회귀해석 프로젝트

한국 박스오피스 기준, 영화 흥행 기준에 영향을 주는 요소에 관한 연구

동국대학교 통계학과

ㅇㅇㅇ, 이 윤 정

20200625

목 차

제1장 서 론

제1절 연구의 목적

영화산업은 문화 산업적 성격을 가지고 있는 21세기의 고부가가치 산업 중 하나이다.[1] 2019년 전체 극장 관객 수는 2억 2668만 명으로 전년 대비 4.8% 증가하여 역대 최고 관객 수를 달성했고, 한국 영화는 2012년 이후 8년 연속 1억 관객 돌파를 달성하게 되었다.[2] 그러나 2019년 개봉한 '상업영화' 45편 중 매출액 1위의 영화를 제외한 44편의 평균 수익률을 산출한다면 수익률이 -8.1%로 적자를 보인다.[3] 이는 영화를 보는 사람들의 수는 점점 증가하지만 높은 손익분기점을 만족하기에는 수익이 부족하기 때문이다. 따라서 수익을 극대화하기 위해서는 흥행 예측 연구를 통하여 스크린 수 조정 혹은 마케팅 전략 수정 등의 과정이 필요하다. 그러므로 본 연구에서는 영화의 흥행에 영향을 주는 요인들을 조사하여 그로 구성된 흥행 예측 모델을 만들고자 한다.

제2절 연구의 범위 및 방법

본 연구에서 영화의 흥행을 판단할 수 있는 지표를 관객 수라고 설정한다.

1. 연구의 범위

본 연구의 범위는 다음과 같다.

○ 영화의 누적 관객 수와 수집한 요소들의 상관관계 도출

○ 앞서 선정한 요소들을 이용하여 누적 관객 수를 예측하는 모델 구현

- 누적 관객 수를 예측하는 모델 선정
- 누적 관객 수를 예측하는 모델 진단

2. 연구의 방법

본 연구의 방법은 다음과 같다.

○ 데이터 수집 및 정제 : 영화관입장권통합전산망, 네이버 영화, 다음

---

1) 최은영, "한국영화산업의 발전방향 분석", 한국콘텐츠학회논문지 8, p.134-143, 2008.11

2) 영화진흥위원회, 2019년 한국 영화산업 결산, p.9

3) 영화진흥위원회, 2019년 한국 영화산업 결산, p.13

영화, 공휴일 API를 이용하여 웹 크롤링 수행 후 필요한 변수 정제

- ㅇ 네이버 영화의 오픈 API를 이용하여 웹 크롤링 수행 후 필요한 변수 정제
- ㅇ 모델 선택 : 후진 제거법, 전진 선택법, 단계적 선택법을 이용하여 예측 모델에 적합한 변수 채택
- ㅇ 모델 진단 : 잔차 분석을 통하여 모델의 적합도를 검정한다

제2장 본 론

제1절 분석대상 및 데이터

1. 분석대상 선정

본 연구의 목적은 수집한 데이터를 이용하여 영화 관객 수에 영향을 미치는 요인을 분석한 후, 예측 모형을 구현하고 검증하는 것이다.

관객 수에 영향을 미치는 요인을 분석하기 위해, 영화 관객들의 관람 양상을 확인한 결과, 부록 [그림2-1]을 통해 하루 동안의 총 관객 수의 약 90%가 박스오피스 10위권 내의 영화를 관람한다는 사실과 부록 [그림 2-2~4]을 통해 개봉일로부터 3주 내의 관객 수가 전체 관객 수에 80%에 육박한다는 사실을 확인하였다. 그러므로, 본 연구에서 2014년부터 2018년에 개봉한 영화 중 일일 박스오피스 순위권에 일정 기간 이상 있었던 817편의 영화를 분석대상으로 선정하였다.

2. 데이터 수집 및 전처리

영화의 흥행 요인은 크게 영화의 내적 요인과 외적 요인으로 구분할 수 있으며, 본 연구에서는 영화의 흥행 요인을 크게 '영화 속성', '구전 효과', '경쟁 요소'로 나누었다. 해당 데이터는 포털 사이트의 영화 정보와 영화관입장권통합전산망, 공공데이터 포털 API에서 수집하였으며, 앞서 설정한 영화의 흥행 지표에 근거 하여 개봉 후 3주차까지의 관객 수를 종속변수로 설정하였다.

(1) 영화 속성

본 연구에서 영화의 흥행에 대한 내적 요인인 ‘영화 속성’을 분석하기 위해 영화관입장권통합전산망인 'KOBIS'에서 2014년부터 2018년까지 일별 박스오피스 상위 랭크에 있는 관객 수, 스크린 수, 배급사, 출연 배우, 감독, 장르, 국가, 관람등급의 데이터를 수집 csv로 저장하였다. 수집된 데이터에는 영화제목, 개봉일, 일일 스크린 수, 누적 스크린 수, 일일 관객 수, 누적 관객 수, 배우, 감독, 제작사, 장르, 국가, 관람 등급, 일일 매출액, 누적 매출액이 포함되어있다. 분석에서 사용하기 위하여 배우, 감독에 대하여 스코어를 부여하였다. 감독 스코어는 최근 3년 동안 동원한 누적 관객 수로 설정하였고, 배우 스코어는 기존 데이터의 배우들 중 주연급 배우 2명에 대하여 최근 3년 동안 동원한 누적 관객 수로 설정하였다. 장르와 국가, 관람등급은 범주형 변수이기 때문에 분석을 위해 더미 변수로 변환하였다.

(2) 구전 효과

본 연구에서 영화의 흥행에 대한 외적 요인인 ‘구전 효과’를 분석하기 위해 주요 포털 사이트에서 제공되는 분석 대상에 대한 영화별 코드를 크롤링하였다. 수집된 코드를 이용하여 포털 사이트 영화 페이지의 HTML에 속한 네티즌 평점과 전문가 평점, '보고싶어요'지수, 개봉 전 기사 수 또한 크롤링하여 csv로 저장하였다. 수집된 데이터에서 포털 사이트의 네티즌 평점과 전문가 평점의 평균은 각각 7.7615, 5.4995이며, [그림 2-5]를 통해 정규분포와 유사한 분포 형태를 취한다는 것을 볼 수 있다. 따라서, 3구간으로 나누어 범주형 변수로 변환 후 더미 변수로 설정하였다. 이때, 네티즌 평점은 3구간을 10점~8점, 8점~6점, 그 외로 설정하여 각각 ‘A’, ‘B’, ‘C’로 부여하였고, 전문가 평점은 3구간을 10~7, 7~5.5, 그 외로 설정하여 각각 ‘A’, ‘B’, ‘C’로 부여하였다.

(3) 경쟁 요소

외적 요인 중 ‘경쟁 요소’를 분석하기 위해 ‘KOBIS’에서 수집한 데이터들을 정제하여, 분석대상의 개봉 시기와 개봉 후 첫 번째 토요일 관객 수, 개봉 첫날 스크린 수, 배급사를 수집하였다. 개봉 시기는 개봉 월과 공휴일 API를 통해 개봉 후 일정 기간 내 공휴일 유무를 더미 변수로 설정하

였으며, 분석에 사용하기 위하여 배급사는 최근 3년 동안 동원한 누적 관객 수를 스코어로 부여하였다.

전처리를 거친 결과 데이터는 다음과 같은 변수를 가지게 된다.

<표 2-1> 변수의 정의

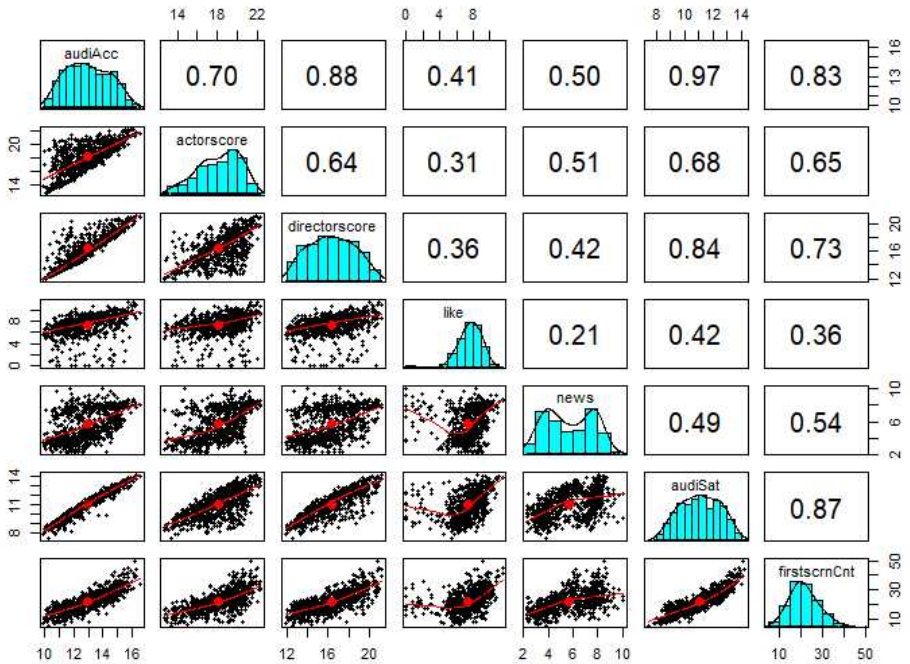
구분		변수(변수명)	추정방법	변수형태
독립변수	영화 속성	감독 (actorscore)	최근 3년 동안 감독 영화가 동원한 관객 수 누적	
		배우 (directorscore)	최근 3년 동안 주연 배우 2명 출연 영화가 동원한 관객 수 누적	
		국적 (nation)	한국 미국 기타	더미변수
		장르(genre)	코미디 어드벤쳐 액션스릴러 멜로 기타	더미변수
		관람등급(Grade)	전체관람가 12+ 15+ 18+	더미변수
	구전 효과	네이버 네티즌평점 (naverScore)	네이버의 네티즌 영화 평점 (A, B, C)	더미변수
		다음 네티즌평점 (daumScore)	다음의 네티즌 영화 평점 (A, B, C)	더미변수
		네이버 전문가평점 (naverspecialScore)	네이버의 전문가 영화 평점 (A, B, C)	더미변수
		다음 전문가평점 (daumspecialScore)	다음의 전문가 영화 평점 (A, B, C)	더미변수
		기사 수 (news)	개봉일 2주 전부터 개봉일까지 기사 수	
		'보고싶어요'지수 (movielike)	개봉 전 네티즌들이 '보고싶다'고 표시한 지수	
	경쟁 요소	개봉 월 (openmonth)	개봉일의 월	더미변수
		공휴일 유무 (redday)	개봉일 이후 2주 이내 공휴일 유무	더미변수
		개봉첫토요일 관객수 (audiSat)	개봉 후 첫 번째 토요일의 관객 수	
		개봉첫날스크린 수 (firstscrncnt)	개봉 첫날 동원한 스크린 수	
		배급사 (distributorscore)	최근 3년 동안 배급한 영화가 동원한 관객 수 누적	
종속변수		누적관객수(audiAcc)	개봉 후 3주 이내의 누적 관객 수	

제2절 회귀 모델 선택

1. 변수 선택

수집된 데이터를 이용하여 예측 모델을 선정하기 이전에 변수 선택을 진행하였다. 종속변수는 앞서 설정한 대로 누적 관객 수이고, 독립변수 선택을 위하여 우선 변수별 상관관계 비교를 진행하였다.

