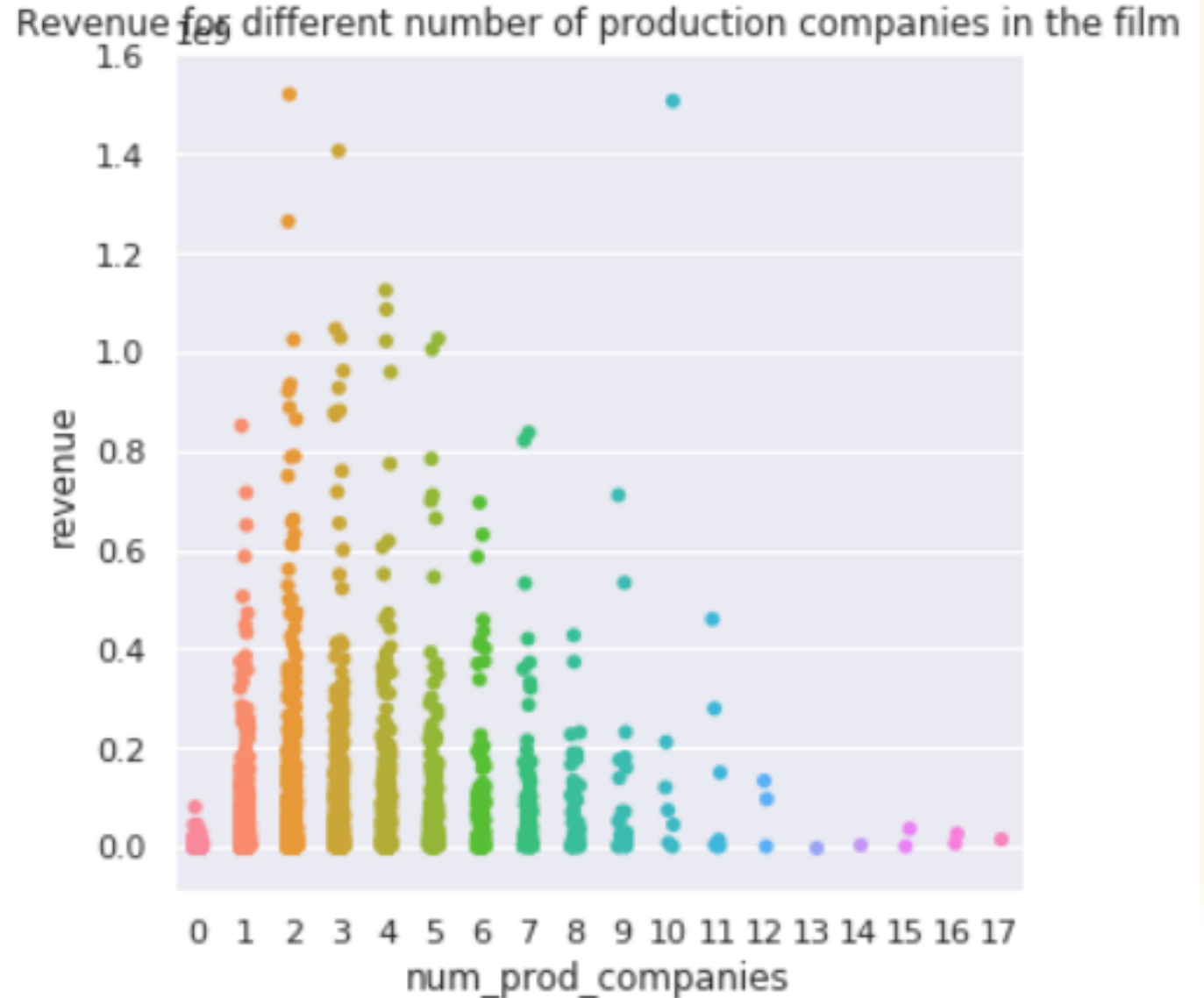
Tagline이 있을 때 수익이 높다.



# Num\_prod\_companies

1. Data  
Dictionary

참여한 제작사 수  
(1~6)개 일때 수익이 높다



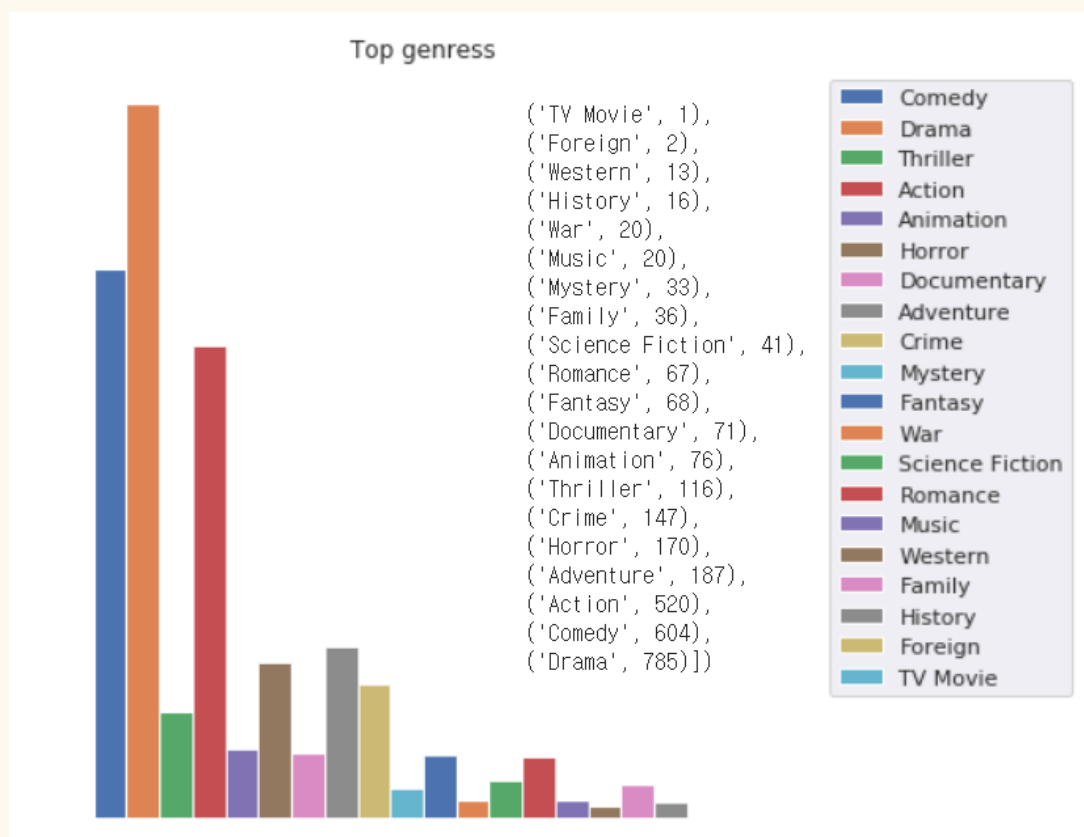


# Genre

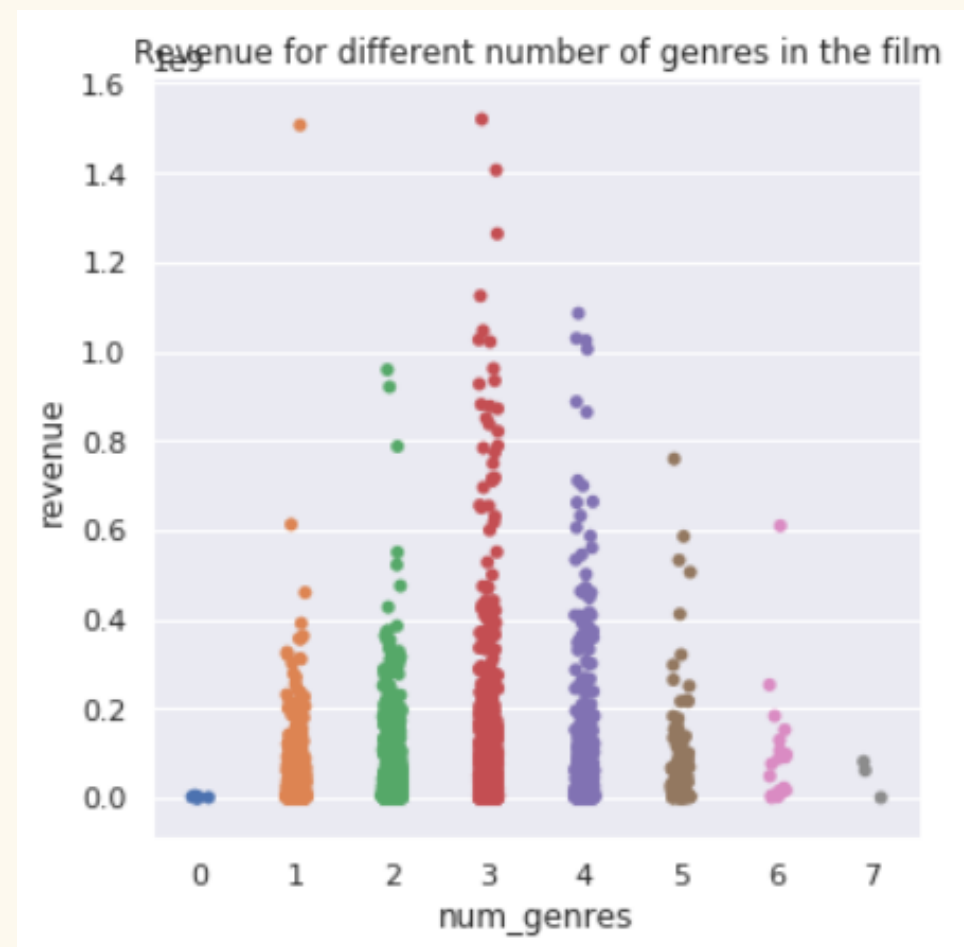
## 1. Data Dictionary

### 장르 갯수

(Drama > Comdey > Action>...)



### 장르 수에 따른 수익분포



# Crew\_members & Cast\_members

## 1. Data Dictionary

제작진 수가 적은 영화가 수익이 높다.



출연진 수가 적을수록 수익도 적다.

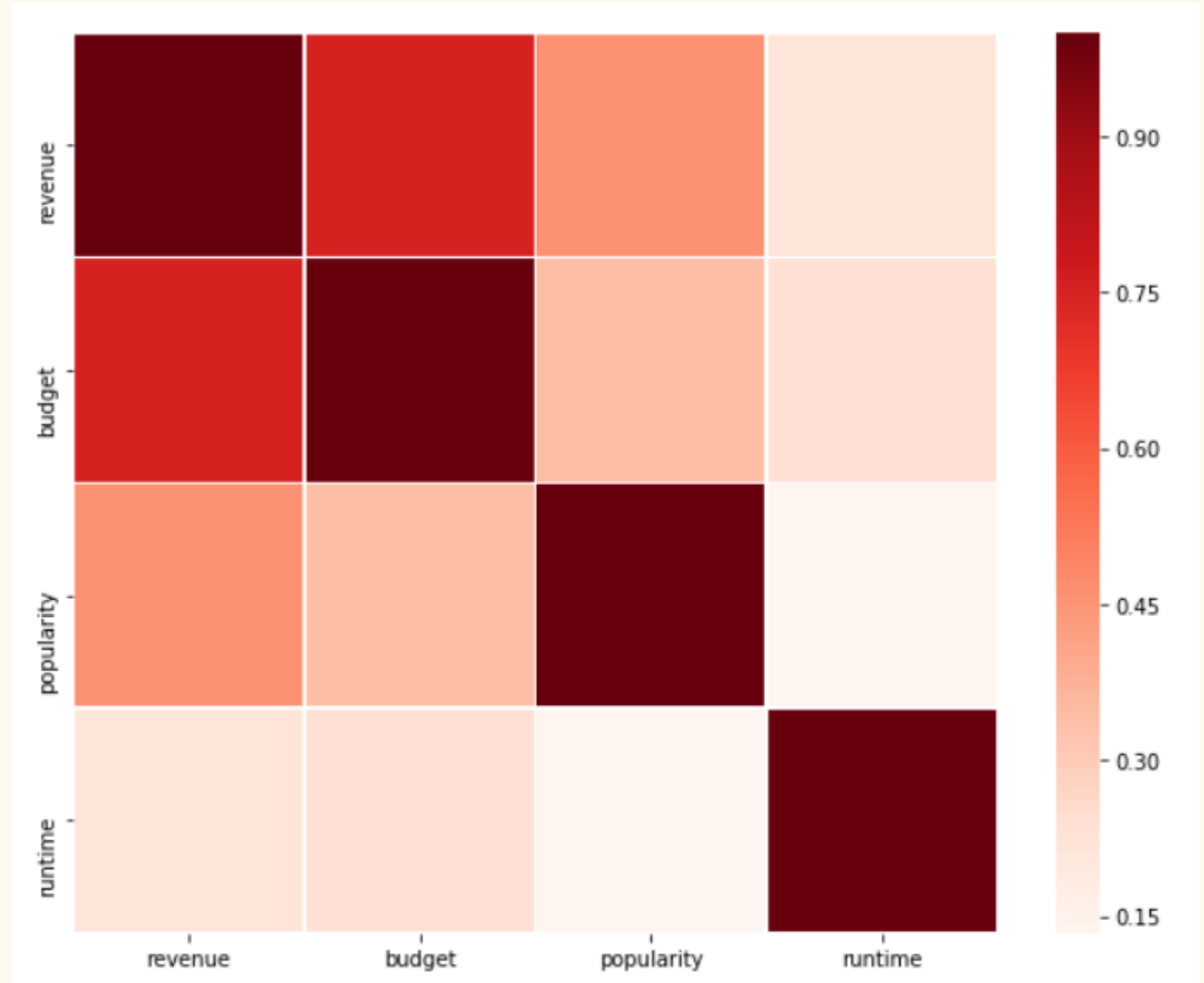


# Heat Map

( Revenue, Budget  
Popularity, runtime )

- revenue와 budget 높은 상관관계
- popularity와 runtime 낮은 상관관계

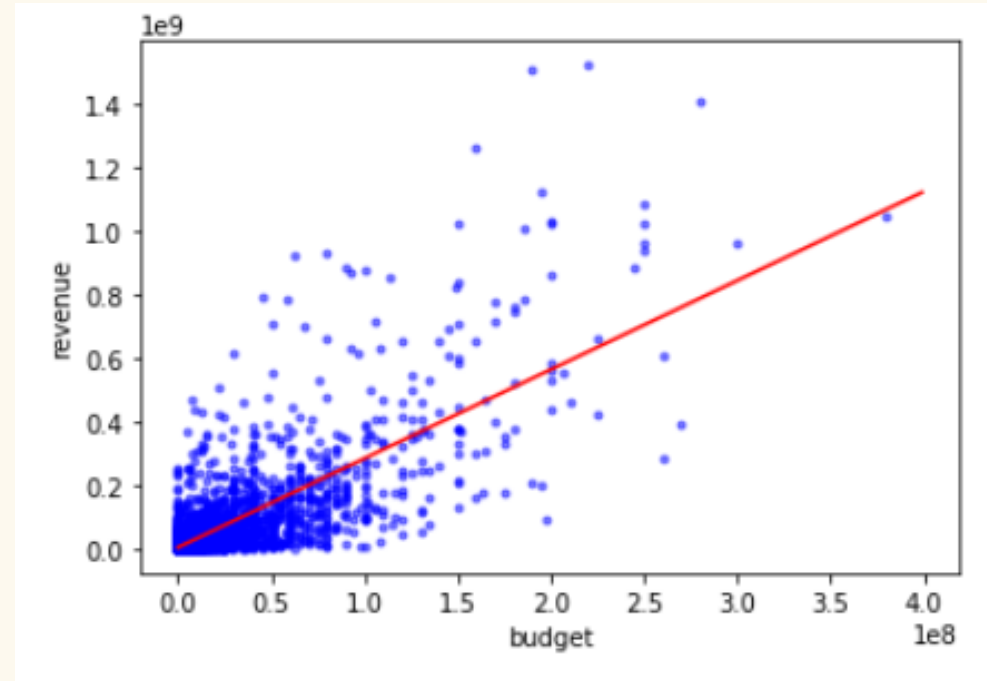
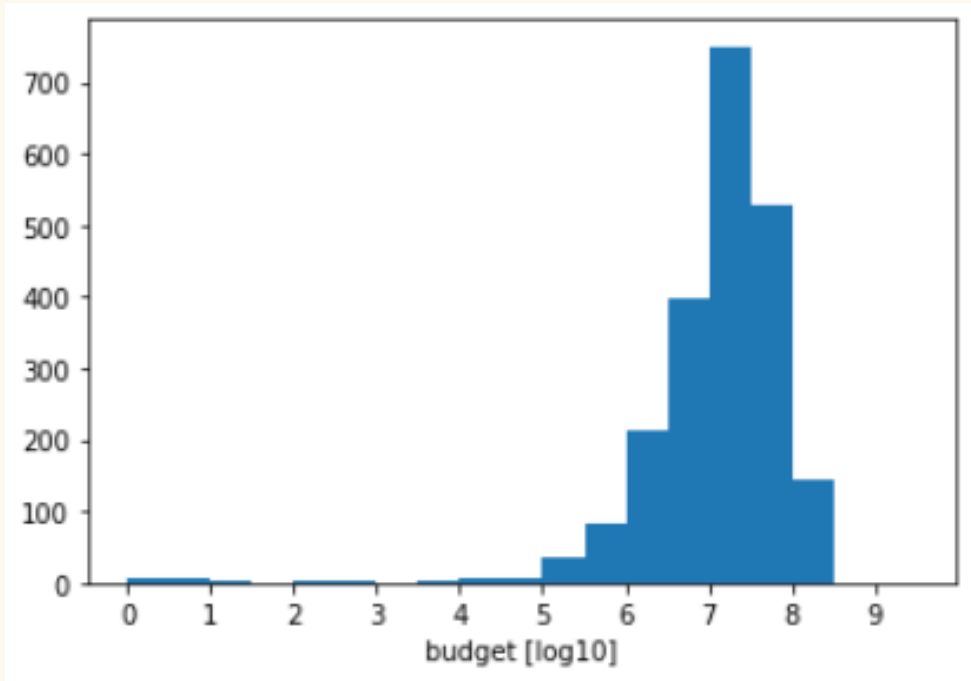
## 1. Data Dictionary



# Budget

## 1. Data Dictionary

- 0 값이 812개 존재 (10 이하의 값이 824개)
- 수익과는 양의 상관관계를 보임, 사용하려면 0값의 처리가 필요

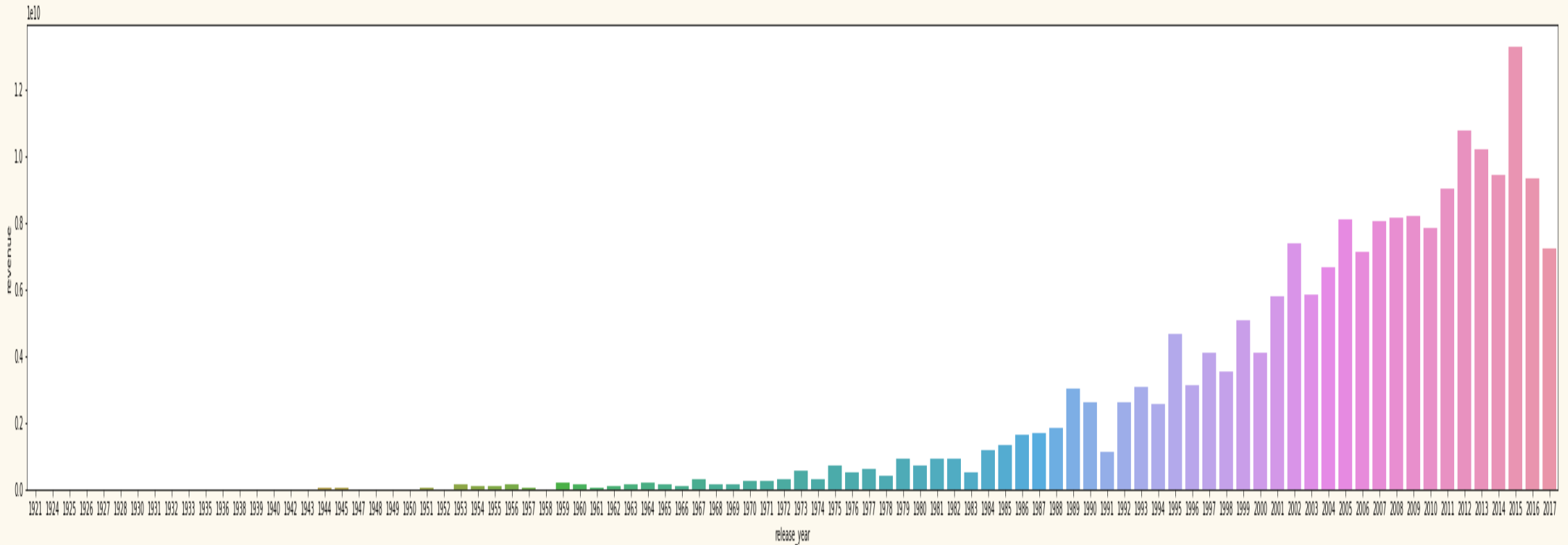




# Release\_date

연도증가와 revenue가 높은 상관관계를 보여 inflation을 접목시키자는 의견  
도출

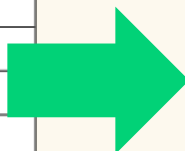
year



# Collecting

## 1. Data Dictionary

	Column_name	데이터 분석 여부	비고(데이터 특성/특징)
1	id	X	데이터 순서(1-3000)
2	belongs_to_collection	O	
3	budget	O	
4	genres	O	Dictionary type
5	homepage	O	
6	imdb_id	X	고유 코드
7	original_language	O	
8	original_title	X	
9	overview	X	
10	popularity	O	
11	poster_path	X	
12	production_companies	O	Dictionary type
13	production_countries	O	Dictionary type
14	release_date	O	
15	runtime	O	
16	spoken_languages	O	Dictionary type
17	status	X	
18	tagline	O	영화마지막대사(문자열)
19	title	X	
20	Keywords	O	Dictionary type
21	cast	O	Dictionary type
22	crew	O	Dictionary type
23	revenue	O	목표 값 (Y)



	Column_name	데이터 분석 여부	비고(데이터 특성/특징)
1	belongs_to_collection	O	
2	budget	O	
3	genres	O	Dictionary type
4	homepage	O	
5	original_language	O	
6	popularity	O	
7	production_companies	O	Dictionary type
8	production_countries	O	Dictionary type
9	release_date	O	
10	runtime	O	
11	spoken_languages	O	Dictionary type
12	tagline	O	영화마지막대사(문자열)
13	Keywords	O	Dictionary type
14	cast	O	Dictionary type
15	crew	O	Dictionary type
16	revenue	O	목표 값 (Y)

# 2

## Pre-Processing



## 2. Pre-Processing

1. (추가) inflation.csv 를 이용한 Data
2. null 처리
3. 클래스 구조 분리
4. 가중치 없애기 (Onehot Encoding)

	Column_name
1	belongs_to_collection
2	budget
3	genres
4	homepage
5	original_language
6	popularity
7	production_companies
8	production_countries
9	release_date
10	runtime
11	spoken_languages
12	tagline
13	Keywords
14	cast
15	crew
16	revenue



# Revenue & Budget

- inflation.csv 를 반영한 새로운 칼럼추가

	budget	revenue	release_year
0	14000000	13314651	2015
1	40000000		
2	3300000		
3	1200000		
4	0		
...	...		
2995	0		
2996	0		
2997	65000000		
2998	42000000		
2999	35000000		

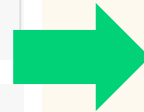
```
# Read training data
f_path = '/content/drive/My Drive/Colab Notebooks/Kaggle/TMDB Box Office Pr
f_name = 'train.csv' #train or test 로 변경
inflation_name = 'inflation_rate.csv'
#InputData = pd.read_csv(f_path + 'InputData' + f_name)
df = pd.read_csv(f_path + f_name)
inflationRate = pd.read_csv(f_path + "inflation_rate.csv", header = None)

def reviseRevenueByInflationRate(df, inflationRate):

    inflationRateYear = inflationRate[0].as_matrix()
    inflationRateString = inflationRate[1].as_matrix()
    inflationRateNumber = []
    for i in range(len(inflationRateString)):
        inflationRateNumber.append(float(inflationRateString[i][0:-1]))

    revisedRevenue = []
    for i in range(df.shape[0]):
        pastRevenue = df['revenue'].iloc[i]
        yearIndex = np.where(inflationRateYear == df['release_year'])[i]

        if (len(yearIndex[0]) == 0):
            for j in range(len(inflationRateNumber)):
                pastRevenue *= (100 + inflationRateNumber[j]) / 100
                revisedRevenue.append(pastRevenue)
            else:
                for j in range(yearIndex[0][0], len(inflationRateNumber)):
                    pastRevenue *= (100 + inflationRateNumber[j]) / 100
                    revisedRevenue.append(pastRevenue)
        df['revisedRevenue'] = revisedRevenue
```



	revisedBudget	revisedRevenue	release_year
0	1.532000e+07	1.347575e+07	2015
1	5.578810e+07	1.327051e+08	2004
2	3.640032e+06	1.444100e+07	2014
3	1.366342e+06	1.821790e+07	2012
4	0.000000e+00	4.797088e+06	2009
...	...	...	...
2995	0.000000e+00	2.816887e+06	1994
2996	0.000000e+00	2.021860e+05	2013
2997	1.089353e+08	1.499231e+08	1996
2998	5.857750e+07	2.398377e+08	2004
2999	4.104720e+07	9.626995e+07	2011

# Collection & Homepage & Tagline

- 有/無를 0과 1로 변환

```
# dataframe -> array(nan: 0, finite: 1)
def arr_onezero(df_column):
    ndata = len(df_column)

    arr1 = np.zeros(ndata)
    ind1 = df_column.notnull()      # True=1, False=0

    arr1[ind1] = 1

    return arr1
```



```
[ ] arr_homepage = arr_onezero(df['homepage'])
    df['!shompageNA'] = arr_homepage
    df.drop(['homepage'], axis=1, inplace=True)
```

```
[ ] arr_collection = arr_onezero(df['belongs_to_collection'])
    df['!scollectionNA'] = arr_collection
    df.drop(['belongs_to_collection'], axis=1, inplace=True)
```

```
df=make_taglineNA(df)
df.drop(['tagline'], axis=1, inplace=True)
```

# Class Features

- 클래스 구조 분리 → 가중치 없애기(Onehot Encoding)
- : Language , Production, Release date , Genre, Keywords, Cast, Crew

```
test['spoken_languages'].value_counts()
```

```
train data
```

```
[{'iso_639_1': 'en', 'name': 'English'}]
```

```
[{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'es', 'name': 'Español'}]
```

```
[{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'fr', 'name': 'Français'}]
```

```
[{'iso_639_1': 'ru', 'name': 'Русский'}]
```

```
[{'iso_639_1': 'fr', 'name': 'Français'}]
```

```
[{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'ar', 'name': 'العربية'}, {'iso_639_1': 'de', 'name': 'Deutsch'}]
```

```
[{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'es', 'name': 'Español'}, {'iso_639_1': 'th', 'name': 'ภาษาไทย'}]
```

```
[{'iso_639_1': 'ja', 'name': '日本語'}, {'iso_639_1': 'zh', 'name': '普通话'}, {'iso_639_1': 'ar', 'name': 'العربية'}, {'iso_639_1':
```

```
[{'iso_639_1': 'ru', 'name': 'Русский'}, {'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'he', 'name': 'עברית'}]
```

```
[{'iso_639_1': 'el', 'name': 'Ελληνικά'}, {'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'it', 'name': 'Italiano'}]
```

```
Name: spoken_languages, Length: 401, dtype: int64 test['Keywords'].value_counts()
```

```
train data
```

```
[{'id': 10183, 'name': 'independent film'}]
```

```
[{'id': 187056, 'name': 'woman director'}]
```

```
[{'id': 179431, 'name': 'duringcreditsstinger'}]
```

```
[{'id': 9937, 'name': 'suspense'}]
```

```
[{'id': 5565, 'name': 'biography'}]
```

```
[{'id': 321, 'name': 'terror'}, {'id': 1523, 'name': 'obsession'}, {'id': 2249, 'name': 'camcorder'}, {'id': 2733, 'name': 'firemen
```

```
[{'id': 10092, 'name': 'mystery'}, {'id': 10540, 'name': 'bollywood'}, {'id': 11734, 'name': 'police corruption'}, {'id': 14536, 'n
```

```
[{'id': 1003, 'name': 'photographer'}, {'id': 4452, 'name': 'butcher'}, {'id': 4459, 'name': 'vegetarian'}, {'id': 5152, 'name': 'm
```

```
[{'id': 1399, 'name': 'senate'}, {'id': 1402, 'name': 'general'}, {'id': 5563, 'name': 'market'}, {'id': 9823, 'name': 'rivalry'},
```

```
[{'id': 1889, 'name': 'lake'}, {'id': 5767, 'name': 'summer camp'}, {'id': 9826, 'name': 'murder'}, {'id': 10714, 'name': 'serial k
```

```
Name: Keywords, Length: 2648, dtype: int64
```

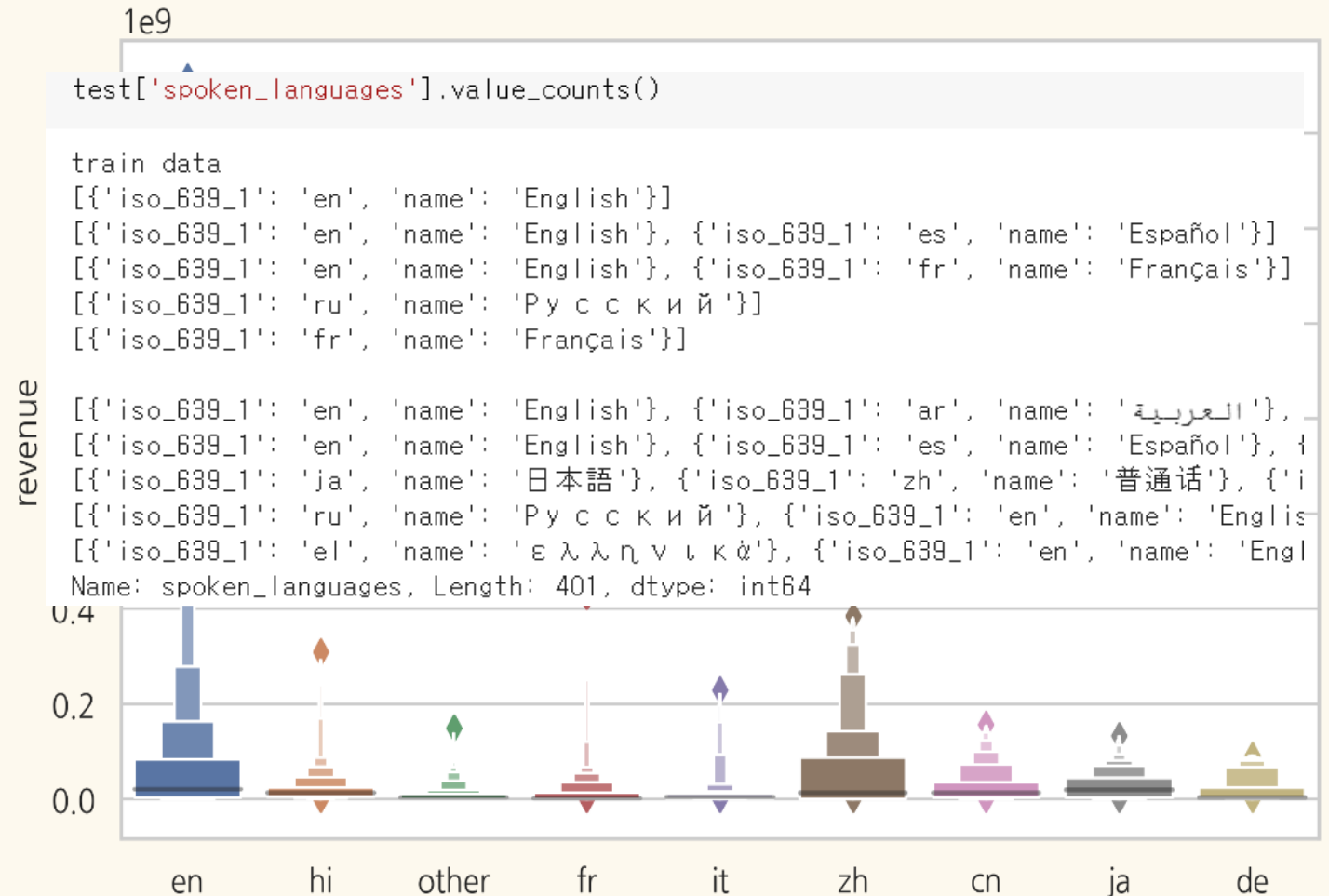
# Original\_language & Spoken\_languages