부록의 [그림 2-6]을 보면, 종속변수와 독립변수의 상관관계를 한눈에 볼 수 있다. 첫 번째 행에 위치한 산점도를 보면 대부분 선형보다는 지수함수의 형태를 취하고 있다. 이는 종속변수인 audiAcc의 단위 수가 매우 커서, 단위 수가 작은 다른 변수들과 상관분석을 실시할 경우 해당 형태를 보이기 때문이다. 이때, log 변환 혹은 제곱근 변환을 취하면 상대적으로 작은 값에 몰려 있던 데이터가 흩어지면서 좌우대칭의 형태로 분포의 모양이 변하게 된다. 부록의 [그림 2-7~8]에서 볼 수 있듯이 변환을 취하게 되면 변수별 선형관계를 확인할 수 있다.



[그림 2-9]

[그림 2-9]를 통해 log 변환 혹은 제곱근 변환을 취한 변수들의 상관관계를 한눈에 파악할 수 있다. log 변환을 취한 후, audiAcc에 대한 news의 상관계수가 0.27에서 0.5로 유의미하게 증가함을 알 수 있다.

2. 유의성 검정

변환된 변수들을 이용한 회귀 예측 모델을 선정하기 위하여 유의성 검정을 실시하였다. 후진 제거법, 전진삽입법, 단계 삽입법을 모두 시행한 결과 <표 2-2>와 같다. 더 자세한 유의성 검정 결과는 부록[그림 2-10~12]에 첨부하였다.

	R-squared	F-statistics	Adjusted R-squared	p-value
후진 제거	0.9658	1475 on 14 and 731 DF	0.9652	<2.2e-16
전진 삽입	0.9665	567.8 on 36 and 709 DF	0.9648	<2.2e-16
단계 삽입	0.9658	1475 on 14 and 731 DF	0.9652	<2.2e-16

<표 2-2>

유의성 검정은 다음의 (1)과 같은 회귀 모형을 설정하였다.

$$\log(Y_i) = \alpha_0 + \alpha_1 ACS_i + \alpha_2 DIS_i + \alpha_3 NEWS_i + \alpha_4 ADS_i + \alpha_5 FSC_i + \alpha_6 GEN\_T_i + \alpha_7 GEN\_A_i + \alpha_8 DMS\_A_i + \alpha_9 DMS\_B_i + \alpha_{10} NMS\_A_i + \alpha_{11} NMS\_B_i + \alpha_{12} OPM\_11_i + \alpha_{13} OPM\_1_i + \alpha_{14} OPM\_3_i + \epsilon_i \quad \cdots (1)$$

여기서, ACS_i = 영화i의 주연 배우 명성, DIS_i = 영화i의 감독 명성,
$NEWS_i$ = 영화i의 개봉 전 기사 수, ADS_i = 영화i의 개봉 주 토요일 관객 수,
FSC_i = 영화i의 첫날 스크린 수, $GEN\_장르_i$ = 각 장르에 대한 영화i의 더미변수,
$DMS\_등급_i$ = 영화i의 다음 네티즌 평점의 더미 변수,
$NMS\_등급_i$ = 영화i의 네이버 네티즌 평점의 더미 변수,
$OPM\_월_i$ = 영화i 개봉 월의 더미 변수, ϵ_i = 오차항

3. 다중공선성 검정

[그림2-13]을 통해 (1)의 회귀 모형의 독립변수 중 audiSat이 다중공선성을 만족한다는 것을 볼 수 있다. 따라서, 변수 audiSat을 제외한 모형을 채택하였다.

```
> vif(fit)
   actorscore directorscore          news       audiSat
     2.096459      3.846787      1.630246      7.067729
 firstscrnCnt     gen스릴러      gen액션     daumScoreA
     4.744275      1.036789      1.209847      4.125225
   daumScoreB   naverScoreA   naverScoreB openmonth11월
     3.305171      4.231951      3.354392      1.027269
openmonth1월  openmonth3월
     1.029697      1.035920
```

[그림2-13]

제3절 회귀 진단

회귀 분석을 시작할 때, 본 연구에서는 오차항과 함수에 대한 몇몇 "가정"을 가정하였다. 앞서 선정된 회귀 모형의 회귀계수 추론과 분석 결과, 해석은 모두 해당 "가정" 하에 진행된 연구이기 때문에 가정이 성립되지 않으면 분석 결과를 신뢰하기 어렵다. 따라서 회귀 모형에 대한 분석 결과를 신뢰하기 위하여 "가정"을 만족하는지 확인하기 위한 작업이 필요하다.

1. 이상치 & 영향치 진단

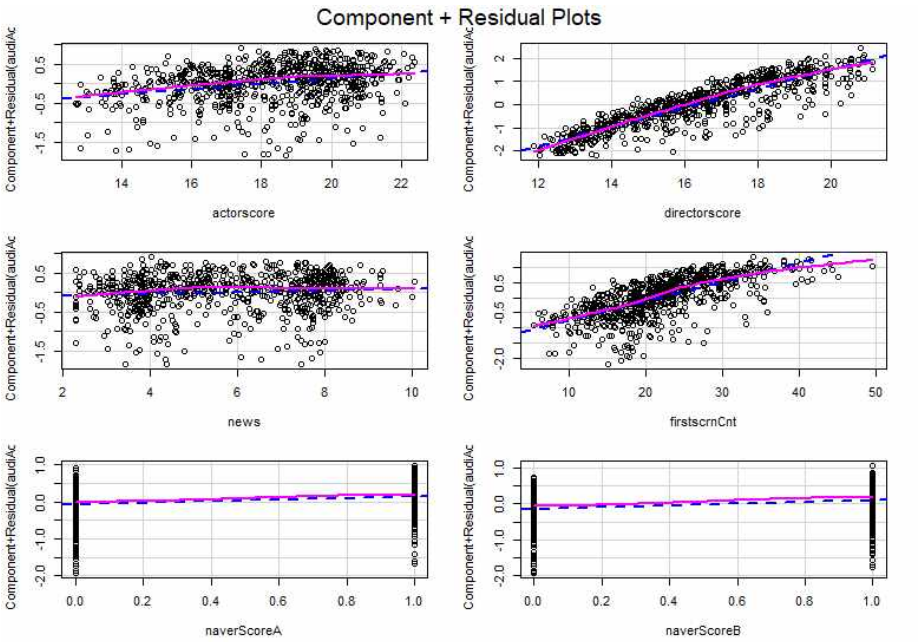
이상치를 다루는 방법에는 여러 가지가 있으나, 본 연구에서는 앞서 채택된 모형으로 이상치 검정을 실시한 후, Bonferroni p가 0.05보다 작은 값 중 가장 작은 값을 이상치로 선정하여 제거하였다. 한 번에 하나 씩 이상치를 제거하고 다시 유의성 검정을 실시하면서 회귀 모형의 변화를 살펴본 결과 [그림 2-14]와 같이 12개의 이상치가 제거됨을 알 수 있다. 이때, 이상치를 제거한 회귀 모형은 [그림 2-15]과 같다.

```
> summary(lastfit)

Call:
lm(formula = audiAcc ~ actorscore + directorscore + news + firstscrnCnt +
    naverScoreA + naverScoreB, data = out682)

Residuals:
     Min       1Q   Median       3Q      Max
-1.87703 -0.22027  0.08093  0.31859  0.95994

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.341293   0.190104  17.576  < 2e-16 ***
actorscore    0.065911   0.012052   5.469 6.23e-08 ***
directorscore 0.413382   0.013731  30.106  < 2e-16 ***
news          0.023297   0.011834   1.969  0.04937 *
firstscrnCnt  0.062894   0.004206  14.952  < 2e-16 ***
naverScoreA   0.187702   0.062205   3.017  0.00264 **
naverScoreB   0.244088   0.055046   4.434 1.07e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4858 on 727 degrees of freedom
  (10 observations deleted due to missingness)
Multiple R-squared:  0.8953,    Adjusted R-squared:  0.8944
F-statistic:  1036 on 6 and 727 DF,  p-value: < 2.2e-16
```

[그림 2-15]

2. 선형성 진단

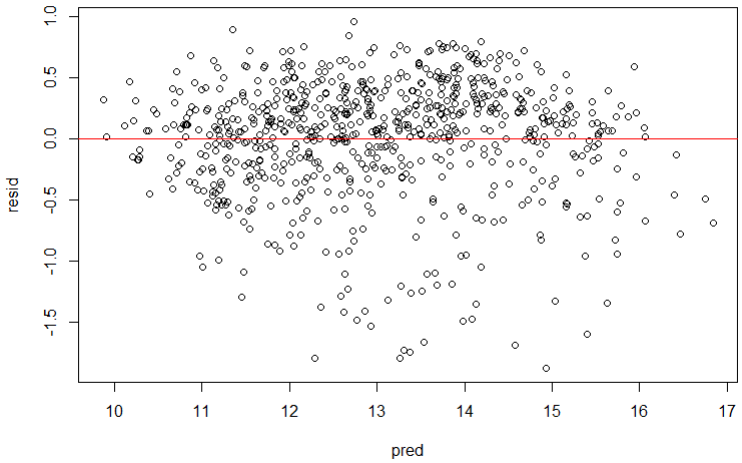
종속변수와 독립변수들 사이의 선형성을 확인하기 위하여 (성분-잔차 산점도)를 이용한다. [그림 2-16]의 분홍색 선은 실제 두 변수 간의 함수 형태를 보여 주고, 파란 점선은 적합 직선을 나타낸다. 두 선이 충분히 일치하므로 선형 관계가 있다고 결론 내릴 수 있다. 또한, 더미 변수인 naverScoreA와 naverScoreB도 선형성을 만족한다.