## Original

- 1) 총 36개의 언어
- 2) 관계성이 커 보이는 언어 선정
- 3) One-hot 인코딩

## Spoken

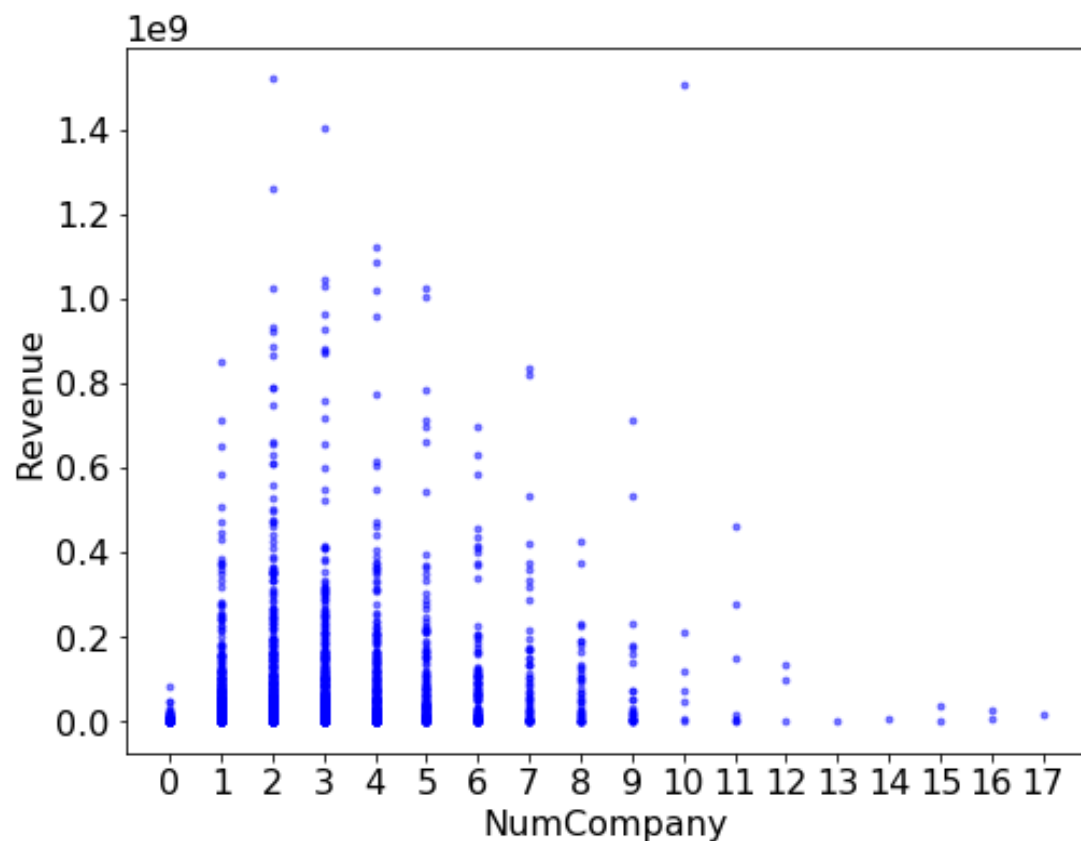
- 번역된 언어 수 카운트 (max:9)



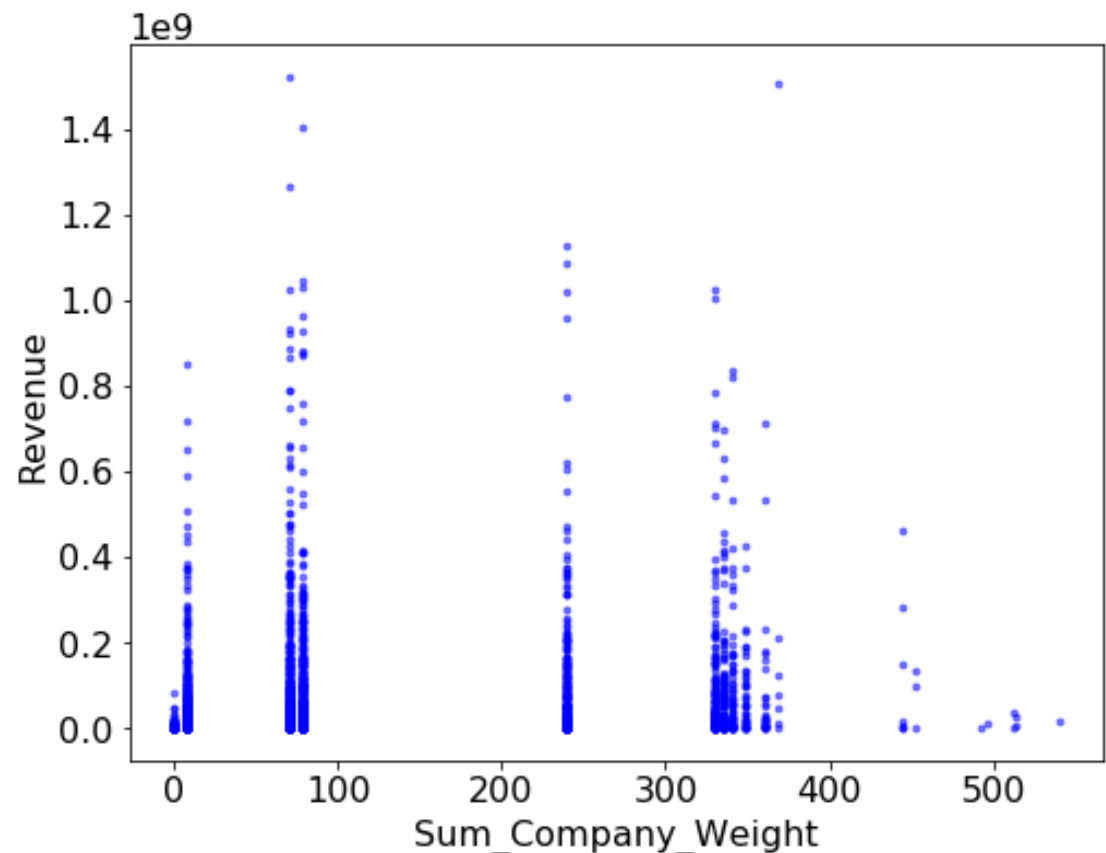


# Production\_companies

1) 참여 제작사의 수

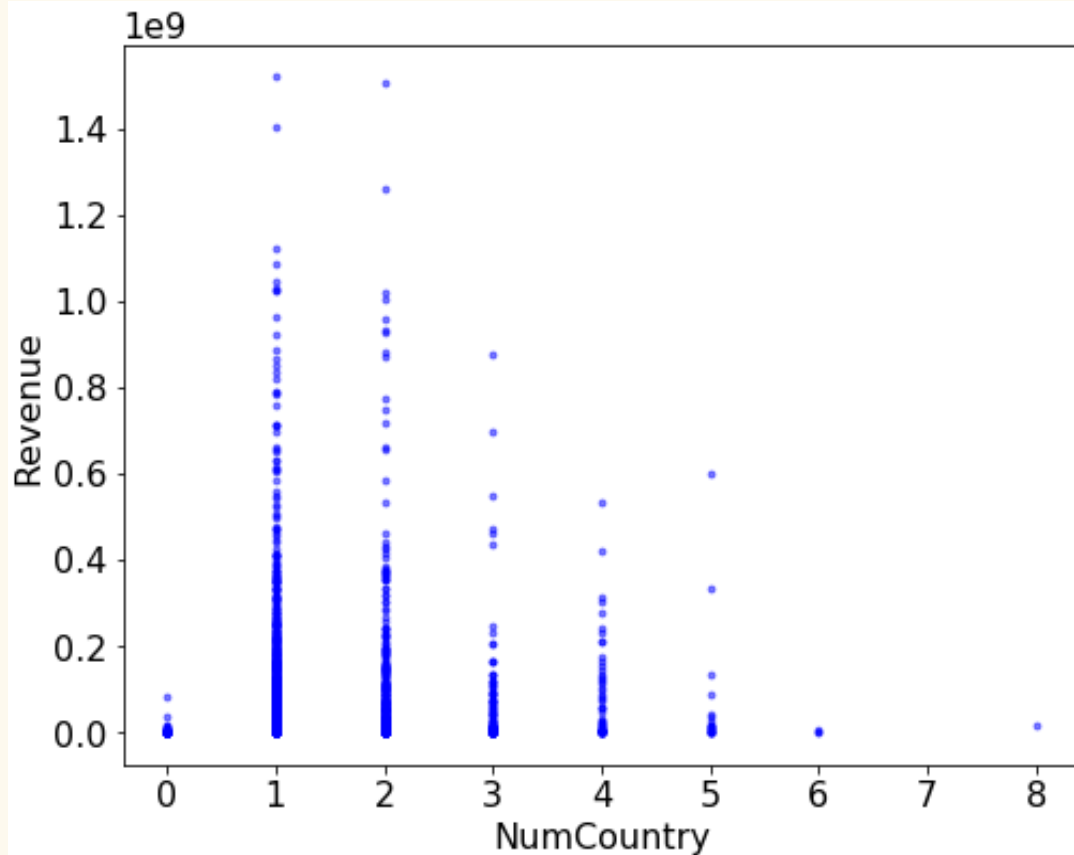


2) 참여 제작사의 영화 제작 빈도의 합을 weight으로 사용

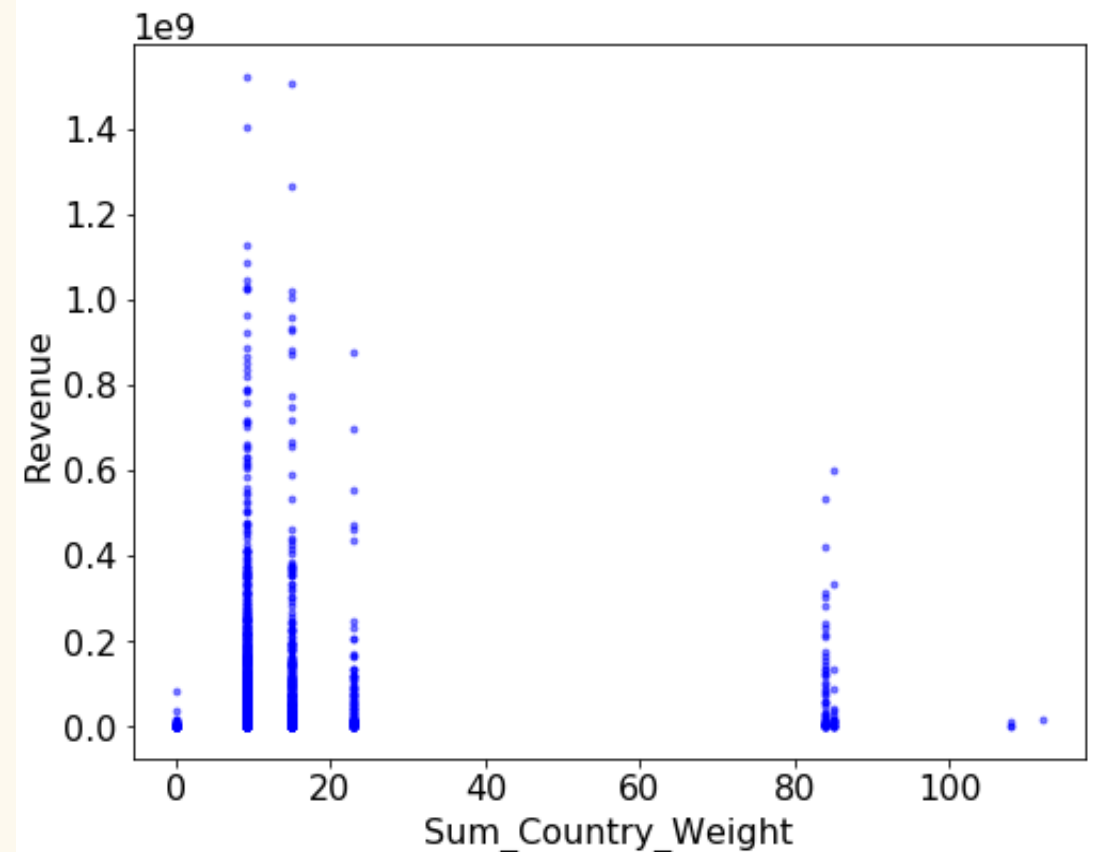


# Production\_countries

## 1) 참여 국가의 수



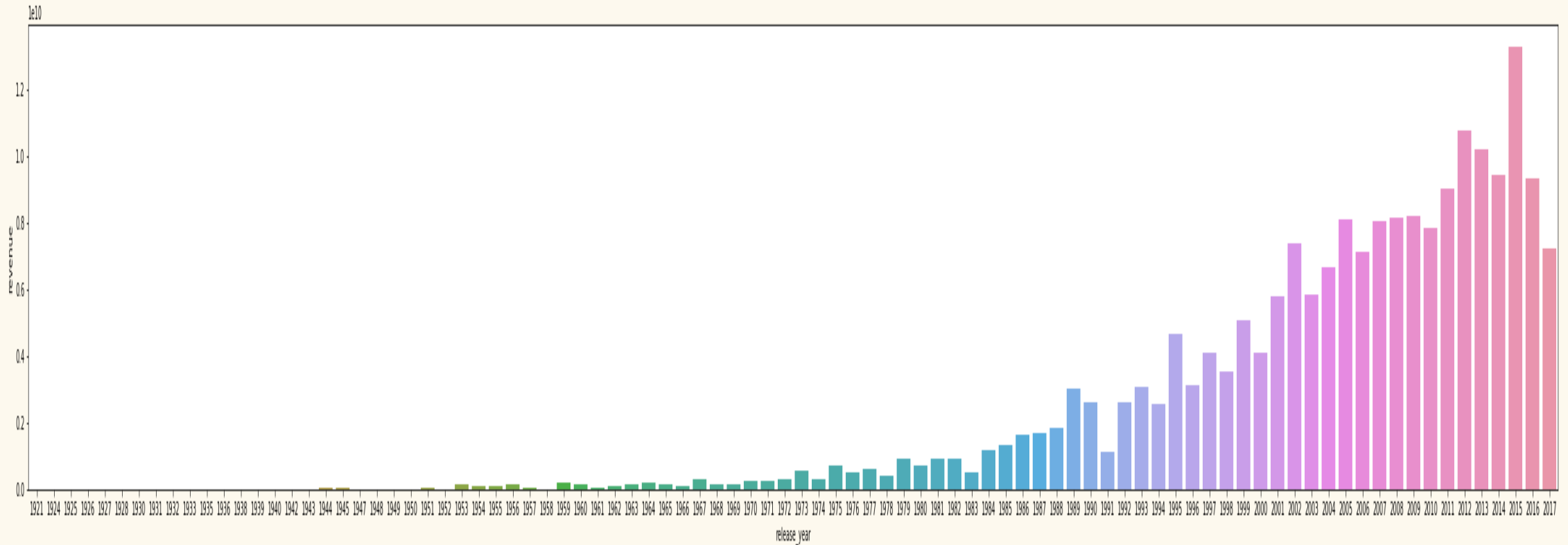
## 2) 참여 국가별 영화 제작 빈도의 합을 weight로 사용



# Release\_date

문자형태 '월/일/년'(예\_ 2/20/15)를 정수형태 year, month, season, day로 변경

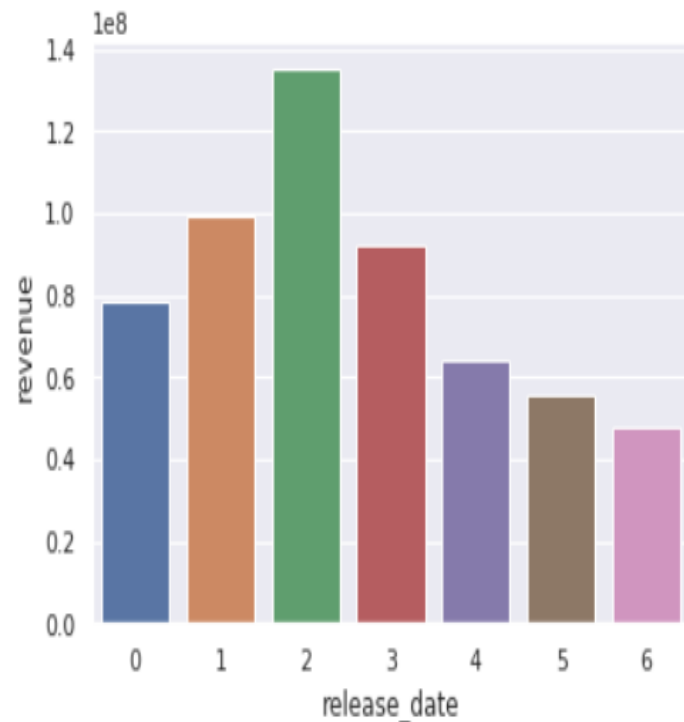
year



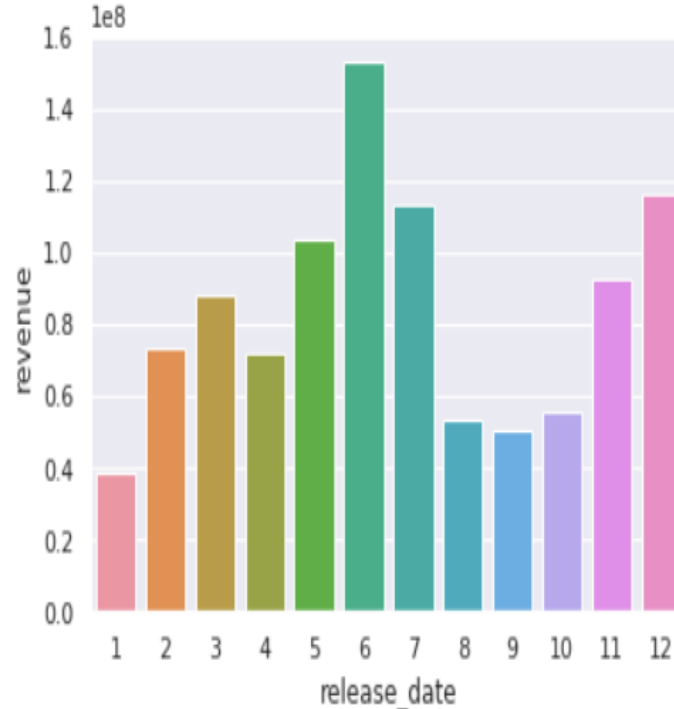
# Release\_date

문자형태 '월/일/년'(예\_ 2/20/15)를 정수형태 year, month, season, day로 변경

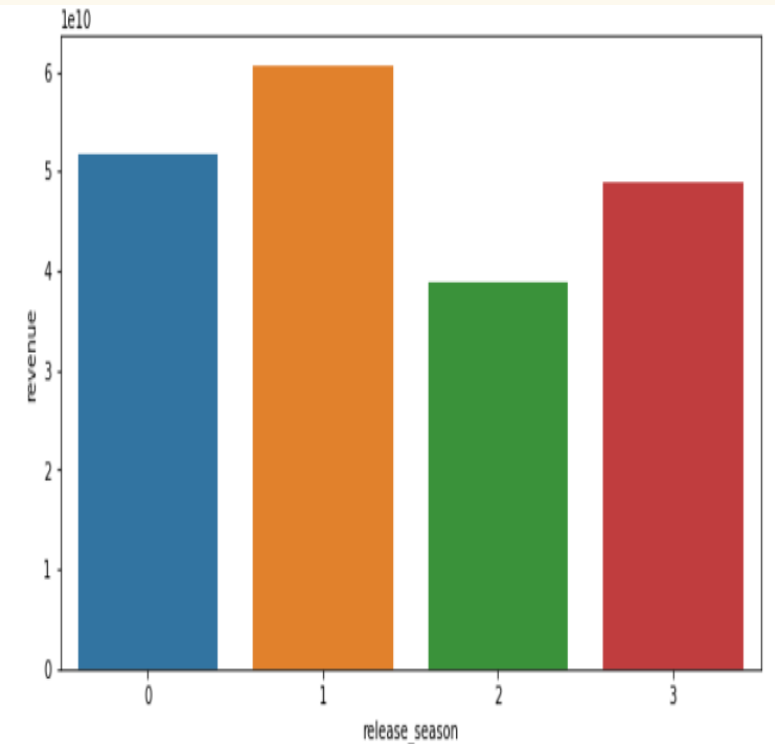
day



month



season



# Day, Month, Season

month, season, day의 가중치를 없애기 위해 모두 One-hot 인코딩

```
month_one_hot_encoded = pd.get_dummies(df["release_month"])
month_one_hot_encoded.rename(columns = {1: 'Jan', 2: "Feb", 3: "Mar", 4: "Apr",
                                         # 5: 'May', 6: "Jun", 7: "Jul", 8: "Aug",
                                         9: 'Sep', 10: "Oct", 11: "Nov", 12: "Dec"}, inplace = True)

day_one_hot_encoded = pd.get_dummies(df["release_day"])
day_one_hot_encoded.rename(columns = {0: 'Sun', 1: "Mon", 2: "Tue", 3: "Wen",
                                       4: 'Thu', 5: "Fri", 6: "Sat"}, inplace = True)

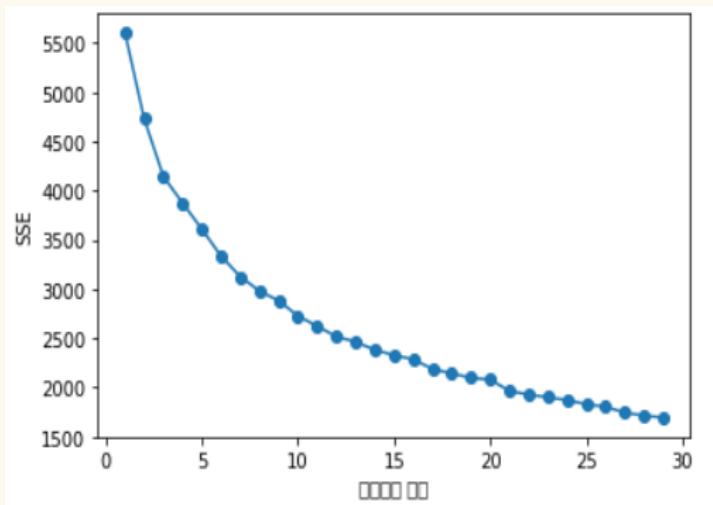
df = pd.concat([df, month_one_hot_encoded], axis=1)
df = pd.concat([df, day_one_hot_encoded], axis=1)

from sklearn.preprocessing import OneHotEncoder
seasons_one_hot_encoded = pd.get_dummies(df["release_season"])
seasons_one_hot_encoded.head(10)
SEASONS = ["spring", "summer", "autumn", "winter"]
seasons_one_hot_encoded.rename(columns = {0: 'spring', 1: "summer", 2: "autumn", 3: "winter"}, inplace = True)
seasons_one_hot_encoded.head()
df = pd.concat([df, seasons_one_hot_encoded], axis=1)
```

Jan	Feb	Mar	Apr	May	Jun
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

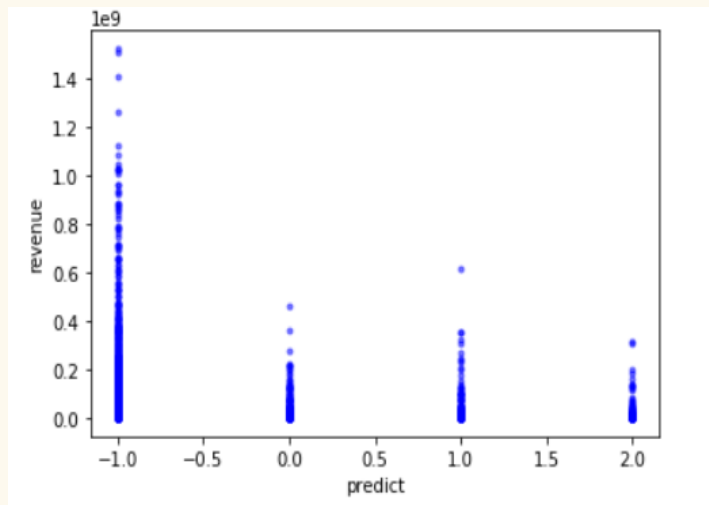
# Genre

## 2. Pre-Processing



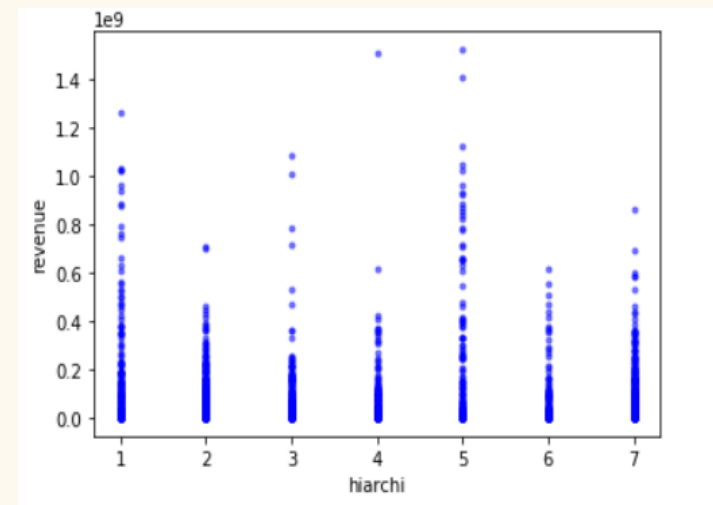
**K-means**

- 1) 군집이 늘어나면 늘어날 수록 SSR 감소. (거리 = Euclidian)
- 2) 군집 개수 별 Scatter plot을 그려 5개의 군집 생성



**DBSCAN**

- 1)  $\text{eps} = 0.5$ ,  $\text{min\_samples} = 120$ 으로 파라미터를 조정
- 2) 군집별 revenue 확인 후 4개로 확정



**Hiarchical**

- 1) 군집개수를 계속 조정
- 2) 군집별로 차이를 보이는지 확인
- 3) 군집 method별로 One-hot 인코딩

# Keywords

1) Keyword 종류 : 7400개

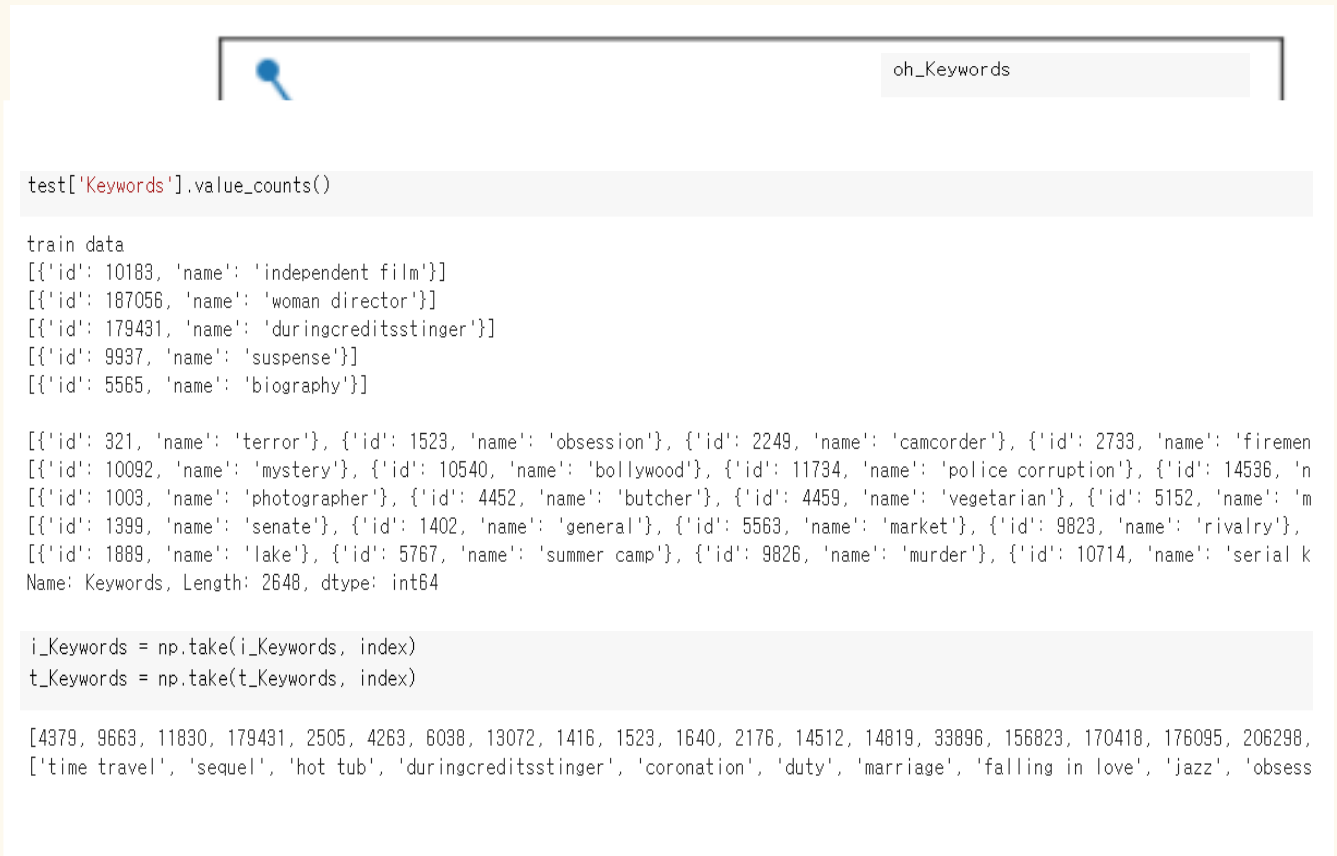
(one-hot 불가)

2) 군집 개수 별 Scatter plot

을 그려 파라미터 선택

3) K-means 군집 형성 (13개)

4) one-hot 인코딩



The screenshot shows a Jupyter Notebook interface with a search bar at the top containing 'oh\_Keywords'. Below it, a code cell contains the following Python code:

```
test['Keywords'].value_counts()

train data
[{'id': 10183, 'name': 'independent film'}}
[{'id': 187056, 'name': 'woman director'}}
[{'id': 179431, 'name': 'duringcreditsstinger'}}
[{'id': 9937, 'name': 'suspense'}}
[{'id': 5565, 'name': 'biography'}}

[{'id': 321, 'name': 'terror'}, {'id': 1523, 'name': 'obsession'}, {'id': 2249, 'name': 'camcorder'}, {'id': 2733, 'name': 'firemen'},
{'id': 10092, 'name': 'mystery'}, {'id': 10540, 'name': 'bollywood'}, {'id': 11734, 'name': 'police corruption'}, {'id': 14536, 'name': 'n'},
{'id': 1003, 'name': 'photographer'}, {'id': 4452, 'name': 'butcher'}, {'id': 4459, 'name': 'vegetarian'}, {'id': 5152, 'name': 'm'},
{'id': 1399, 'name': 'senate'}, {'id': 1402, 'name': 'general'}, {'id': 5563, 'name': 'market'}, {'id': 9823, 'name': 'rivalry'},
{'id': 1889, 'name': 'lake'}, {'id': 5767, 'name': 'summer camp'}, {'id': 9826, 'name': 'murder'}, {'id': 10714, 'name': 'serial k'}
Name: Keywords, Length: 2648, dtype: int64

i_Keywords = np.take(i_Keywords, index)
t_Keywords = np.take(t_Keywords, index)

[4379, 9663, 11830, 179431, 2505, 4263, 6038, 13072, 1416, 1523, 1640, 2176, 14512, 14819, 33896, 156823, 170418, 176095, 206298,
'time travel', 'sequel', 'hot tub', 'duringcreditsstinger', 'coronation', 'duty', 'marriage', 'falling in love', 'jazz', 'obsess
```



# Cast (출연진)

```
{'cast_id': 4, 'character': 'Lou', 'credit_id': '52fe4ee7c3a36847f82afae7', 'gender': 2, 'id': 52997, 'name': 'Rob Corddry', 'order': 0}  
{'cast_id': 5, 'character': 'Nick', 'credit_id': '52fe4ee7c3a36847f82afaeb', 'gender': 2, 'id': 64342, 'name': 'Craig Robinson', 'order': 1}  
{'cast_id': 6, 'character': 'Jacob', 'credit_id': '52fe4ee7c3a36847f82afaef', 'gender': 2, 'id': 54729, 'name': 'Clark Duke', 'order': 2}  
{'cast_id': 7, 'character': 'Adam Jr.', 'credit_id': '52fe4ee7c3a36847f82afaf3', 'gender': 2, 'id': 36801, 'name': 'Adam Scott', 'order': 3}
```

- 이름 및 주연 순위 정보
- 주연 횟수 합을 구하여 사용

MaincastSum	
0	3
1	12
2	6
3	3
4	3
...	...