[그림 2-16]

3. 등분산성 검정

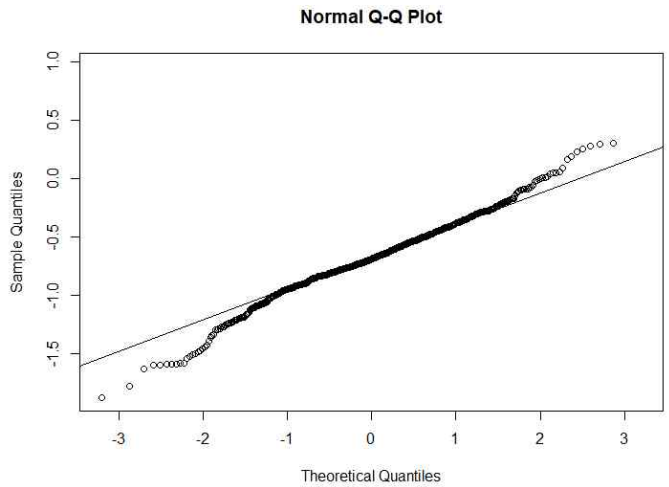
오차의 등분산성을 확인하기 위하여 (잔차와 종속변수 추정치 산점도)를 이용한다. 이때, 잔차는 추정된 회귀 모형이 종속 변수의 변동을 설명하지 못하는 부분에 해당되므로 산점도가 평균 0을 중심으로 무작위하게 흩어져 있어야 한다. 따라서, 해당 회귀 모형의 잔차[그림 2-17]은 등분산성을 만족한다.



[그림 2-17]

4. 정규성 검정

잔차의 정규성을 검정하기 위하여 Normal Q-Q plot을 이용하였다. [그림 2-18]의 잔차들이 y=x 직선과 유사하기 때문에 해당 회귀 모형의 잔차는 정규성을 만족한다.



[그림 2-18]

제4절 회귀 모형 예측

본 연구에서 구현한 회귀 모형을 통해 실제로 영화의 누적 관객수를 예측할 수 있는지 확인하기 위하여, 2019년에 개봉한 영화 중 7편을 회귀 모형에 적용하였다.

이때, 회귀 모형에 로그 변환을 적용하였기 때문에 원 스케일에 대한 모형이 아니다. 이는 로그 변환으로 핏팅된 모형은 exp 변환을 적용하여도 종속 변수와 독립 변수 간의 관계가 '더하기'가 아닌, '곱하기' 관계이므로 원 스케일로 되돌려서 비교하기에는 어려움이 존재함을 의미한다. 그렇기 때문에 예측 데이터 또한 회귀 모형을 구현할 때 변환한 방식과 동일하게 적용하였다. [표 2-3]은 회귀 모형에 실제 데이터를 적용하여 도출된 누적 관객수와 예측 관객수에 대한 정보이다.

영화 명	log(누적 관객수)	log(예측 관객수)
기생충	*15.97236*	*12.71231*
맨 인 블랙: 인터내셔널	13.64268	12.92020
사바하	14.67468	13.34832
스파이더맨: 파 프롬 홈	15.83750	14.09206
알라딘	*15.25435*	*12.06984*
어벤져스: 엔드게임	16.37675	15.18133
쥬만지: 넥스트 레벨	13.86033	12.88407

[표 2-3]

예측 결과, '맨 인 블랙: 인터내셔널', '사바하', '스파이더맨: 파 프롬 홈', '어벤져스: 엔드게임', '쥬만지: 넥스트 레벨' 의 경우 log(누적 관객수)와 log(예측 관객수)가 유사함을 볼 수 있다. 반면에 '기생충', '알라딘' 의 경우 log(누적 관객수)와 log(예측 관객수)의 차이가 크게 존재하였으므로, 앞서 정의한 흥행 요인 외에도 영화의 누적 관객수에 영향을 미치는 요인이 존재함을 알 수 있다.

1. 기생충

예측에 사용된 실제 데이터를 분석한 결과, 봉준호 감독이 2014년부터 2018년에 개봉한 영화는 설국열차(재개봉)과 옥자였다. 하지만, 옥자는 OTT사이트인 netflix를 중심으로 개봉되었다. 'KOBIS'에서는 극장에서 개봉한 영화에 대한 자료만을 취급하기 때문에 '옥자'에 대한 데이터를 구할 수 없었다. 그러므로 영화 '기생충'의 directorscore 데이터에는 감독의 명성에 비해 누락된 값이 존재하므로 log(누적 관객수)와 log(예측 관객수)의 차이가 발생하였다.

2. 알라딘

예측에 사용된 실제 데이터를 분석한 결과, 영화 '알라딘'의 주연 배우인 '메나 마수드'는 신인 배우기 때문에 actorscore가 0이다. 또한, 감독인 '가이 리치'는 데이터를 수집한 기간동안 개봉한 영화의 성적이 나빴다. 따라서 알라딘의 데이터는 actorscore와 directorscore가 상대적으로 낮았다.하지만 '알라딘'은 개봉 초반 관객 수 추이는 좋지 못했으나 입소문을 타면서 흥행에 성공한 경우이다. 본 연구에서 사용되는 데이터는 모두 과거에 축적된 성적을 바탕으로 구성되었기 때문에 입소문과 같이 개봉 후에 영향을 받는 요인은 고려하지 않았다. 그러므로 신인배우와 성적이 좋지 않은 감독의 작품인 영화 '알라딘'의 log(예측 관객수)는 log(누적 관객수) 간의 차이가 발생하였다.

제3장 결 론

예측 모델 분석 결과 배우, 감독, 개봉 전 기사수, 첫날 스크린 수, 네이버 네티즌 평점을 포함하는 모델이 회귀계수의 설명력이 높게 나타났다. 초기에 독립변수가 될 수 있는 후보군이 많았지만, 변수 선택과 유의성 검정을 통하여 설명력이 가장 높은 모델을 채택하였다.

```
> summary(lastfit)

Call:
lm(formula = audiAcc ~ actorscore + directorscore + news + firstscrnCnt +
    naverScoreA + naverScoreB, data = out682)

Residuals:
     Min       1Q   Median       3Q      Max
-1.87703 -0.22027  0.08093  0.31859  0.95994

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.341293   0.190104  17.576  < 2e-16 ***
actorscore    0.065911   0.012052   5.469 6.23e-08 ***
directorscore 0.413382   0.013731  30.106  < 2e-16 ***
news          0.023297   0.011834   1.969  0.04937 *
firstscrnCnt  0.062894   0.004206  14.952  < 2e-16 ***
naverScoreA   0.187702   0.062205   3.017  0.00264 **
naverScoreB   0.244088   0.055046   4.434 1.07e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4858 on 727 degrees of freedom
  (10 observations deleted due to missingness)
Multiple R-squared:  0.8953,    Adjusted R-squared:  0.8944
F-statistic:  1036 on 6 and 727 DF,  p-value: < 2.2e-16
```

[그림 2-19]

F-통계량의 값이 1036이고 p-값은 2.2e-16보다 작기 때문에 유의수준 5% 하에서 추정된 회귀 모형이 통계적으로 유의하다. 조정된 결정계수는 0.8944이므로 추정된 회귀 모형의 설명력은 89.44%라고 할 수 있다. 또한, 모든 변수들은 모두 유의수준을 만족한다.

따라서 회귀계수를 따라 회귀 식을 구성해보면 (2)와 같다.

$$\log(Y_i) = 3.341293 + 0.065911 * ACS_i + 0.413382 * DIS_i + 0.023297 * NEWS_i + 0.062894 * FSC_i + 0.187702 * NMS\_A_i + 0.244088 * NMS\_B_i + \epsilon_i \quad \cdots (2)$$

여기서, ACS_i = 영화 i 의 주연 배우 명성

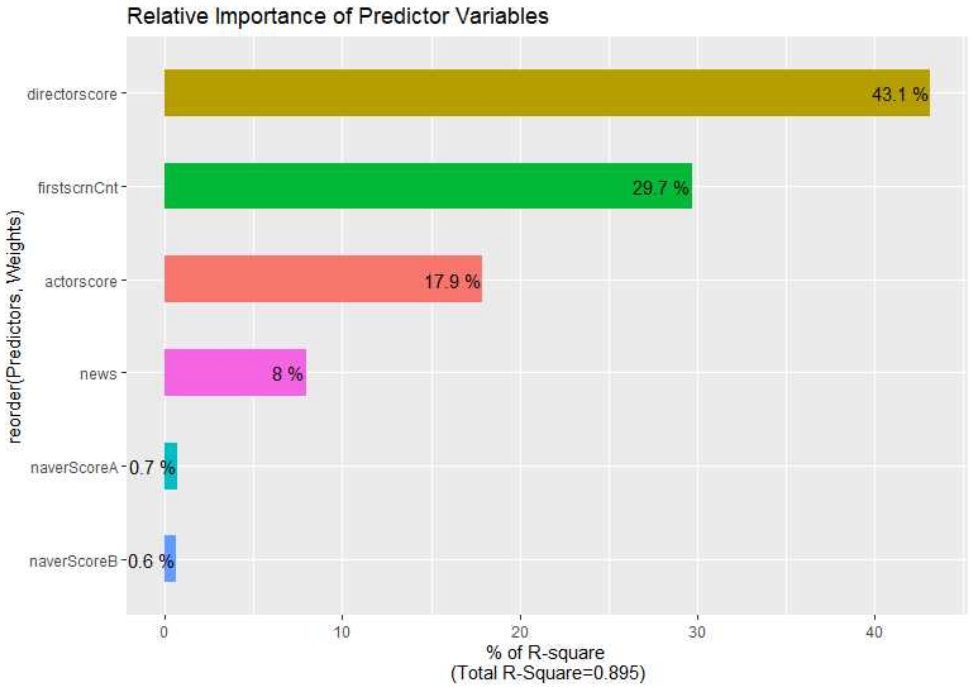
DIS_i = 영화 i 의 감독 명성

$NEWS_i$ = 영화 i 의 개봉 전 기사 수

FSC_i = 영화 i 의 첫날 스크린 수

$NMS\_등급_i$ = 영화 i 의 네이버 네티즌 평점의 더미 변수

ϵ_i = 오차항



[그림 2-20]

이때, (2)에서 각 변수들의 기여도는 [그림 2-20]과 같다. 따라서, 영화의 흥행에 영향을 미치는 변수는 감독, 첫날 스크린 수, 배우, 개봉 전 뉴스 수와 평점이지만 그중에서도 감독과 첫날 스크린 수가 조금 더 중요하다는 것을 알 수 있다.

본 연구에서는 영화의 흥행 요인을 분석하기 위해 흥행한 영화 뿐만 아니라 흥행하지 못한 영화도 고려하고자 하였으나, 데이터를 수집한 'KOBIS'에서 제공하는 일일 박스오피스 데이터는 최대 10위권까지만 제공이 되기 때문에 흥행성이 보장된 영화만을 수집할 수 밖에 없었다.