# Crew (제작진)

```
{'credit_id': '59ac067c92514107af02c8c8', 'department': 'Directing', 'gender': 0, 'id': 1449071, 'job': 'First Assistant Director', 'name': 'Kelly Cantley'}  
{'credit_id': '52fe4ee7c3a36847f82afad7', 'department': 'Directing', 'gender': 2, 'id': 3227, 'job': 'Director', 'name': 'Steve Pink'}  
{'credit_id': '5524ed25c3a3687ded000d88', 'department': 'Writing', 'gender': 2, 'id': 347335, 'job': 'Writer', 'name': 'Josh Heald'}
```

- 각 Crew들의 역할 정보
- 감독과 작가만 사용
- Sum, Mean, Max로 가공

	DirectorSum	DirectorMean	DirectorMax	WriterSum	WriterMean	WriterMax
0	3	3.0	3	1	1.0	1
1	6	6.0	6	0	0.0	0
2	1	1.0	1	0	0.0	0
3	1	1.0	1	1	1.0	1
4	1	1.0	1	1	1.0	1
...	...	...	...	...	...	...

# 전처리 전 → 후

	Column_name
1	id
2	belongs_to_collection
3	budget
4	genres
5	homepage
6	imdb_id
7	original_language
8	original_title
9	overview
10	popularity
11	poster_path
12	production_companies
13	production_countries
14	release_date
15	runtime
16	spoken_languages
17	status
18	tagline
19	title
20	Keywords
21	cast
22	crew
23	revenue

	Column_name
1	belongs_to_collection
2	budget
3	genres
4	homepage
5	original_language
6	popularity
7	production_companies
8	production_countries
9	release_date
10	runtime
11	spoken_languages
12	tagline
13	Keywords
14	cast
15	crew
16	revenue

## 최종 Column 114개

[inflation.csv 적용결과]

- budget : 예산
- revisedBudget : inflation을 반영한 예산
- revenue : 수익
- revisedRevenue : 인플레이션을 고려한 수익

[모델 예측 결과]

- RF\_predict : RandomForest 예측값
- SVR\_predict : SVR 예측값
- line\_predict : lineare regression 예측값
- Poly\_predict : Polynomial regression 예측값
- KNN\_predict : KNN regression 예측값

- IscollectionNA : 영화 모음집 // 0 = 시리즈X, 1 = 시리즈O
- genres : 장르 종류 OnehotEncoding // 20개
- genres\_cluster : 장르 군집 번호 OnehotEncoding // genres\_k 5개, DBSCAN 4개, hiarchi 7개
- Cnt\_genres : 장르 수
- IshomepageNA : 홈페이지 링크 유무 // 0 = 홈페이지X, 1 = 홈페이지O
- 8\_languages : 원본 언어 순위//one-hot Encoding // 상위 8개 + other
- Cnt\_spok\_language : 더빙 언어 수
- Cnt\_prod\_companies : 제작사 협찬 수
- prod\_company\_weight2 : 제작사 회사별 빈도의 합을 weight로 저장
- Cnt\_prod\_countries : 제작사 나라 수
- prod\_country\_weight2 : 제작사 회사별 빈도의 합을 weight로 저장
- release\_year : 개봉 연도
- release\_month : 개봉 달 OnehotEncoding// 12 달
- release\_day : 개봉 요일OnehotEncoding // 7 요일
- release\_season : 개봉 계절 OnehotEncoding //0:봄, 1:여름, 2:가을, 3:겨울
- runtime : 영화 상영 시간 // NAN는 평균으로 처리
- IsTaglineNA : 표어 유무 /// 0 = 표어X, 1 = 표어O
- Cnt\_keywords : 영화 키워드 수
- Keywords\_cluster : 영화 키워드 군집 Onehotencoding // 1 - 13
- MaincastSum : 주연 배우가 출연한 영화 수의 합
- DirectorSum : 감독이 촬영한 영화의 수의 합
- DirectorMean : 감독이 만든 영화 수의 평균
- DirectorMax : 감독이 만든 영화 수 중 최댓값
- WriterSum : 작가가 쓴 영화 수의 합
- WriterMean : 작가가 쓴 영화 수의 평균
- WriterMax : 작가가 쓴 영화 수의 최댓값

# 3 Modelin g



# 모델링

5명의 팀원이 각각 모델을 맡아 오차값이 가장 적은 결과를 탐색.

- Linear Regression
- Support Vector Regressor
- Polynomial Regression
- KNN Regression
- RandomForest
- 평가기준 : rmsle

```
new_f.shape
```

```
(3000, 109)
```

```
new_f.isnull().sum()
```

```
budget          0  
popularity      0  
runtime         0  
revenue         0  
IscollectionNA  0  
..  
DirectorMax     0  
WriterSum       0  
WriterMean      0  
WriterMax       0  
revisedBudget   0  
Length: 109, dtype: int64
```

# Linear Regression

선형회귀 진행 순서

- Forward stepwise 방식으로 칼럼 골라내기
- 스케일링 적용
- Ridge, Lasso 적용
- 차원축소(PCA) 적용
- 변형된 budget 과 revenue 사용해보기

# Linear Regression

Forward Stepwise Regression 방식 (순차적 칼럼 추가)

	budget	runtime	popularity	count_language
0	14000000	93.0	6.575393	1
1	40000000	113.0	8.248895	1
2	3300000	105.0	64.299990	1
3	1200000	122.0	3.174936	2
4	0	118.0	1.148070	1
...	...	...	...	...
2995	0	102.0	9.853270	1
2996	0	102.0	3.727996	1
2997	65000000	120.0	14.482345	1
2998	42000000	90.0	15.725542	1
2999	35000000	106.0	10.512109	1



# Linear Regression

의미 있었던 칼럼 : 63개

예산  
인기도  
시리즈 여부  
홈페이지 여부  
개봉 연도, 요일  
장르, 장르 군집  
등

119

의미 없었던 칼럼 : 56개

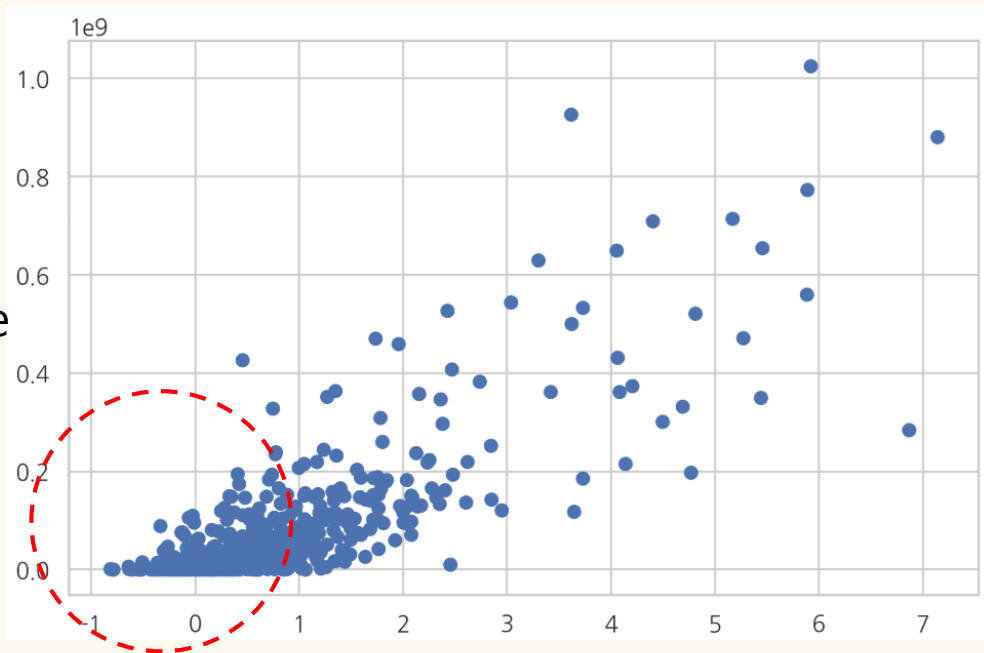
개봉 계절, 월  
참여회사 수  
번역 언어 수  
원본 언어  
작가  
장르(DBSCAN)  
등

# Linear Regression

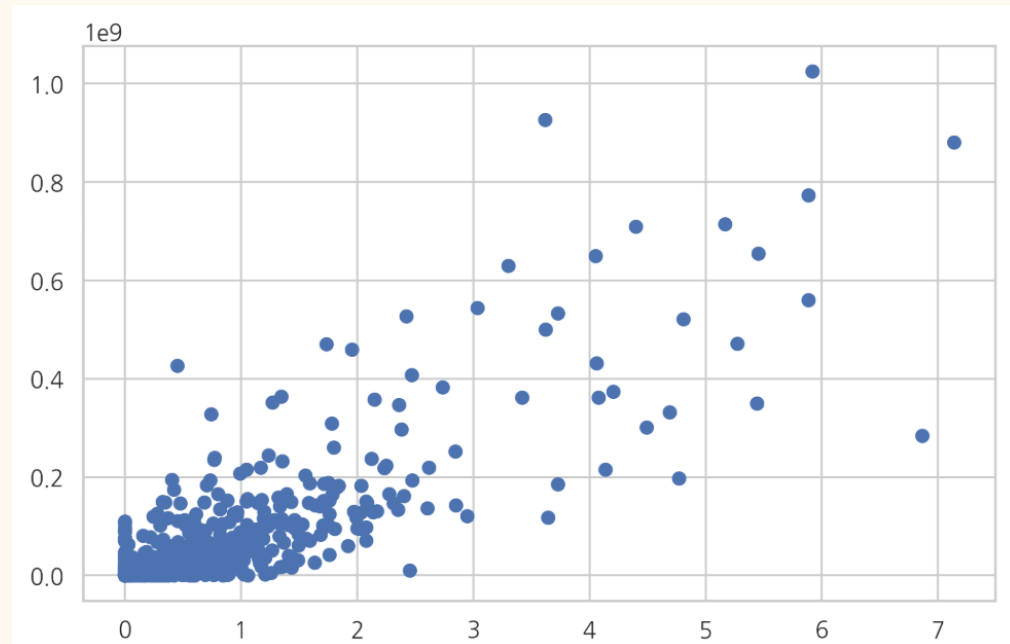
## RMSLE 계산

(-)Revenue 로 인해 ERROR

실제  
Revenue



(-)Revenue들 모두 1로 조정



예측 Revenue

# Linear Regression

기본 점수 (스케일링, ridge, lasso 등 미적용)

R 스퀘어

0.676

RMSE

80410943

RMSLE

2.65

Budget 칼럼만 적용시

0.55

```
rmsle_scores = np.sqrt(-scores)
print(rmsle_scores.mean())
```

2.6550141800836733

# Linear Regression

스케일러 / Ridge, Lasso

스케일러 사용  
(standard)

```
In [144]: # 스케일링 한거  
          lin.fit(sx,y)  
          lin.score(sx,y)  
  
Out [144]: 0.675791635854329
```

안 한게 점수가 좋음

Ridge, Lasso 사용  
(스케일러 적용)

```
In [146]: lasso = Lasso().fit(sx, y)  
          lasso.score(sx, y)  
  
Out [146]: 0.6760925858803627  
  
In [147]: ridge = Ridge().fit(sx, y)  
          ridge.score(sx, y)  
  
Out [147]: 0.6760925114249147
```

근소하게 안 좋음  
(미적용시 점수 0.6760925858803687)

# Polynomial Regression

- 2차 다항회귀 사용
- Recursive feature elimination함. (칼럼 88개 사용)  
(모두 포함한 후 중요도가 낮은 변수를 하나씩 제거)
- Bagging regressor 사용
- 8:2로 train\_test\_split

# Polynomial Regression

최종 점수

R 스퀘어

0.74

(Bagging으로 인한 변동)

RMSLE

2.41

특이사항

Revenue, Budget에 log 취해보도 별 차이 없음

스케일링 했으나 별 차이 없음 (Robust, MinMax, Standard)

# Support Vector Regressor

- Train/Test set을 8:2로 분리
- Standard scaler를 이용
- Grid search를 이용하여 모델의 하이퍼 파라미터를 결정

Kernel	$\epsilon$	C	degree
Linear			-
Poly	0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0	0.01, 0.1, 1, 10, 100, 1000	2, 3, 5, 7, 10
Rbf, Sigmoid			-

- Train set을 가지고 모델을 훈련
- Test set에 대해서 모델을 평가( $R^2$ , RMSLE)



# Support Vector Regressor

- Input features

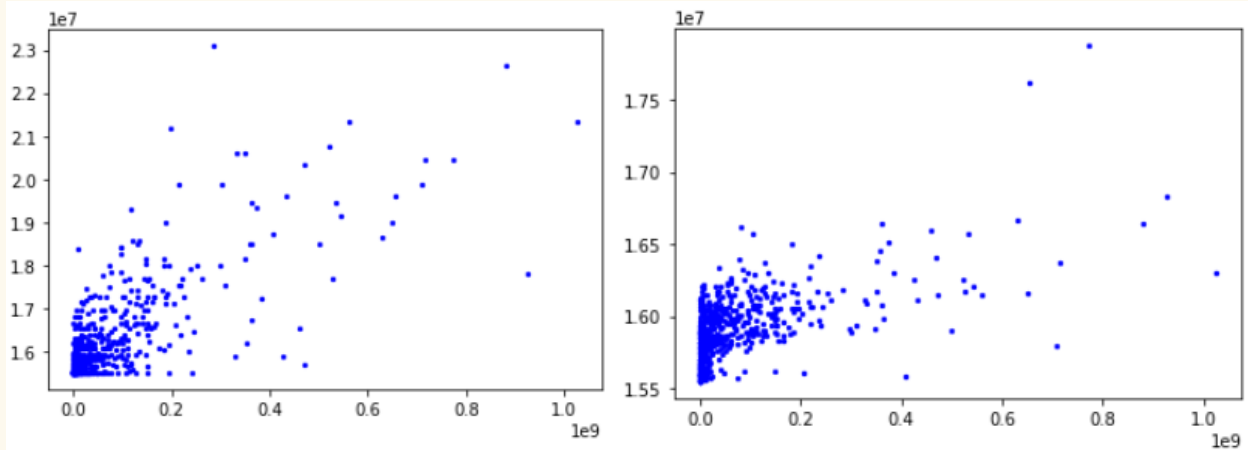
1	budget
2	budget, release_year, popularity
3	budget, release_year, IscollectionNA, IshompageNA
4	budget, release_year, Cnt_spok_language, Cnt_prod_companies, MaincastSum, DirectorMean, WriterMean
5	budget, release_year, popularity, Cnt_spok_language, Cnt_prod_companies, DBSCAN_0, DBSCAN_1, DBSCAN_2, DBSCAN_3, MaincastSum, DirectorMean, WriterMean
6	budget, runtime, popularity, IscollectionNA, IshompageNA, en, cn, fr, ja, hi, it, de, zh, others, Cnt_spok_language, Cnt_prod_companies, prod_company_weight2, Cnt_prod_countries, prod_country_weight2, release_year, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Sun, Mon, Tue, Wen, Thu, Fri, Sat, IsTaglineNA, Cnt_Keywords, MaincastSum, DirectorSum, DirectorMean, DirectorMax,

# Support Vector Regressor

## 1) Linear

Input1: budget  
 $R^2 = 0.57$ , RMSLE = 2.99

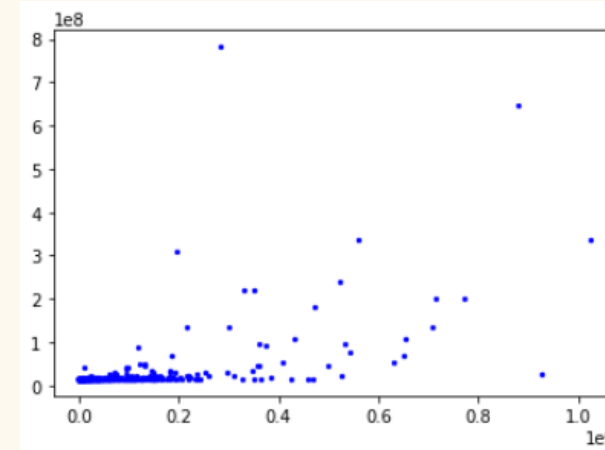
Input6  
 $R^2 = 0.36$ , RMSLE = 3.01



Best:  $C = 1000$ ,  $\epsilon = 0.01$

## 2) Poly

Input1: budget  
 $R^2 = 0.30$ , RMSLE = 2.90



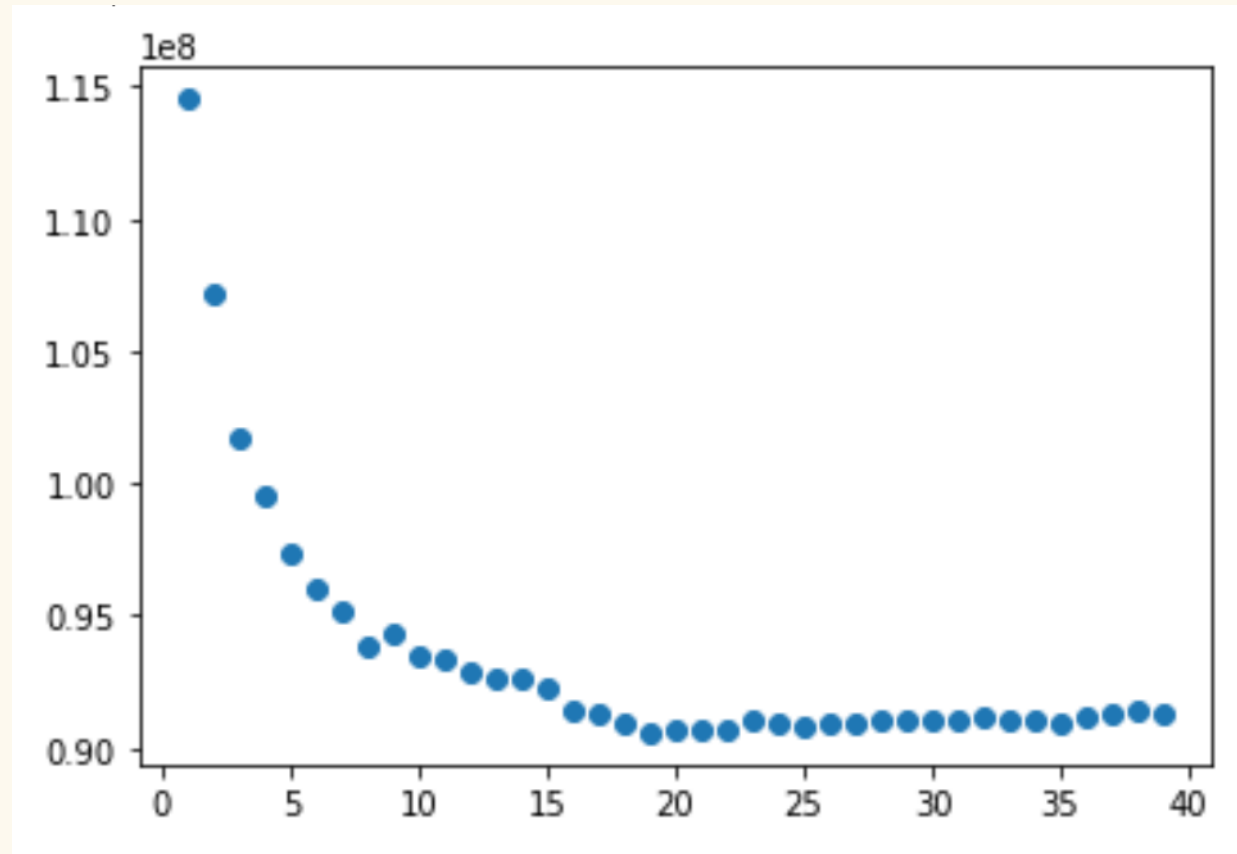
Best:  $C = 1000$ ,  $\epsilon = 2$

다른 입력세트에 대해서는  
 모델 예측이 거의 안됨

- Rbf와 sigmoid 커널을 적용하여도, RMSLE가 3이하로 줄지않았고, 예측 또한 정확도가 크게 떨어졌다.
- 결과적으로 budget 특성만 사용하여 LinearSVR을 했을 때 가장 좋은 결과를 얻음
- Input 특성 혹은 하이퍼 파라미터의 범위 선택에 문제가 있었을 것으로 예상

# KNN Regression

- 파라미터 – 탐색할 이웃 수( $k$ ) / 거리 측정 방법
- Small  $k \rightarrow$  overfitting
- Large  $k \rightarrow$  underfitting



# KNN Regression

- 특성 선택

```
from itertools import combinations
cc = list(combinations(InputDataXb.columns, 2))
for j in range(len(cc)):
    InputDataXCombination = pd.DataFrame(InputDataXb, columns=cc[j], copy = False)
    InputDataXCombination = pd.concat([InputDataXa, InputDataXCombination], axis = 1)
```

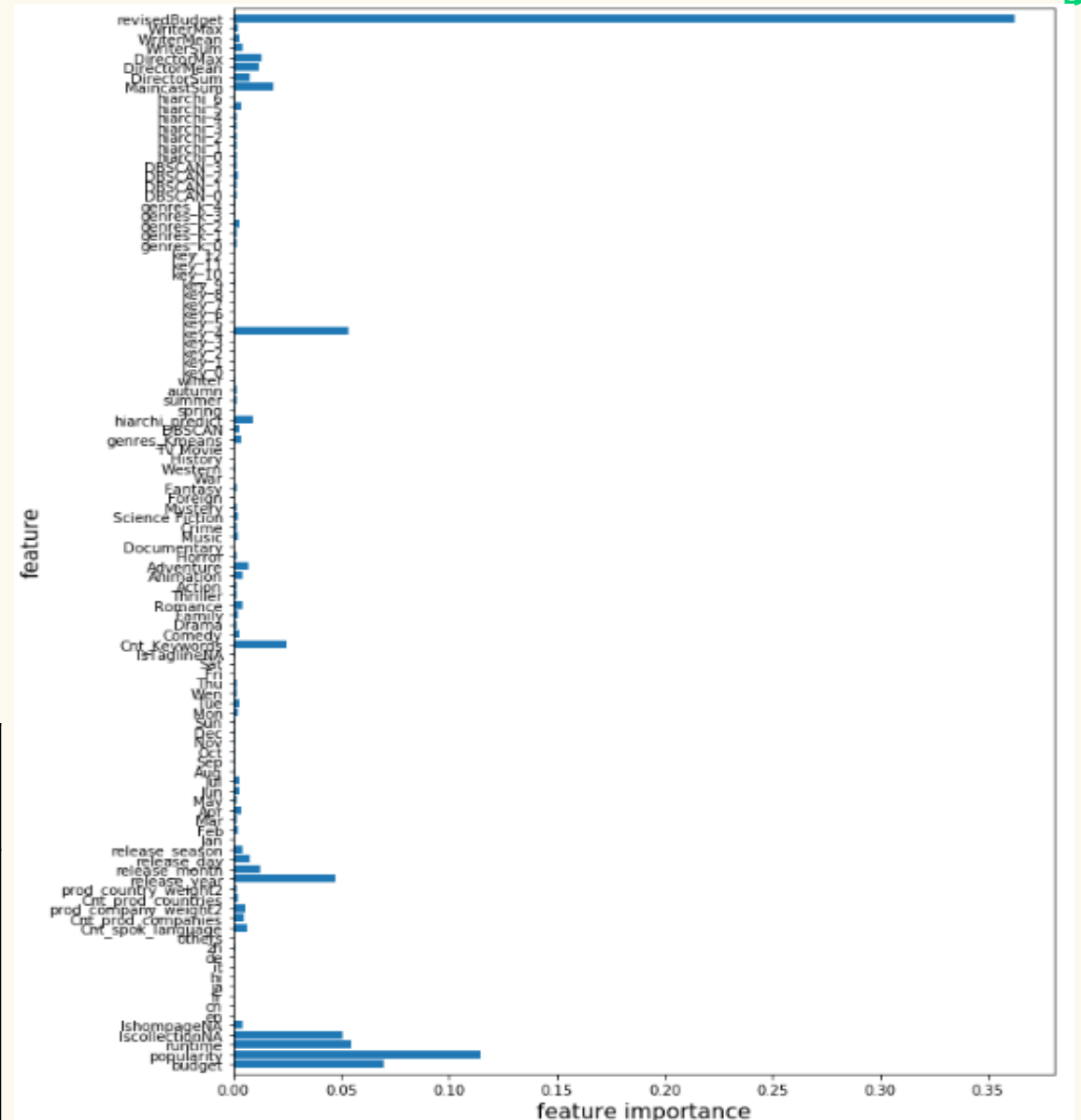
```
'budget', 'IscollectionNA', 'DBSCAN_1', 'DBSCAN_2', 'TV Movie', 'Horror', 'Foreign'
```

RMSLE: 2.652

# Random Forest

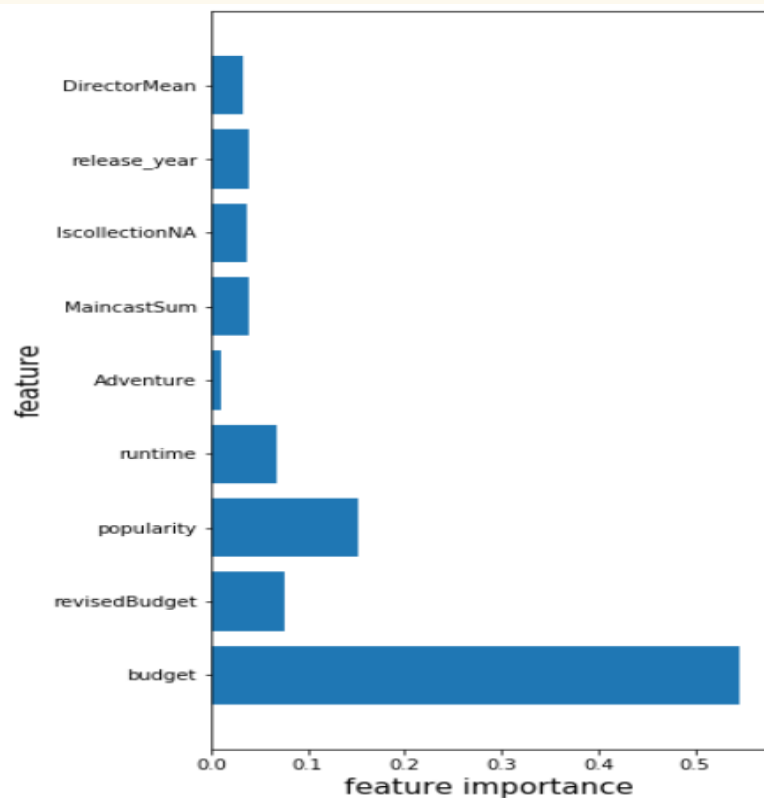
- Train/Test set을 8:2로 분리
- n\_estimators = 100
- Standard Scaler를 이용
- cross\_validation 사용 X
- y = revenue

n_estimators	Scaler	cross validation	y
10 100 150 200	MinMax Scaler Standard Scaler X	10 X	revised Revenue revenue

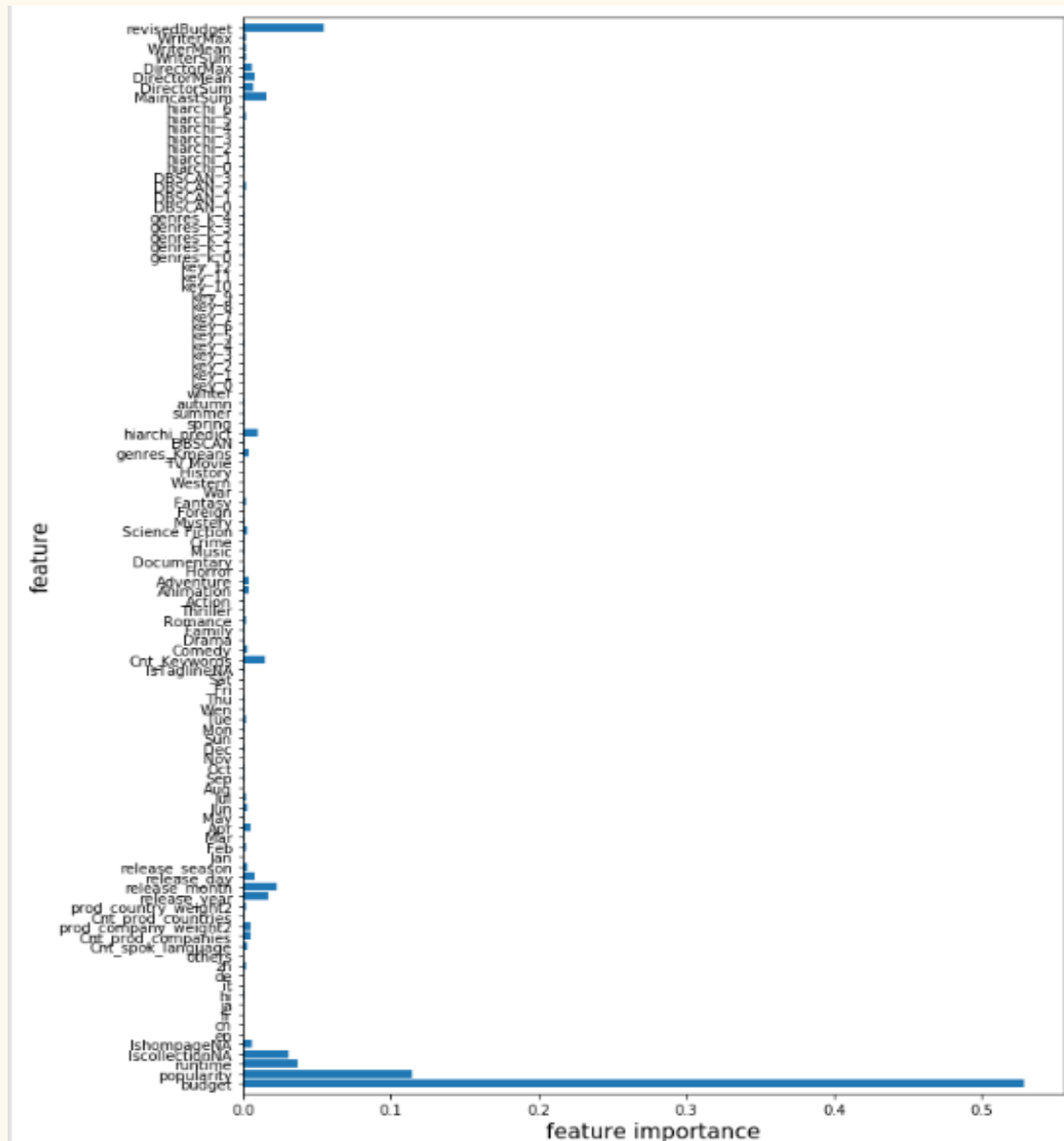


# Random Forest

- 전체 Feature를 사용.