처음 독립변수의 후보군을 선정할 때, 명절 특수를 노리고 그 부근에 개봉하는 영화가 적지 않고, 여름에는 공포영화가 흥행하기 때문에 개봉 시기가 유의미할 것이라고 예상했다. 이에 본 연구에서는 개봉 시기를 개봉월과 개봉일 이후 2주 이내 공휴일 유무로 정의하였으나 이는 흥행에 영향을 미치지 않는다는 결과가 도출되었다. 이는 데이터를 수집하는 과정에

서 어느 정도 흥행성이 보장된 영화만 수집하였기 때문에, 개봉 시기가 미치는 영향력이 더 낮다고 생각된다.

앞서 시행한 회귀 모델 예측을 살펴보면, 유의한 예측 결과를 도출한 경우도 존재하지만 '기생충'과 '알라딘'과 같이 유의하지 않은 예측 결과를 도출한 경우도 존재한다. 이는 본 연구에서 OTT 플랫폼을 고려하지 않은 채 수집된 데이터를 사용했을 뿐만 아니라, 입소문과 같이 영화 개봉 후에도 흥행에 영향을 미치는 요인을 고려하지 않았기 때문이다. 영화산업이 발전함에 따라 영화관 밖에서도 영화를 관람할 수 있는 방법이 다양해지고, '옥자'처럼 OTT 플랫폼에서 독점 개봉을 하는 경우도 존재한다. 그뿐만 아니라, 소셜 미디어의 확장은 입소문을 통한 역주행과 같이 영화 개봉 후에도 영화 흥행에 영향을 끼치고 있다. 그렇기 때문에 향후 연구에서는 기존의 정의한 흥행 요인 이외에도 추가적인 흥행 요인을 고려한 분석이 필요하다.

참 고 문 헌

[1] 영화진흥위원회 (2020). 2019년 한국영화산업 결산. 영화진흥위원회.

[2] 최은영 (2018), 한국영화산업의 발전방향 분석, 한국콘텐츠학회논문지 8(11), 134-143

[3] 장은지, 현민지 (2018), SNS를 활용한 영화 흥행 분석 및 예측, 한국정보과학회 학술발표논문집, 2324-2326

[4] 전범수, 이준영 (2019), 한국 개봉 흥행 영화 평점이 성과에 미치는 영향 : 일반 이용자와 전문가 평점의 비교, 언론과학연구, 19(4), 227-253

[5] 김세윤 (2018), 데이터 분석을 활용한 한국 영화 흥행 예측, 숭실대학교 소프트웨어특성화대학원 석사학위 논문

[6] 김영현, 홍정한 (2011), 영화 흥행 결정 요인과 흥행 성과 예측 연구, 한국통계학회 논문집, 18(6), 859-869.

ABSTRACT

<표 목 차>

[그림목차]

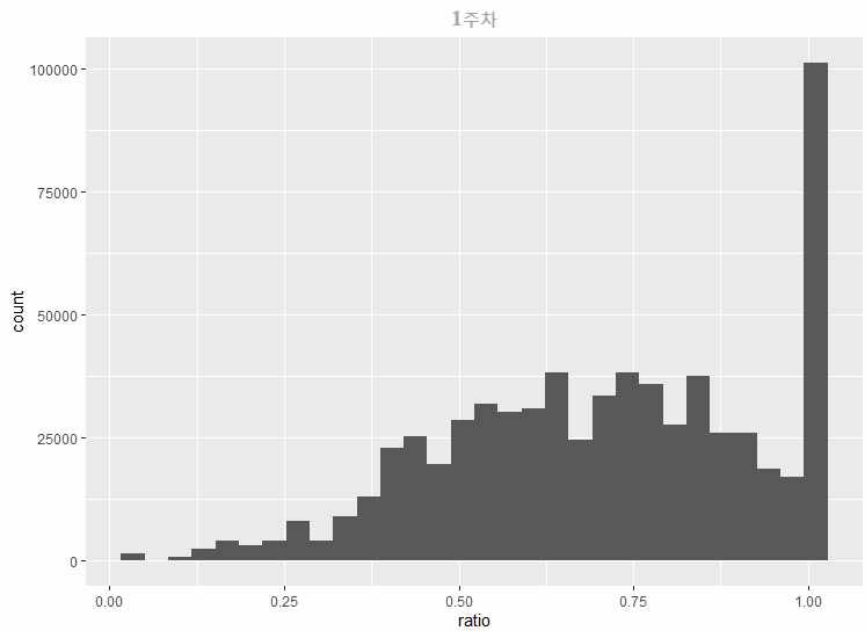
2018년 06월 27일(수)

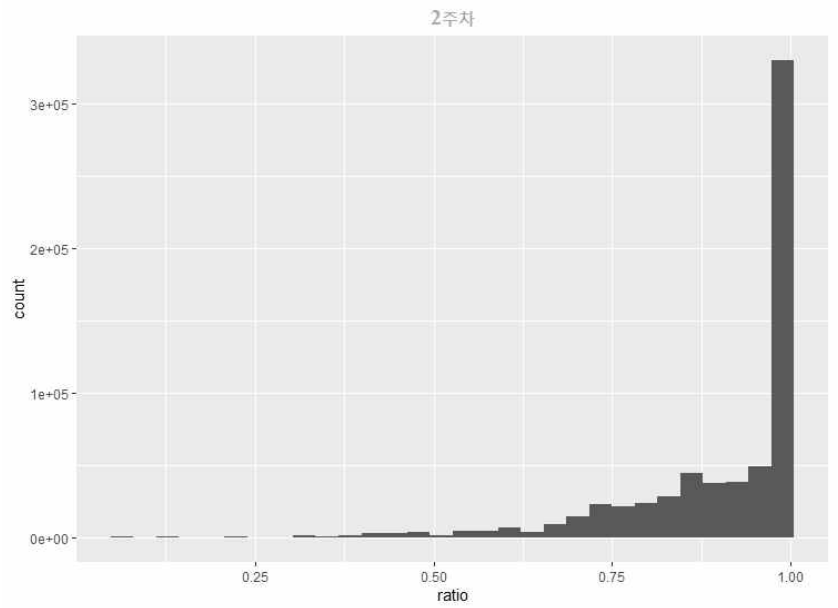
순위	영화명	개봉일	좌석판매율	좌석점유율	좌석수	매출액	누적매출액	관객수	누적관객수
1	마녀	2018-06-27	17.9%	25.5%	680,364	738,666,200	807,070,100	121,990	129,826
2	탑정: 리턴즈	2018-06-13	20.3%	23.0%	614,863	746,905,700	19,958,037,872	124,510	2,342,325
3	허스토리	2018-06-27	11.4%	14.6%	389,408	257,451,300	458,356,300	44,340	76,746
4	쥬라기 월드: 폴른 킹덤	2018-06-06	17.3%	8.6%	229,772	236,549,500	47,517,930,837	39,713	5,395,800
5	오션스8	2018-06-13	21.1%	7.2%	191,654	243,659,600	10,188,621,263	40,374	1,163,789
6	시카리오: 데이 오브 솔다도	2018-06-27	21.1%	6.6%	176,287	222,528,700	272,981,200	37,187	42,828
7	독전	2018-05-22	15.6%	2.5%	66,478	66,585,700	43,093,127,313	10,367	5,018,682
8	리틀 맨하탄	2018-06-27	4.8%	1.4%	36,251	10,137,500	16,028,500	1,727	2,348
9	여중생A	2018-06-20	13.4%	1.3%	33,567	32,586,700	638,554,200	4,509	76,393
10	빅식크: 매직체인지	2018-06-27	9.2%	1.1%	28,532	15,307,400	23,117,400	2,619	3,565

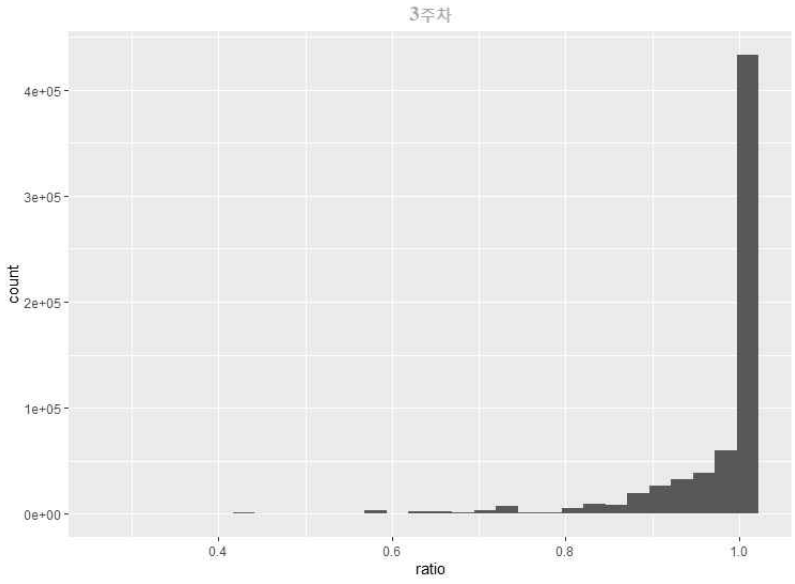
더보기

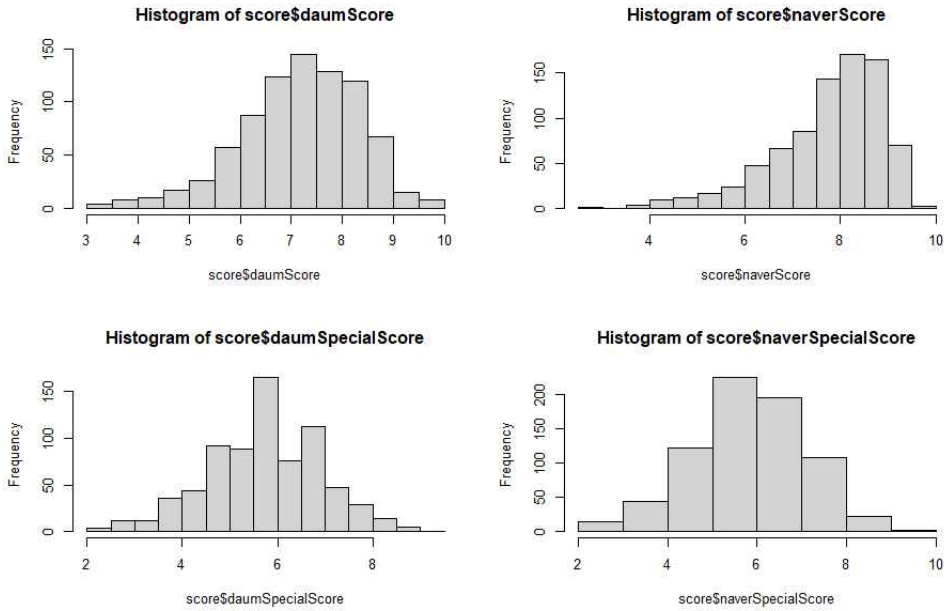
총 236건

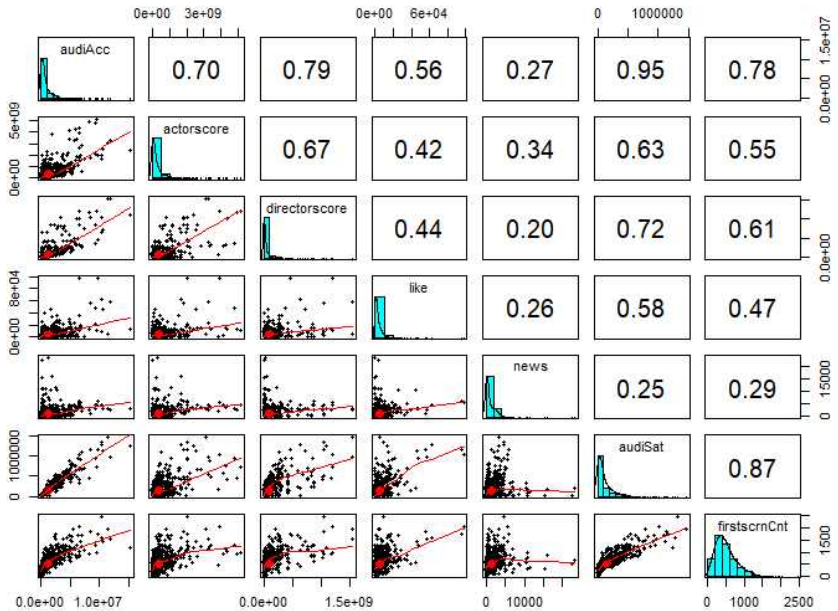
엑셀

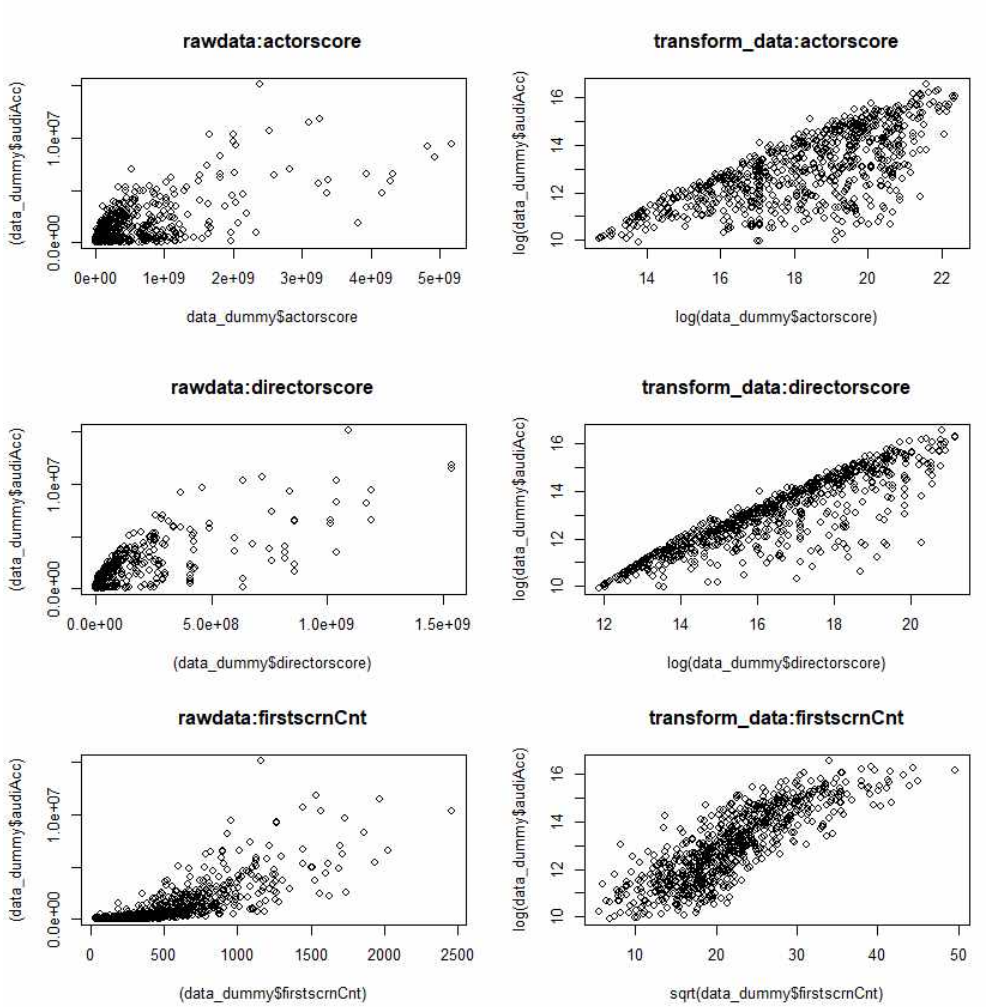


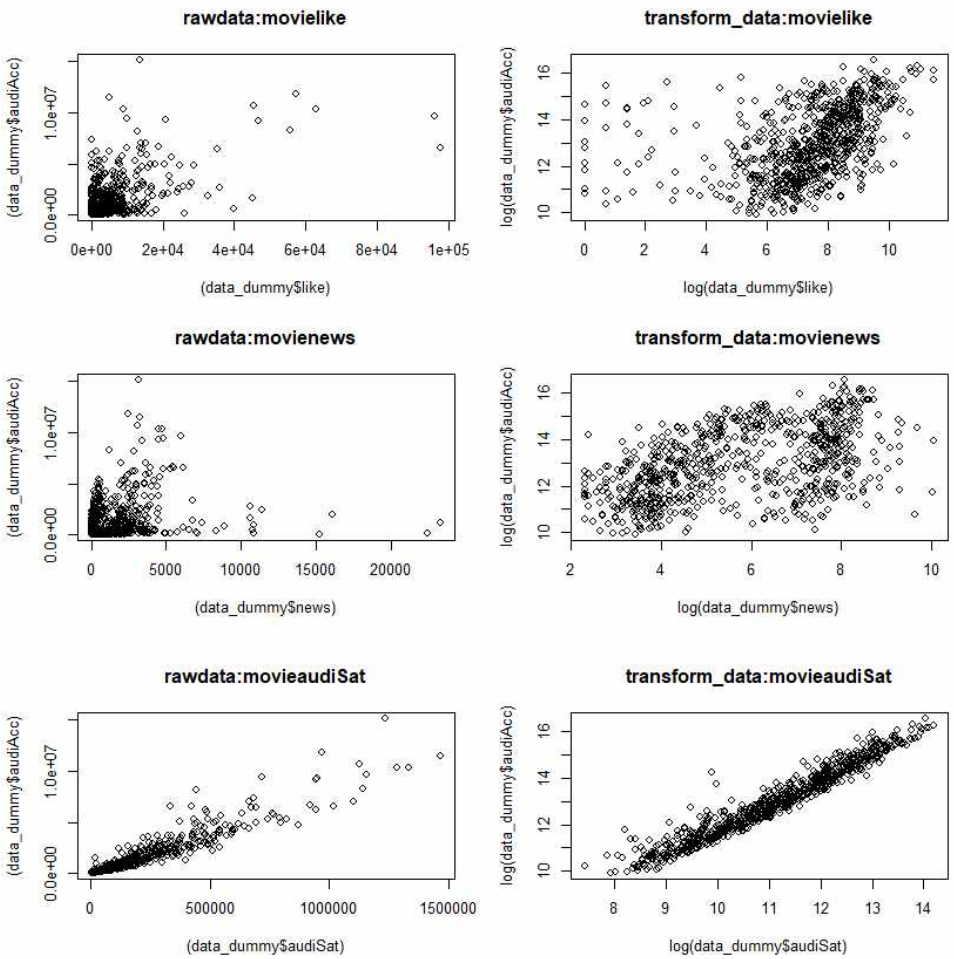












```
> summary(back)

Call:
lm(formula = audiAcc ~ actorscore + directorscore + news + audiSat +
    firstscrnCnt + gen스릴러 + gen액션 + daumScoreA + daumScoreB +
    naverScoreA + naverScoreB + openmonth11월 + openmonth1월 +
    openmonth3월, data = tdata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.86935 -0.17314  0.00177  0.16151  1.68917

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.482380   0.128938   3.741 0.000198 ***
actorscore     0.032027   0.006899   4.643 4.08e-06 ***
directorscore  0.109025   0.009153  11.912  < 2e-16 ***
news           0.037457   0.006928   5.407 8.71e-08 ***
audiSat        0.905413   0.019440  46.575  < 2e-16 ***
firstscrnCnt  -0.013583   0.003099  -4.383 1.34e-05 ***
gen스릴러     -0.090618   0.050668  -1.788 0.074117 .
gen액션       -0.062225   0.027116  -2.295 0.022028 *
daumScoreA     0.160066   0.046623   3.433 0.000630 ***
daumScoreB     0.080581   0.037876   2.127 0.033714 *
naverScoreA    0.219664   0.046537   4.720 2.82e-06 ***
naverScoreB    0.091972   0.037878   2.428 0.015418 *
openmonth11월 -0.064290   0.039356  -1.634 0.102788
openmonth1월  -0.168155   0.038774  -4.337 1.65e-05 ***
openmonth3월  -0.144868   0.037725  -3.840 0.000134 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2795 on 731 degrees of freedom
  (10 observations deleted due to missingness)
Multiple R-squared:  0.9658,    Adjusted R-squared:  0.9652
F-statistic:  1475 on 14 and 731 DF,  p-value: < 2.2e-16
```

```
> summary(forw)

Call:
lm(formula = audiAcc ~ actorscore + directorscore + like + news +
    audiSat + firstscrnCnt + gen그외 + gen드라마 + gen멜로 +
    gen스릴러 + gen액션 + gen코미디 + nation그외 + nation미국 +
    nation한국 + Grade12세이상관람가 + Grade15세이상관람가 +
    Grade전체관람가 + Grade청소년관람불가 + reddayN + reddayY +
    daumScoreA + daumScoreB + daumScoreC + daumSpecialScoreA +
    daumSpecialScoreB + daumSpecialScoreC + naverScoreA + naverScoreB +
    naverScoreC + naverSpecialScoreA + naverSpecialScoreB + naverSpecialSc
oreC +
    openmonth10월 + openmonth11월 + openmonth12월 + openmonth1월 +
    openmonth2월 + openmonth3월 + openmonth4월 + openmonth5월 +
    openmonth6월 + openmonth7월 + openmonth8월 + openmonth9월,
    data = tdata)

Residuals:
    Min      1Q  Median      3Q     Max
-0.8937 -0.1690 -0.0015  0.1558  1.6636

Coefficients: (9 not defined because of singularities)
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.3780453  0.1616332   2.339 0.019617 *
actorscore           0.0341322  0.0072354   4.717 2.88e-06 ***
directorscore        0.1091435  0.0094756  11.518  < 2e-16 ***
like                 0.0059414  0.0064811   0.917 0.359599
news                 0.0324274  0.0092782   3.495 0.000504 ***
audiSat              0.9051246  0.0210827  42.932  < 2e-16 ***
firstscrnCnt        -0.0131914  0.0031919  -4.133 4.01e-05 ***
gen그외              0.0115094  0.0438489   0.262 0.793028
gen드라마            0.0676971  0.0450038   1.504 0.132961
gen멜로              0.0822993  0.0612993   1.343 0.179837
gen스릴러           -0.0489301  0.0631286  -0.775 0.438547
gen액션             -0.0280445  0.0474092  -0.592 0.554347
gen코미디                   NA         NA      NA       NA
nation그외          -0.0050424  0.0431553  -0.117 0.907017
nation미국          -0.0211670  0.0336155  -0.630 0.529107
nation한국                  NA         NA      NA       NA
Grade12세이상관람가  0.0042502  0.0368048   0.115 0.908097
Grade15세이상관람가  0.0060467  0.0333643   0.181 0.856238
Grade전체관람가      0.0440150  0.0451569   0.975 0.330036
Grade청소년관람불가         NA         NA      NA       NA
reddayN              0.0488557  0.0342004   1.429 0.153585
reddayY                     NA         NA      NA       NA
daumScoreA           0.1523767  0.0489508   3.113 0.001927 **
daumScoreB           0.0820205  0.0389491   2.106 0.035570 *
daumScoreC                  NA         NA      NA       NA
daumSpecialScoreA   -0.0116576  0.0555124  -0.210 0.833728
daumSpecialScoreB   -0.0411731  0.0339594  -1.212 0.225756
daumSpecialScoreC           NA         NA      NA       NA
naverScoreA          0.2174613  0.0498000   4.367 1.45e-05 ***
naverScoreB          0.0945434  0.0395719   2.389 0.017148 *
naverScoreC                 NA         NA      NA       NA
naverSpecialScoreA   0.0006031  0.0539099   0.011 0.991078
naverSpecialScoreB   0.0192892  0.0339960   0.567 0.570623
naverSpecialScoreC          NA         NA      NA       NA
openmonth10월       -0.0452367  0.0505973  -0.894 0.371596
openmonth11월       -0.0813733  0.0525084  -1.550 0.121655
openmonth12월       -0.0444199  0.0502497  -0.884 0.377005
openmonth1월        -0.1873566  0.0517916  -3.618 0.000319 ***
openmonth2월        -0.0355085  0.0526493  -0.674 0.500255
openmonth3월        -0.1565945  0.0514558  -3.043 0.002427 **
openmonth4월        -0.0241436  0.0508017  -0.475 0.634753
openmonth5월        -0.0602371  0.0501657  -1.201 0.230245
openmonth6월        -0.0042009  0.0525222  -0.080 0.936273
openmonth7월         0.0185848  0.0498622   0.373 0.709467
openmonth8월        -0.0014145  0.0502937  -0.028 0.977570
openmonth9월                NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.281 on 709 degrees of freedom
  (10 observations deleted due to missingness)
Multiple R-squared:  0.9665,    Adjusted R-squared:  0.9648
F-statistic: 567.8 on 36 and 709 DF,  p-value: < 2.2e-16
```

```
> summary(both)

Call:
lm(formula = audiAcc ~ actorscore + directorscore + news + audiSat +
    firstscrnCnt + gen스릴러 + gen액션 + daumScoreA + daumScoreB +
    naverScoreA + naverScoreB + openmonth11월 + openmonth1월 +
    openmonth3월, data = tdata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.86935 -0.17314  0.00177  0.16151  1.68917

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.482380   0.128938   3.741 0.000198 ***
actorscore     0.032027   0.006899   4.643 4.08e-06 ***
directorscore  0.109025   0.009153  11.912  < 2e-16 ***