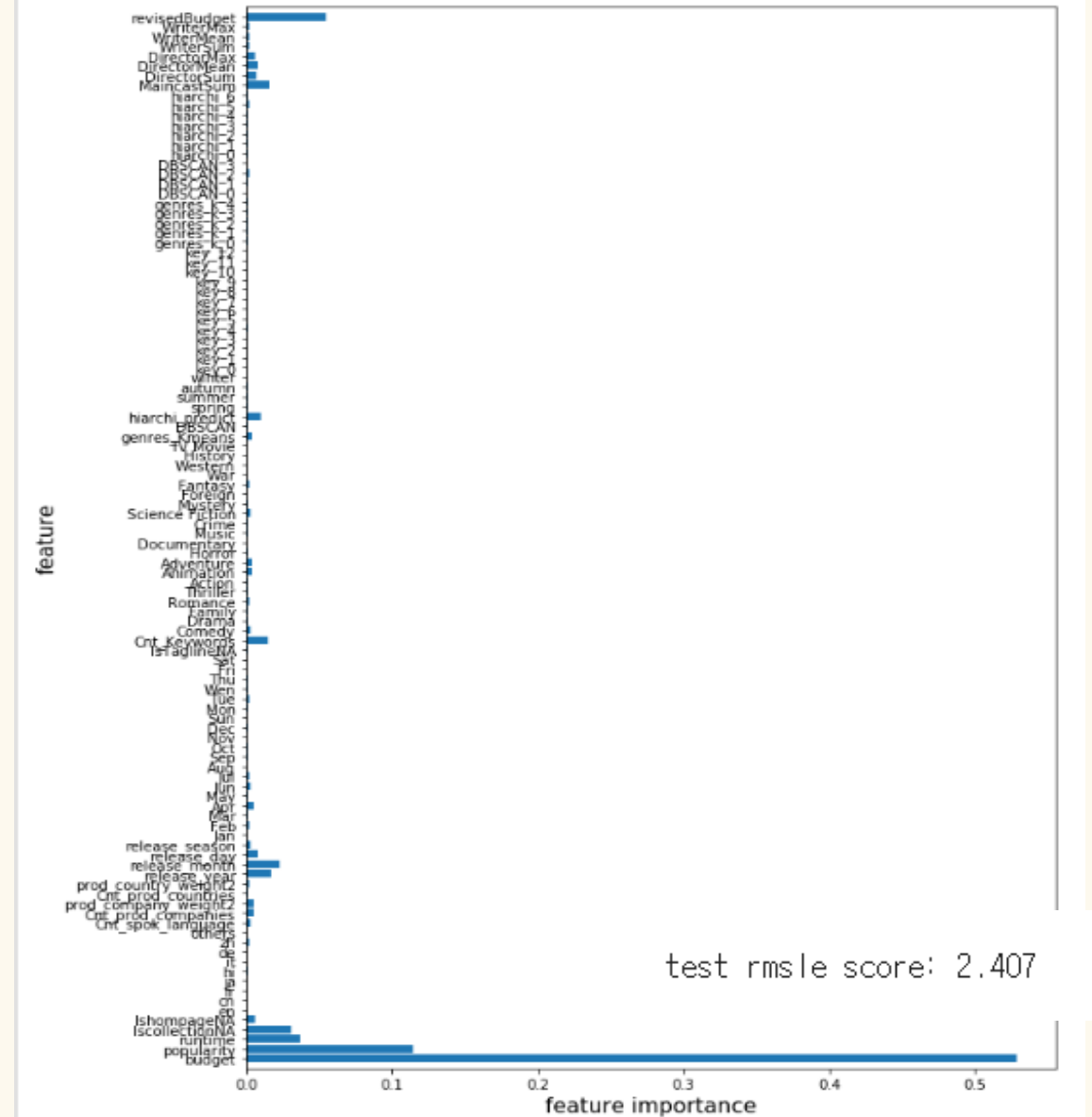


test rmsle score: 2.383



# Random Forest

- rmsle score : 2.407
- test set의 예측결과를 새로운 features로 추가 후 csv파일 생성





# 4 Result



# 모델 결합

- **rmsle score**

KNN Regression	linear regression	PolynomialRegression	SVR LinearRegression	RandomForest
2.65	2.57	2.41	2.99	2.41

- 총 5개의 모델 결과값을 사용해 모델을 결합한 후,  
 앞의 Test set(600개)를 사용해 최종 예측

# 모델 결합

## • 결합 방법

### 1. Ensemble

### 2. Random Forest

1) 모델의 예측값

2) 전체 Features + 예측값

WriterMax	revisedBudget	revenue	RF_prediot	Poly_prediot	SVR_prediot	line_prediot	knn_prediot
1	5.583444e+06	9247881	3.339965e+06	6906272.9	1.562176e+07	5.076366e+06	6695578.3
1	0.000000e+00	7	6.639283e+06	9343689.1	1.551988e+07	0.000000e+00	8933799.6
0	2.842403e+06	33700	1.993595e+06	2340898.4	1.557810e+07	1.576381e+07	13586217.2
0	1.683470e+08	10017322	1.286798e+08	94502435.0	1.837242e+07	2.451384e+08	254624717.7
1	0.000000e+00	8910819	3.716956e+06	17467935.7	1.551988e+07	0.000000e+00	31438439.9
0	5.914382e+07	141069860	9.755452e+07	117963291.4	1.668418e+07	1.335422e+08	78510007.7
0	1.455461e+07	68572378	2.539862e+07	25627610.0	1.589828e+07	2.806181e+07	40836870.8
0	4.163132e+07	25000000	4.500909e+07	26341261.9	1.578185e+07	8.377913e+07	6448008.0
0	2.133247e+07	40485039	7.127620e+07	58084561.3	1.581096e+07	1.409353e+08	48674733.9
1	2.700295e+07	67192859	9.463372e+07	61201191.6	1.610203e+07	1.074240e+08	56642591.5
0	1.346104e+07	53191886	3.947444e+07	36631305.2	1.584006e+07	3.498369e+07	15512144.2
1	0.000000e+00	11000000	6.839870e+06	8845008.3	1.551988e+07	1.986871e+07	8933799.6
0	0.000000e+00	55240	1.226779e+07	40311756.2	1.551988e+07	1.632492e+07	6634983.9
1	3.748639e+06	2500000	1.482983e+07	26043744.1	1.552716e+07	5.420244e+06	3761342.8

Random Forest(2) data set

# 모델 결합

- **Ensemble 결과**
  - rmsle score : 2.60
  - 5개의 모델의 결과값을  
평균한 값과 동일한 결과
  - 결론 : 오차 최소화에 부적합

	SVR_prediot	Poly_prediot	line_prediot	knn_prediot	RF_prediot
0	1.562176e+07	6906272.9	5.076366e+06	6695578.3	3.339965e+06
1	1.551988e+07	9343689.1	0.000000e+00	8933799.6	6.639283e+06
2	1.557810e+07	2340898.4	1.576381e+07	13586217.2	1.993595e+06
3	1.837242e+07	94502435.0	2.451384e+08	254624717.7	1.286798e+08
4	1.551988e+07	17467935.7	0.000000e+00	31438439.9	3.716956e+06
5	1.668418e+07	117963291.4	1.335422e+08	78510007.7	9.755452e+07
6	1.589828e+07	25627610.0	2.806181e+07	40836870.8	2.539862e+07
7	1.578185e+07	26341261.9	8.377913e+07	6448008.0	4.500909e+07
8	1.581096e+07	58084561.3	1.409353e+08	48674733.9	7.127620e+07
9	1.610203e+07	61201191.6	1.074240e+08	56642591.5	9.463372e+07

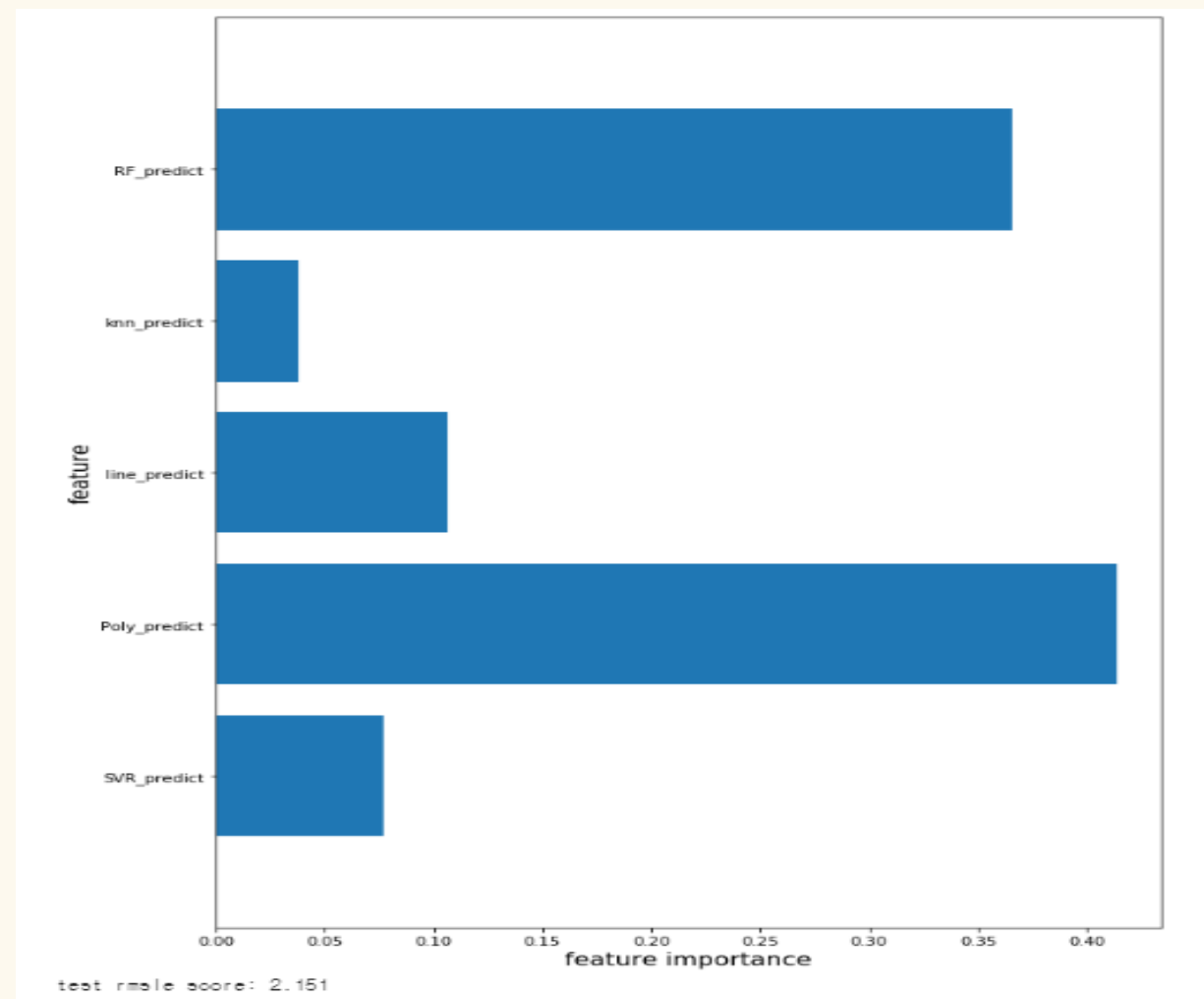
KNN Regression	linear regression	PolynomialRegression	SVR LinearRegression	RandomForest
2.65	2.57	2.41	2.99	2.41

# 모델 결합

- RandomForest(1) 결과

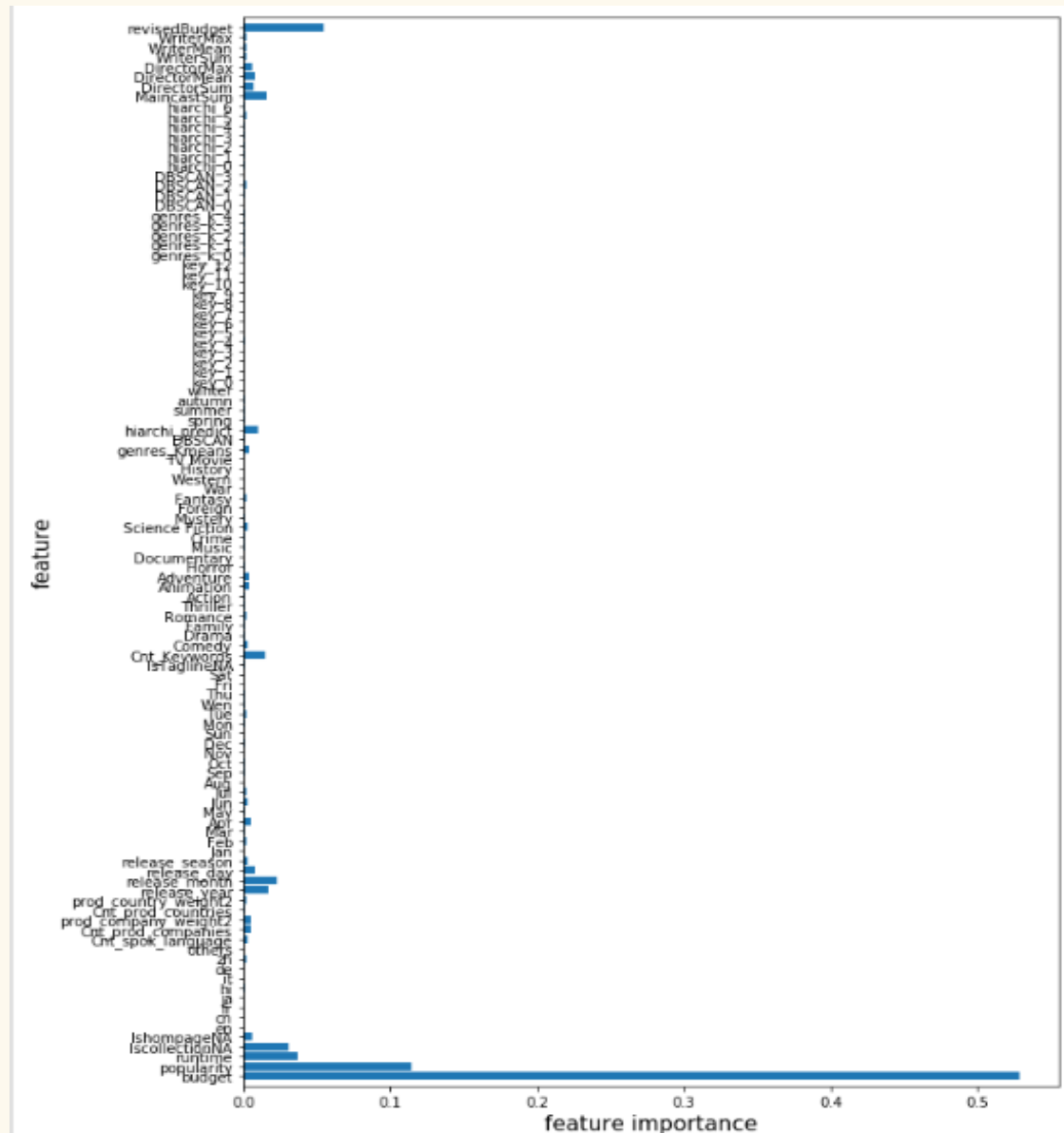
- rmsle score : 2.15

- ensemble의 결과보다 오차가 감소했지만, 최소값이 나오지 않음



# 모델 결합

- RandomForest(2) 결과
  - rmsle score : 2.12
  - 모델의 예측값만 사용한 (1) 보다 전체 Features 에 예측값을 더한 결과가 오차가 더 감소함.



# 최종 결과

• rmsle score : 2.123

<결과 요인>

1. 평가기준 rmsle 을 고려해 budget, revenue를 log scale로 분석해보지 못한점
2. 자연어 컬럼(tagline, title 등), 클래스 변수(gender 등) 버린 점
3. length, sum, conut, min, max를 칼럼별로 해보지 못한 점
4. Onehot인코딩으로 너무 늘어난 칼럼 개수
5. 장르+계절, 장르+감독, 키워드+배우 등 분류한 칼럼을 접목시키지 못한 점
6. 성능이 좋은 모델링들을 더 사용해보지 못한 점



**Thank  
You**





# 보완사항

## <전처리>

1. 가장 영향력이 높은 Budget의 0인 값을 10년 단위로 평균값을 계산해 채운다.
2. 평가기준(RMSLE)을 고려해 budget을 log scale로 분석해본다.

## <차원축소>

1. PCA, KPCA 사용해보기

## <모델링>

1. 전체 모델링에 각각 Scaler 사용, Cross validation 사용해보기
2. XGBoosting , LightGBM 사용해보기
3. 2-Stage modeling 사용해보기

# Budget

1. 가장 영향력이 높은 Budget의 0인 값을 10년 단위로 평균값을 계산해 채운다.

## Linear Regression

before 2.6550141800836733

After 2.825685717353439

## Polynomial Regression

before RMSLE : 2.365723710164881

After RMSLE : 2.45758339844723

## XGBoost

before 5.74

After 5.58

## LightGBM

before 2.42

After 2.44

## Random Forest

before test rmsle score: 2.823

After test rmsle score: 2.881

# Budget

2. 평가기준 rmsle 을 고려해 budget을 log scale로 분석해본다.

## Linear Regression

before 2.6550141800836733

After 2.7014866043790486

## Polynomial Regression

before RMSLE : 2.365723710164881

After RMSLE : 2.475292988212593

## XGBoost

before 2.33

After 2.35

## LightGBM

before 2.42

After 2.41

## Random Forest

before test rmsle score: 2.834

After test rmsle score: 2.889

# 차원 축소

## 1. PCA

### Scaled(Minmax)

55개의 component로 설명 가능  
(분산 설명력 95.8%)

하지만 결과가 매우 좋지 않았다.  
(Poly- Regression의 R-squared)

Train 0.8900150941659914  
Test 0.3411340896172331

### Not - Scaled

1개의 component로 설명 가능  
(97%)

하지만 역시 결과가 좋지 않았다.  
(Poly-Regression의 R-squared),  
Linear-Regression의 R-squared

Train 0.8231168725395394  
Test 0.21868205578030264

Train 0.9286341665935001  
Test 0.6647720739413802  
crossvalidationcores : [0.61395072 0.67866318 0.62949846 0.70147065 0.67975445]  
RMSLE : 2.365723710164881

# 차원 축소

## 2. KPCA

Train	0.9351529064467807
Test	0.6419057136591135
	crossvalidationcores : [0.63082047 0.68276183 0.58618295 0.68770176 0.66387541]
	RMSLE : 2.365723710164881

# 모델링

## 1. XGBoost

	Train RMSLE	Train $R^2$	Test RMSLE
Budget	5.74	0.64	5.44
Fixed_budget	5.58	0.65	5.35
Log_budget	2.33	0.25	2.38
Log_fixed_budget	2.35	0.26	2.39

- Budget의 0 값을 연도 구간별 평균으로 채워준 Fixed\_budget을 사용했을 때, RMSLE가 조금 감소함
- Log scale로 금액을 분석하였을 때, RMSLE가 크게 감소함
- Log scale의 경우 크게 예측된 몇 개의 값으로 인해  $R^2$ 가 크게 감소함

# 모델링

## 2. LightGBM

	Train RMSLE	Train $R^2$	Test RMSLE
Budget	2.42	0.757	4.24
Fixed_budget	2.44	0.764	4.13
Log_budget	2.41	0.76	4.28
Log_fixed_budget	2.442	0.764	4.13

- Train 과 Test의 RMSLE 값을 비교했을 때 상당한 차이가 나타남.
- Budget의 종류를 바꿨을 때 큰 차이를 보이지 않음.