news           0.037457   0.006928   5.407 8.71e-08 ***
audiSat        0.905413   0.019440  46.575  < 2e-16 ***
firstscrnCnt  -0.013583   0.003099  -4.383 1.34e-05 ***
gen스릴러     -0.090618   0.050668  -1.788 0.074117 .
gen액션       -0.062225   0.027116  -2.295 0.022028 *
daumScoreA     0.160066   0.046623   3.433 0.000630 ***
daumScoreB     0.080581   0.037876   2.127 0.033714 *
naverScoreA    0.219664   0.046537   4.720 2.82e-06 ***
naverScoreB    0.091972   0.037878   2.428 0.015418 *
openmonth11월 -0.064290   0.039356  -1.634 0.102788
openmonth1월  -0.168155   0.038774  -4.337 1.65e-05 ***
openmonth3월  -0.144868   0.037725  -3.840 0.000134 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2795 on 731 degrees of freedom
  (10 observations deleted due to missingness)
Multiple R-squared:  0.9658,    Adjusted R-squared:  0.9652
F-statistic:  1475 on 14 and 731 DF,  p-value: < 2.2e-16
```

```
> ###################이상치
> outlierTest(outsatfit)
     rstudent unadjusted p-value Bonferroni p
395 -4.658773         3.7794e-06    0.0028194
288 -4.228778         2.6478e-05    0.0197530
25  -4.192956         3.0911e-05    0.0230600
> outlierTest(out395back)
     rstudent unadjusted p-value Bonferroni p
25  -4.365607          1.449e-05     0.010795
288 -4.319889          1.775e-05     0.013224
> outlierTest(out25back)
     rstudent unadjusted p-value Bonferroni p
287 -4.400981         1.2371e-05    0.0092038
428 -4.090295         4.7837e-05    0.0355900
> outlierTest(out287back)
     rstudent unadjusted p-value Bonferroni p
427 -4.215951         2.7973e-05     0.020784
472 -4.039881         5.9105e-05     0.043915
> outlierTest(out427back)
     rstudent unadjusted p-value Bonferroni p
471 -4.104003         4.5160e-05     0.033508
3   -4.072259         5.1622e-05     0.038304
> outlierTest(out471back)
     rstudent unadjusted p-value Bonferroni p
3   -4.150662         3.7045e-05     0.027451
191 -4.013867         6.5876e-05     0.048814
> outlierTest(out3back)
     rstudent unadjusted p-value Bonferroni p
190 -4.111221         4.3814e-05     0.032422
> outlierTest(out190back)
     rstudent unadjusted p-value Bonferroni p
369 -4.061791         5.3962e-05     0.039878
349 -4.061558         5.4014e-05     0.039917
> outlierTest(out369back)
     rstudent unadjusted p-value Bonferroni p
349 -4.151764         3.6888e-05     0.027223
16  -4.043787         5.8189e-05     0.042943
> outlierTest(out349back)
    rstudent unadjusted p-value Bonferroni p
16 -4.122626         4.1765e-05     0.030781
> outlierTest(out16back)
     rstudent unadjusted p-value Bonferroni p
161 -4.021586         6.3841e-05     0.046987
> outlierTest(out161back)
     rstudent unadjusted p-value Bonferroni p
682 -4.014271         6.5818e-05     0.048376
> outlierTest(out682back)
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
     rstudent unadjusted p-value Bonferroni p
337 -3.919456         9.7144e-05     0.071304
```

<Code>

1. 네이버 평점 크롤링(python)

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
df=pd.read_csv('C:\\Users\\yoonj\\Desktop\\regression\\codes_result1.txt',sep="\t")
df2=pd.read_csv('C:\\Users\\yoonj\\Desktop\\regression\\codes_result1.txt',sep="\t")
df2.rename(columns = {'codes' : 'movieScore'}, inplace = True)
df2
df=pd.concat([df,df2],axis=1)
df['movieScore']='NA'
df
df
#네이버 영화 관람객 평점 크롤링
def get_tag(i):
    req = requests.get("https://movie.naver.com/movie/bi/mi/basic.nhn?code="+ str(i))
    html = req.text
    soup = BeautifulSoup(html, 'html.parser')
    print(str(soup.select(".score_area .netizen_score .sc_view .star_score")[0]))
    return str(soup.select(".score_area .netizen_score .sc_view .star_score")[0])
def split_tag(tag: str) -> str:
    print(tag.split('>')[6].split('<')[0])
    return tag.split('>')[6].split('<')[0]
idx = 0
for i in df['codes'].values:
    df.movieScore.values[idx] = split_tag(get_tag(str(i)))
    idx +=1
```

```
df.to_csv('C:\\Users\\yoonj\\Desktop\\regression\\data\\codes_result.csv')
```

2. 다음 평점 크롤링(python)

```
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_driver_path = 'C:/chrome_driver/chromedriver.exe' # chrome driver의 경로
driver = webdriver.Chrome(chrome_options=chrome_options, executable_path=chrome_driver_path)
url_r = 'https://movie.daum.net/moviedb/main?movieId='
daum_code = '108595'
url = url_r + daum_code
driver.get(url)
soup = BeautifulSoup(driver.page_source, 'html.parser')
mvscore = soup.find_all('span', {'class': 'score_rating'})
daumScore = mvscore[0].string
daumspecialScore = mvscore[1].string
print(mvscore)
def get_daumScore(i):
    chrome_options = Options()
    chrome_options.add_argument('--headless')
    chrome_driver_path = 'C:/chrome_driver/chromedriver.exe' # chrome driver의 경로
    driver = webdriver.Chrome(chrome_options=chrome_options, executable_path=chrome_driver_path)
    url_r = 'https://movie.daum.net/moviedb/main?movieId='
    daum_code = str(i)
    url = url_r + daum_code
    driver.get(url)
```

```
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    mvscore = soup.find_all('span', {'class': 'score_rating'})
    #if(mvscore == '') : return print(str(i))
    daumScore = mvscore[0].string
    daumspecialScore = mvscore[1].string
    return daumScore, daumspecialScore
import pandas as pd
df=pd.read_csv('C:\\Users\\Jiwon\\Desktop\\Movie\\regression\\data\\daum
ms.csv', encoding='CP949')
df1 = df
df = pd.concat([df,df1], axis = 1)
df
df.columns =["movieNm", "codes", "daumScore", "daumSpecialScore"]
df['daumScore'] = 'NA'
df['daumSpecialScore'] = 'NA'
res = df
res
idx = 0
for i in res['codes'].values:
    print(str(i)  + "idx : " + str(idx))
    res.daumScore.values[idx], res.daumSpecialScore.values[idx] =
get_daumScore(str(i))
    print("daumScore : " + res.daumScore.values[idx] + "special: " +
res.daumSpecialScore.values[idx])
    idx +=1
x, y = get_daumScore('116145)
df.daumScore.values[54], df.daumSpecialScore.values[54] =
get_daumScore('100359')
res['daumScore'][54] = 6.6
res['daumSpecialScore'][54]= 6.1
res.iloc[54]
import time
```

```
df_del = res
df_del=df_del.iloc[792:]
df_del
idx = 55
for i in df_del['codes'].values:
    print(str(i)  + "idx : " + str(idx))
    res.daumScore.values[idx],      res.daumSpecialScore.values[idx]      =
get_daumScore(str(i))
    print("daumScore : " + res.daumScore.values[idx] + "special: " +
res.daumSpecialScore.values[idx])
    idx +=1
res['daumScore'][72] = 8.9
res['daumSpecialScore'][72]= 0
res.iloc[72]
idx = 73
for i in df_del['codes'].values:
    print(str(i)  + "idx : " + str(idx))
    res.daumScore.values[idx],      res.daumSpecialScore.values[idx]      =
get_daumScore(str(i))
    print("daumScore : " + res.daumScore.values[idx] + "special: " +
res.daumSpecialScore.values[idx])
    idx +=1
res['daumScore'][245] = 7.6
res['daumSpecialScore'][245]= 0
res.iloc[245]
idx = 246
for i in df_del['codes'].values:
    print(str(i)  + "idx : " + str(idx))
    res.daumScore.values[idx],      res.daumSpecialScore.values[idx]      =
get_daumScore(str(i))
    print("daumScore : " + res.daumScore.values[idx] + "special: " +
res.daumSpecialScore.values[idx])
```

```
    idx +=1
def get_daumScore2(i):
    chrome_options = Options()
    chrome_options.add_argument('--headless')
    chrome_driver_path = 'C:/chrome_driver/chromedriver.exe' # chrome driver의 경로
    driver = webdriver.Chrome(chrome_options=chrome_options, executable_path=chrome_driver_path)
    url_r = 'https://movie.daum.net/moviedb/main?movieId='
    daum_code = str(i)
    url = url_r + daum_code
    driver.get(url)
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    mvscore = soup.find_all('span', {'class': 'score_rating'})
    if(len(mvscore) == 1) : return mvscore[0].string, str(0)
    daumScore = mvscore[0].string
    daumspecialScore = mvscore[1].string
    return daumScore, daumspecialScore
idx = 792
for i in df_del['codes'].values:
    print(str(i)  + "idx : " + str(idx))
    res.daumScore.values[idx], res.daumSpecialScore.values[idx] = get_daumScore2(str(i))
    print("daumScore : " + res.daumScore.values[idx] + "special: " + res.daumSpecialScore.values[idx])
    idx +=1
mvscore
len(mvscore)
res.to_csv('C:\\Users\\Jiwon\\Desktop\\Movie\\regression\\data\\daum_score.csv',encoding = 'CP949')
```

3. 네이버 영화 기사 수 크롤링(python)

```
# 네이버 영화 기사 수 크롤링
import pandas as pd
import json
import os
import re
import numpy as np
from bs4 import BeautifulSoup
import requests
from pandas import Series, DataFrame
import datetime
import time
MovieData=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\movie_final_data1.csv', encoding='CP949')
news_date=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\news_date.csv', encoding='CP949')
special_score=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\special_score.csv', encoding='CP949')
red_date = pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\red_date.csv', encoding='CP949')
red_date.columns=['startRd','endRd']
red_date
for i in range(0,88):
    red_date['startRd'][i] = datetime.datetime.strptime(red_date['startRd'][i],"%Y-%m-%d").date()
    red_date['endRd'][i] = datetime.datetime.strptime(red_date['endRd'][i],"%Y-%m-%d").date()
red_date
for i in range(0,817):
    MovieData['openDt'][i] = datetime.datetime.strptime(MovieData['openDt'][i],"%Y-%m-%d").date()
specialscore=special_score['moviespecialScore']
```

```
specialscore
Movies=[i for i in MovieData['movieNm']]
Search=zip(Movies)
Plus=[]
for i in Search:
    a=list(i)
    Plus.append(a)

MovieDirector=[]
for i in Plus:
    b=' '.join(i)
    MovieDirector.append(b)
MovieDirector = DataFrame(MovieDirector)
#검색리스트와 기간을 url에 넣을 수 있는 함수
def URLmaker(query,sdate,edate):
    try:
        base_url =
'https://search.naver.com/search.naver?where=news&query={}&sm=tab_opt
&sort=0&photo=0&field=0&reporter_article=&pd=3&ds={}&de={}'
        encoded_query=requests.utils.quote(query, encoding='MS949')
        base_url = base_url.format(encoded_query, sdate, edate)
    except:
        print(i)
        pass
    return base_url
#앞에 만든 함수에 검색과 기간을 넣어서 최종 url 생성
FinalUrl=[]
for i in range(len(MovieDirector)):
    url=URLmaker(MovieDirector[0][i], news_date['startDt'][i],
news_date['openDt'][i])
    FinalUrl.append(url)
FinalUrl[3]
```

```
news_count=[]
num=0
def get_tag(i):
        rep = requests.get(i)
        source = rep.text
        soup = BeautifulSoup(source, 'html.parser')
        #print(soup.select("#main_pack > div > div.section_head > div.title_desc.all_my > span"))
        tag = str(soup.select("#main_pack > div > div.section_head > div.title_desc.all_my > span"))
        tag_split = tag[14:]
        tag_split = tag_split[0:-9]
        print(tag_split)
        return tag_split

for i in FinalUrl:
    news_count.append(get_tag(i))

news_count=pd.DataFrame(news_count)
news_count
news_count.columns=['news']
news_count
type(news_count['news'][0])
for i in range(0,816):
    news_count['news'][i] = news_count['news'][i].replace(',', '')
news_count
redday = pd.DataFrame(np.zeros((816,1)))
redday.columns=['redday']
redday
for i in range(0,816):
    for j in range(0,88):
        if(abs(MovieData['openDt'][i]-red_date['startRd'][j]).days      <=14
```

```
and abs(MovieData['openDt'][i]-red_date['endRd'][j]).days <=14):
                redday['redday'][j]='Y'
redday
for i in range(0,816):
    redday['redday'][i] = str(redday['redday'][i])
    redday['redday'][i] = redday['redday'][i].replace('0.0','N')
redday
MovieData=pd.concat([MovieData,news_count,specialscore,redday],axis=1)
MovieData
MovieData['movieNm'][637]
MovieData.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\MovieData.
csv',index=False,encoding='utf-8-sig')
------------------------------
MovieData                                                    =
pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\MovieData.csv',
encoding='utf-8-sig')
sataudi                                                      =
pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\Sat_audicnt.csv',
encoding='CP949')
sataudi
#정글북 -> 엑셀로 삭제
#sataudi=sataudi.drop(sataudi.index[636])
#sataudi.reset_index()
audiSat=sataudi['audiSat']
audiSat
MovieData=pd.concat([MovieData,audiSat],axis=1)
MovieData
MovieData.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\final_data.cs
v',index=False,encoding='CP949')
```

4. 공휴일 데이터 및 개봉 후 첫 번째 토요일 관객 수 (python)

```
import calendar
from datetime import datetime
from dateutil.relativedelta import relativedelta
import pandas as pd
import numpy as np
def getDayName(year, month, day):
    dayString = ["월요일","화요일","수요일","목요일","금요일","토요일","일요일"]
    aday = datetime.date(year, month, day)
    bday = aday.weekday()
    return dayString[bday]
print(getDayName(2020,6,23))
type(getDayName(2020,6,23))
data=pd.read_csv('C:\\Users\\tldus\\Desktop\\regression\\data\\add_mvScore.csv', encoding = 'CP949')
data
date = data['date']
type(date)
date_split = pd.DataFrame(columns=['year','month','day'])
for i in range(0,15756):
    year = int(date[i].split('-')[0])
    month = int(date[i].split('-')[1])
    day = int(date[i].split('-')[2])

    date_split=date_split.append(pd.DataFrame([[year,month,day]], columns=['year','month','day']), ignore_index=True)
type(date_split['year'][0])
days = []
for i in range(0,15756):
    day_of_weeks = getDayName(date_split['year'][i],date_split['month'][i],date_split['day'][i])
    days.append(day_of_weeks)
```

```
days = pd.DataFrame(days)
days.columns=['days_of_week']
days
data=pd.concat([data,days], axis=1)
data
data.to_csv('C:\\Users\\tldus\\Desktop\\regression\\data\\datamart_days.csv
',index=False,encoding='utf-8-sig')
----------------------------------
redday=pd.read_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\reddays_j
oin2014.csv', encoding = 'CP949')
redday
redday[['day']] = redday[['day']].applymap(str).applymap(lambda s:
"{}-{}-{}".format(s[0:4],s[4:6],s[6:]))
redday
redday.to_csv('C:\\Users\\tldus\\Desktop\\regression\\data\\redday_format.c
sv',index=False,encoding='utf-8-sig')
--------------------------------
reddays=redday.loc[:,['day']]
reddays
redday_split = pd.DataFrame(columns=['year','month','day'])
for i in range(0,88):
    year = int(redday['day'][i].split('-')[0])
    month = int(redday['day'][i].split('-')[1])
    day = int(redday['day'][i].split('-')[2])

    redday_split=redday_split.append(pd.DataFrame([[year,month,day]],
columns=['year','month','day']), ignore_index=True)
redday_split
startredday = []
for i in range(0,88):
    my_date=datetime(redday_split ['year'][i],redday_split
['month'][i],redday_split ['day'][i])
```

```
    weeks2 = my_date + relativedelta(days=-14)
    startredday.append(weeks2)

startredday = pd.DataFrame(startredday)
startredday.columns=['startRd']
startredday
red_date = pd.concat([startredday,reddays], axis=1)
red_date
red_date.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\red_date.csv',i
ndex=False,encoding='CP949')
reddays=np.array(reddays['day'].tolist())
reddays
holidays = pd.DataFrame(np.zeros((15756,1)))
holidays.columns=['holiday']
holidays
for i in reddays:
    for j in range(0, 15756):
        if(i == data['date'][j]):
            holidays['holiday'][j]='Y'
        else:
            holidays['holiday'][j]='N'
holidays['holiday'].value_counts()
data=pd.concat([data,holidays], axis=1)
data
-------------------------------------------------
data=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\datamart_re
dday.csv', encoding='utf-8-sig')
data
sat = data['movieNm'].unique()
sat = pd.DataFrame(sat)
sat.columns=['movieNm']
sat
```