# 모델링

## 3. 2-Stage modeling

### 각 모델별최종 kaggle 결과

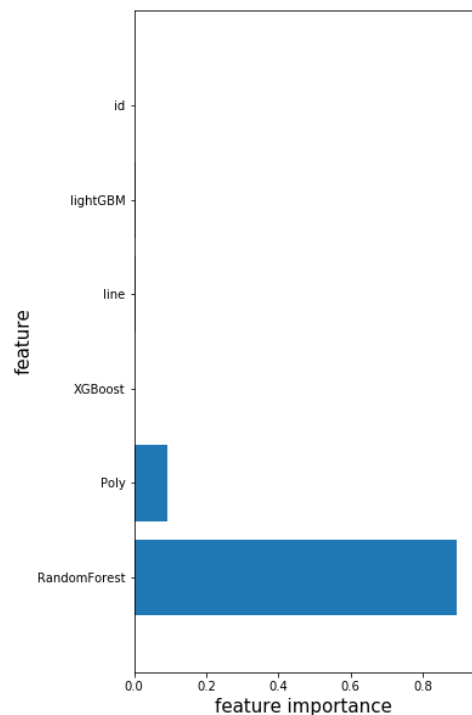
LightGBM	linear regression	PolynomialRegression	XGBoost	RandomForest
4.13	7.24	3.97	2.38	2.60



# 모델링

## 3. 2-Stage modeling (5개의 모델 예측값 사용시)

[24]  
 훈련 세트 정확도 : 0.997  
 테스트 세트 정확도 : 0.702



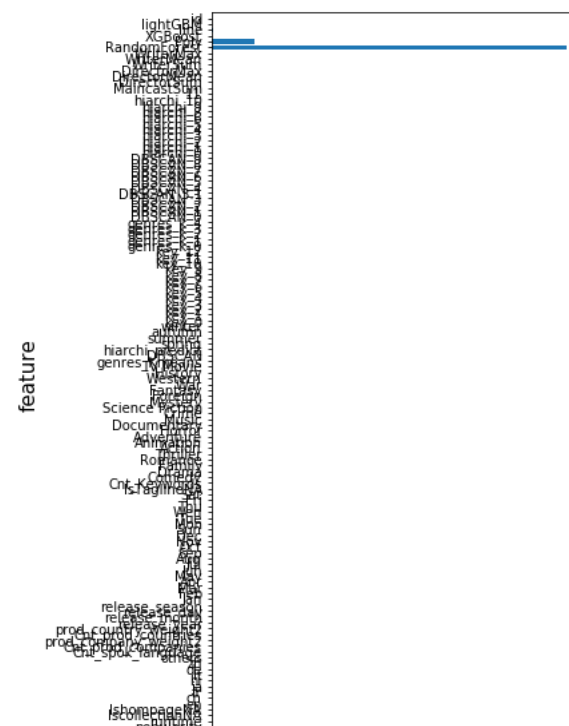
**5개 모델링을  
RandomForest 돌린경우**  
 rmsle :2.702

Name	Submitted	Wait time	Execution time	Score
two_predict1 (1).csv	just now	1 seconds	0 seconds	2.70205

Complete

[Jump to your position on the leaderboard](#)

훈련 세트 정확도 : 0.996  
 테스트 세트 정확도 : 0.718



**5개 모델링과 전체를  
RandomForest 돌린경우**  
 rmsle :2.469

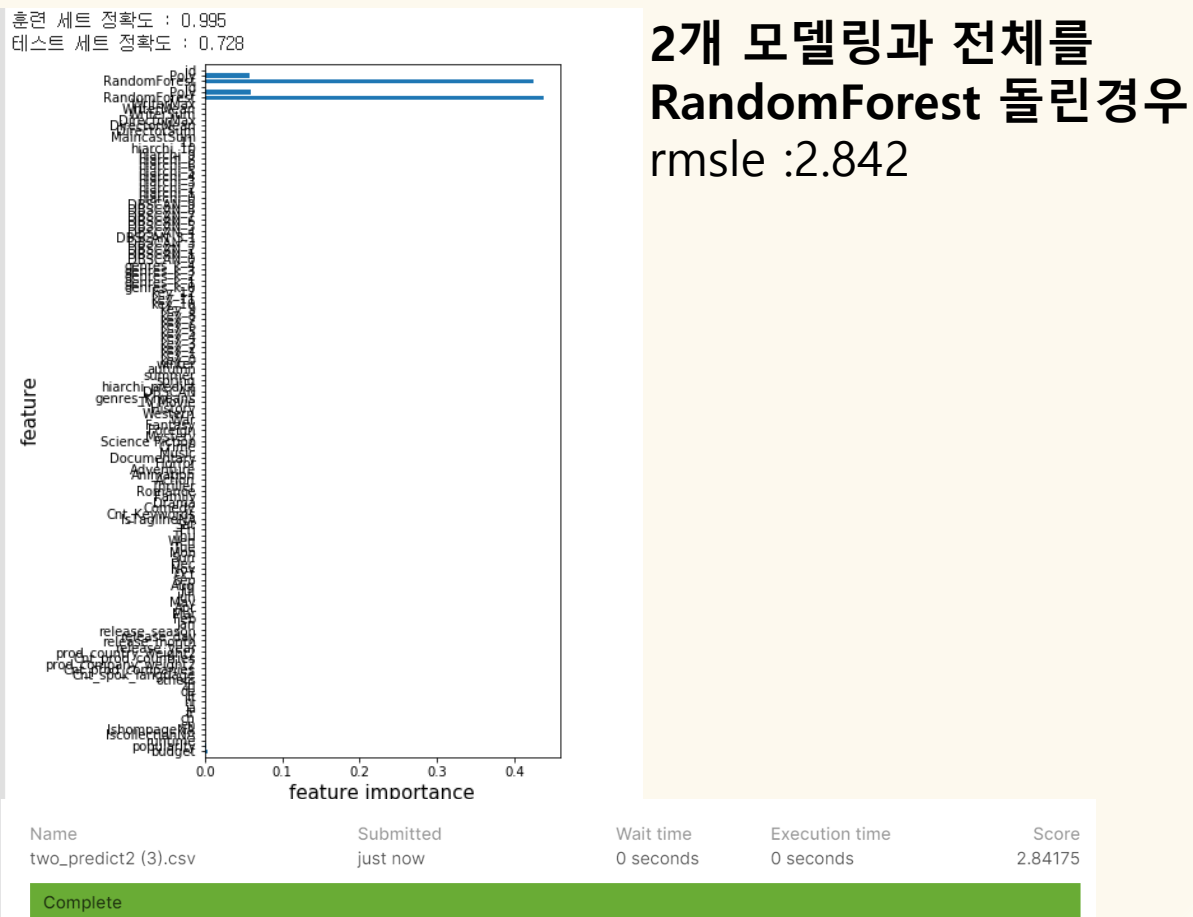
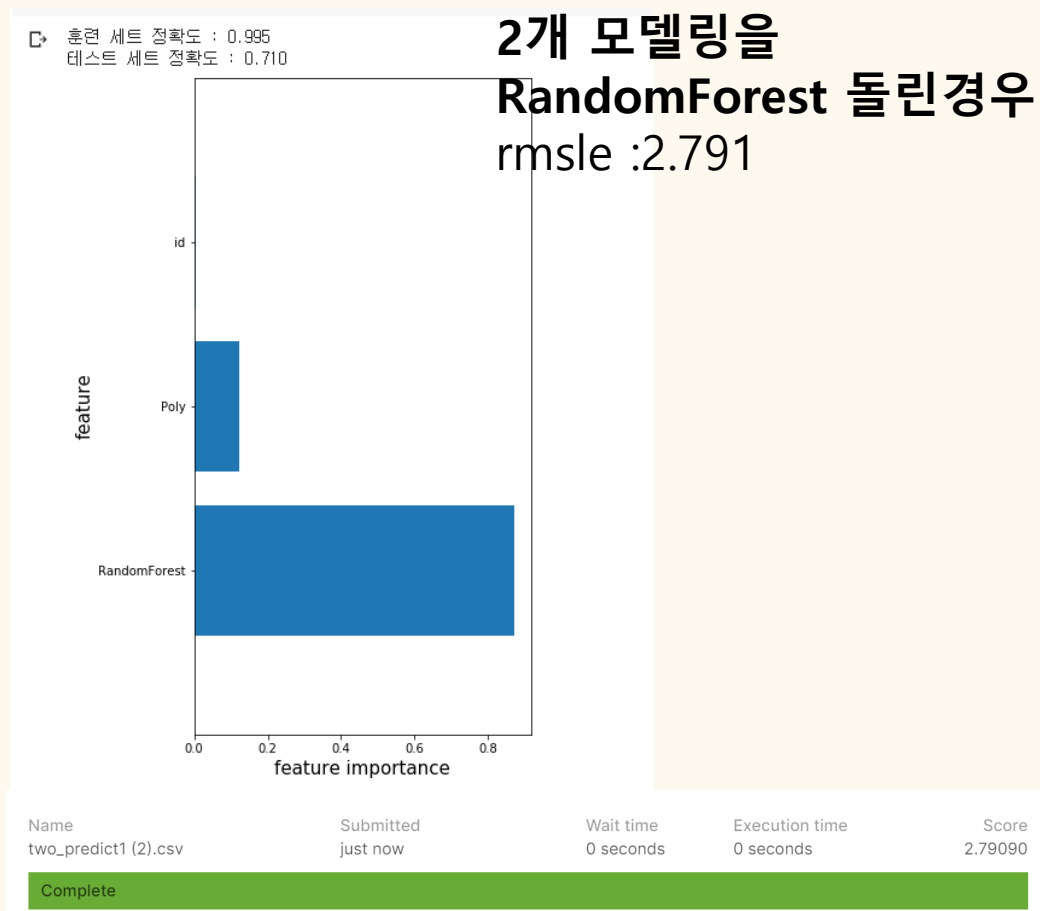
Name	Submitted	Wait time	Execution time	Score
two_predict2.csv	just now	0 seconds	0 seconds	2.46868

Complete

[Jump to your position on the leaderboard](#)

# 모델링

## 3. 2-Stage modeling (Importance Top2의 수익 예측값 사용시)



# 모델링

## 3-3. Ensemble

```

y1 = df['revenue']

new_df = df.drop(['revenue'], axis =1,inplace=False )
x = new_df
feature_names = x.columns.tolist() #항목 이름들
X_train, X_test, y_train, y_test = train_test_split(x, y1, test_size=0.2, random_state = 2)

#Standard Scaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x)
X_scaled = scaler.transform(x)
x = X_scaled

# Training classifiers
reg1 = xgb.XGBRegressor(booster='gblinear', objective='reg:squarederror', early_stopping_round=10)
reg2 = RandomForestRegressor(random_state=1, n_estimators=10)
reg3 = LinearRegression()
er = VotingRegressor([('xgb', reg1), ('rf', reg2), ('lr', reg3)])

kfold = KFold(n_splits=3) # KFold 객체 생성

#cross validation
scores = cross_val_score(er, X_train,y_train,cv =kfold)

er.fit(X_train,y_train)
print("훈련 세트 정확도 : {:.3f}".format(er.score(X_train,y_train)))
print("테스트 세트 정확도 : {:.3f}".format(er.score(X_test,y_test)))

```

```

훈련 세트 정확도 : 0.813
테스트 세트 정확도 : 0.660
[ 28005434. 50515537 -10039087. 81629582 19411140. 07601851 ...
 64643967. 05332165 47768261. 0003391 19887933. 83452165]

```

## Ensemble voting regressor 결과

rmsle :

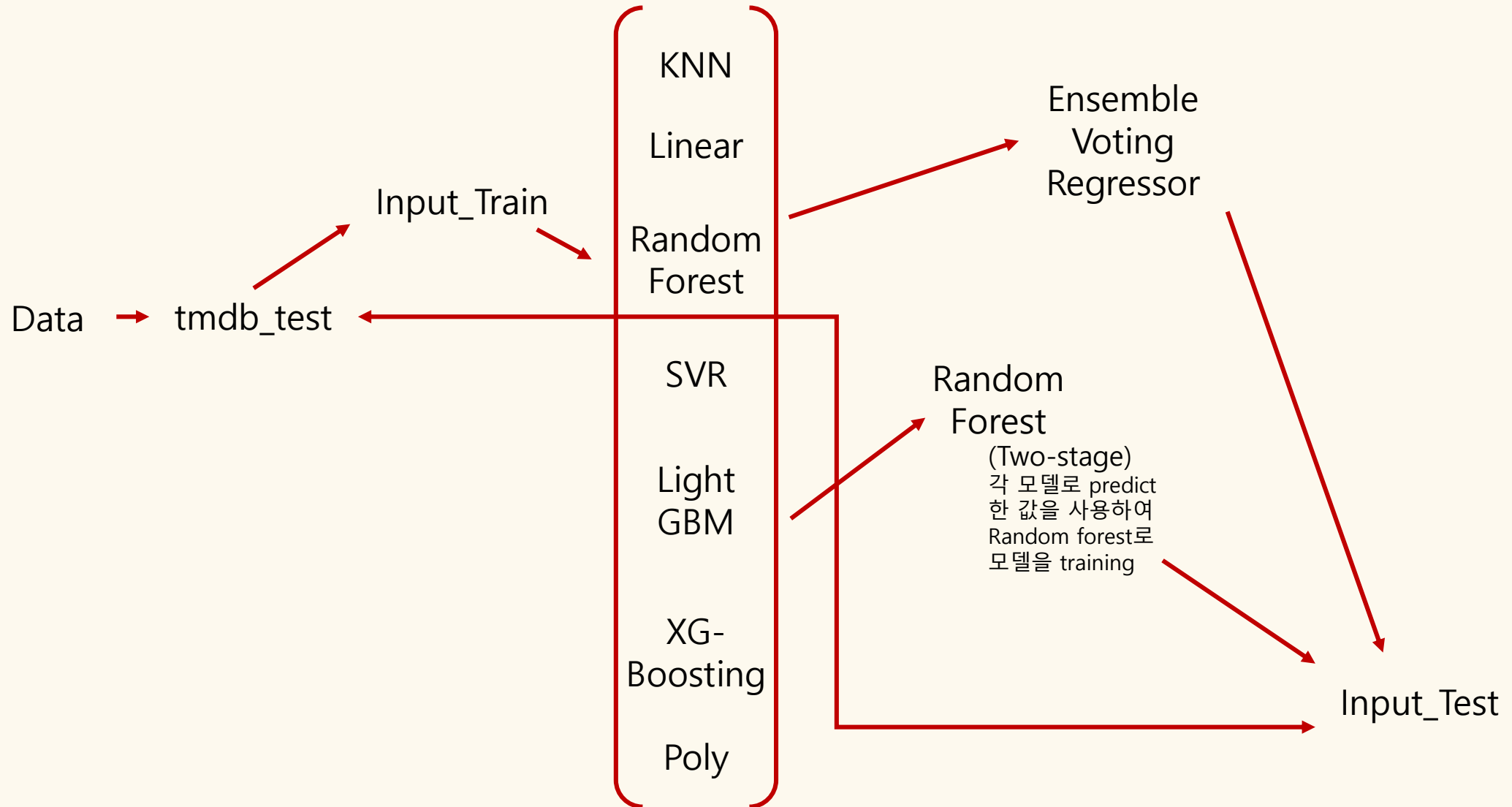
Your most recent submission

Name	Submitted	Wait time	Execution time	Score
en_predict (2).csv	just now	0 seconds	0 seconds	5.97592

Complete

[Jump to your position on the leaderboard ▼](#)

# 모델링 과정



# 보완 후 최종결과

- Revenue와 큰 상관관계를 가지는 Budget의 0 값이 800개나 되어 그 부분을 수정해주는 것이 큰 효과가 있을 것으로 기대했으나, 결과에 크게 영향을 끼치지 못하였다.
- 차원 축소를 적용한 입력 자료 또한 결과를 크게 개선시키지 못했다.