```
#제자, 옥한음
sat=sat.drop(sat.index[637])
sat.reset_index()
sat
is_first = data['weeks'] == 1
is_sat = data['days_of_week'] == '토요일'
true_sat = data[is_first & is_sat]
true_sat.reset_index(inplace=True)
true_sat
#미녀와 야수
true_sat=true_sat.drop(sat.index[320])
true_sat.reset_index()
#너의 췌장을
true_sat=true_sat.drop(sat.index[133])
true_sat.reset_index()
true_sat1= true_sat['movieNm']
true_sat2= true_sat['audiCnt']
true_sat=pd.concat([true_sat1,true_sat2],axis=1)
true_sat
sat1=pd.merge(sat,true_sat,on='movieNm',how='left')
sat1
sat1.columns=['movieNm','audiSat']
sat1
print(sat1['audiSat'].value_counts(dropna=False))
sat1 = sat1.fillna(0)
sat1 = sat1.astype({'audiSat': 'int'})
print(sat1.dtypes)
sat1
sat1.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\Sat_audicnt.csv',in
dex=False,encoding='CP949')

################################################################################
```

```
from datetime import datetime
from dateutil.relativedelta import relativedelta
import pandas as pd
date=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\movie_final
_data1.csv', encoding='CP949')
date
openDt = date['openDt']
type(openDt)
opendt_split = pd.DataFrame(columns=['year','month','day'])
for i in range(0,816):
    year = int(openDt[i].split('-')[0])
    month = int(openDt[i].split('-')[1])
    day = int(openDt[i].split('-')[2])

    opendt_split=opendt_split.append(pd.DataFrame([[year,month,day]],
columns=['year','month','day']), ignore_index=True)
opendt_split
--------------
open_month=opendt_split['month']
type(open_month)
for i in range(0,816):
    open_month[i]=str(open_month[i])+'월'
open_month
open_month.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\open_mont
h.csv',index=False,encoding='CP949')
-----------
startdt = []
for i in range(0,816):

my_date=datetime(opendt_split['year'][i],opendt_split['month'][i],opendt_split
['day'][i])
    weeks2 = my_date + relativedelta(days=-14)
```

```
    startdt.append(weeks2)

startDt = pd.DataFrame(startdt)
startDt.columns=['startDt']
startDt
news_date = pd.concat([startDt,openDt], axis=1)
news_date
type(news_date['startDt'][0])
for i in range(0,816):
    news_date['startDt'][i]                                                    =
news_date['startDt'][i].strftime('%Y-%m-%d')
type(news_date['startDt'][0])
for i in range(0,816):
    news_date['startDt'][i] = news_date['startDt'][i].replace("-",".")
    news_date['openDt'][i] = news_date['openDt'][i].replace("-",".")
news_date
news_date.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\news_date.c
sv',index=False,encoding='CP949')
```

5. 네이버, 다음 평점 범주화 (python)

```
import pandas as pd
import numpy as np
score=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\final_score
.csv', encoding='CP949')
score_dummy=score
score
#다음 네티즌 평점 범주화
for i in range(0,816):
    if(score['daumScore'][i]>=8 and score['daumScore'][i]<=10):
        score_dummy['daumScore'][i] = 'A'
    elif(score['daumScore'][i]>=6 and score['daumScore'][i]<8):
        score_dummy['daumScore'][i] = 'B'
```

```
    else:
        score_dummy['daumScore'][i] = 'C'
score_dummy
#네이버 네티즌 평점 범주화
for i in range(0,816):
    if(score['movieScore'][i]>=8.5 and score['movieScore'][i]<=10):
        score_dummy['movieScore'][i] = 'A'
    elif(score['movieScore'][i]>=6.5 and score['movieScore'][i]<8.5):
        score_dummy['movieScore'][i] = 'B'
    else:
        score_dummy['movieScore'][i] = 'C'
score_dummy
#다음 전문가 평점 범주화
for i in range(0,816):
    if(score['daumSpecialScore'][i]>=7                                    and
score['daumSpecialScore'][i]<=10):
        score_dummy['daumSpecialScore'][i] = 'A'
    elif(score['daumSpecialScore'][i]>=5.5                                and
score['daumSpecialScore'][i]<7):
        score_dummy['daumSpecialScore'][i] = 'B'
    else:
        score_dummy['daumSpecialScore'][i] = 'C'
score_dummy
#네이버 전문가 평점 범주화
for i in range(0,816):
    if(score['moviespecialScore'][i]>=7                                   and
score['moviespecialScore'][i]<=10):
        score_dummy['moviespecialScore'][i] = 'A'
    elif(score['moviespecialScore'][i]>=5.5                               and
score['moviespecialScore'][i]<7):
        score_dummy['moviespecialScore'][i] = 'B'
    else:
```

```
            score_dummy['moviespecialScore'][i] = 'C'
score_dummy
score_dummy.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\score_du
mmy.csv',index=False,encoding='CP949')
```

6. data 크롤링 (R)

```
install.packages('MASS')
install.packages('psych')
install.packages('car')
install.packages("Hmisc")
install.packages("dummies")
install.packages('ggplot2')

library(ggplot2)
library(dummies)
library(Hmisc)
library(MASS)
library(psych)
library(car)

data = read.csv('C:/Users/yoonj/Desktop/regression/data/finaldata.csv')

#더미변수화
data_dummy<-dummy.data.frame(data)
str(data_dummy)

#변환을 위해 더미인 변수와 아닌 변수 잠시 분리
tdata_nondummy = data_dummy[,c(1:7)]
pairs.panels(tdata_nondummy)

tdata_dummy= data_dummy[,-c(1:7)]
```

```
par(mfrow=c(2,2))
plot((data_dummy$audiAcc)~(data_dummy$actorscore),
main="rawdata:actorscore")
plot(log(data_dummy$audiAcc)~log(data_dummy$actorscore),
main="transform_data:actorscore")

plot((data_dummy$directorscore),(data_dummy$audiAcc),
main="rawdata:directorscore")
plot(log(data_dummy$directorscore),log(data_dummy$audiAcc),
main="transform_data:directorscore")

plot((data_dummy$firstscrnCnt),(data_dummy$audiAcc),
main="rawdata:firstscrnCnt")
plot(sqrt(data_dummy$firstscrnCnt),log(data_dummy$audiAcc),
main="transform_data:firstscrnCnt")

plot((data_dummy$like),(data_dummy$audiAcc), main="rawdata:movielike")
plot(log(data_dummy$like),log(data_dummy$audiAcc),
main="transform_data:movielike")

plot((data_dummy$news),(data_dummy$audiAcc),
main="rawdata:movienews")
plot(log(data_dummy$news),log(data_dummy$audiAcc),
main="transform_data:movienews")

plot((data_dummy$audiSat),(data_dummy$audiAcc),
main="rawdata:movieaudiSat")
plot(log(data_dummy$audiSat),log(data_dummy$audiAcc),
main="transform_data:movieaudiSat")

plot((data_dummy$distributorscore),(data_dummy$audiAcc),
```

```
main="rawdata:distributorscore")
plot(log(data_dummy$distributorscore),log(data_dummy$audiAcc),
main="transform_data:distributorscore")

#변환
tdata_nondummy$audiAcc = log(data_dummy$audiAcc)
tdata_nondummy$actorscore = log(data_dummy$actorscore)
tdata_nondummy$directorscore = log(data_dummy$directorscore)
tdata_nondummy$like = log(data_dummy$like)
tdata_nondummy$audiSat = log(data_dummy$audiSat)
tdata_nondummy$firstscrnCnt = sqrt(data_dummy$firstscrnCnt)
tdata_nondummy$news = log(data_dummy$news)

pairs.panels(tdata_nondummy)

#변환 후 재 병합
tdata=cbind(tdata_nondummy,tdata_dummy)

fit <- lm(audiAcc~., data = tdata)
summary(fit)

#backward
back<-step(fit, direction = "backward", trace = T)
back$anova

summary(back)
anova(back)

#forward
forw<-step(fit, direction = "forward", trace = 0)
forw$anova
```

```
summary(forw)
anova(forw)

#both
both = stepAIC(fit, direction = "both")
both$anova

summary(both)
anova(both)

######back=both << 채택
backdata<-tdata[,c(1,2,3,5,6,7,11,12,23,24,29,30,36,38,40)]
fit <- lm(audiAcc~., data = backdata)
summary(fit)
vif(fit)

#audisat -
backdata<-backdata[,-c(5)]
outsatfit <- lm(audiAcc~., data = backdata)
summary(outsatfit)
vif(outsatfit)



###################이상치
outlierTest(outsatfit)

#395번 제거
out395<-backdata[-c(395),]
rownames(out395) <-NULL
out395fit<-lm(audiAcc~., data=out395)
summary(out395fit)
```

```
out395back<-step(out395fit, direction = "backward", trace = T)
summary(out395back)
out395back$anova

outlierTest(out395back)
par(mfrow = c(2,2))
plot(out395back)

#25번 제거
out25<-out395[-c(25),]
rownames(out25) <-NULL
out25fit<-lm(audiAcc~., data=out25)
summary(out25fit)

out25back<-step(out25fit, direction = "backward", trace = T)
summary(out25back)
out25back$anova

outlierTest(out25back)
par(mfrow = c(2,2))
plot(out25back)

#287번 제거
out287<-out25[-c(287),]
rownames(out287) <-NULL
out287fit<-lm(audiAcc~., data=out287)
summary(out287fit)

out287back<-step(out287fit, direction = "backward", trace = T)
summary(out287back)
out287back$anova
```

```
outlierTest(out287back)
par(mfrow = c(2,2))
plot(out287back)


#427번 제거
out427<-out287[-c(427),]
rownames(out427) <-NULL
out427fit<-lm(audiAcc~., data=out427)
summary(out427fit)

out427back<-step(out427fit, direction = "backward", trace = T)
summary(out427back)
out427back$anova

outlierTest(out427back)
par(mfrow = c(2,2))
plot(out427back)


#471번 제거
out471<-out427[-c(471),]
rownames(out471) <-NULL
out471fit<-lm(audiAcc~., data=out471)
summary(out471fit)

out471back<-step(out471fit, direction = "backward", trace = T)
summary(out471back)
out471back$anova

outlierTest(out471back)
par(mfrow = c(2,2))
```

```
plot(out471back)

#3번 제거
out3<-out471[-c(3),]
rownames(out3) <-NULL
out3fit<-lm(audiAcc~., data=out3)
summary(out3fit)

out3back<-step(out3fit, direction = "backward", trace = T)
summary(out3back)
out3back$anova

outlierTest(out3back)
par(mfrow = c(2,2))
plot(out3back)

#190번 제거
out190<-out3[-c(190),]
rownames(out190) <-NULL
out190fit<-lm(audiAcc~., data=out190)
summary(out190fit)

out190back<-step(out190fit, direction = "backward", trace = T)
summary(out190back)
out190back$anova

outlierTest(out190back)
par(mfrow = c(2,2))
plot(out190back)

#369번 제거
out369<-out190[-c(369),]
```

```
rownames(out369) <-NULL
out369fit<-lm(audiAcc~., data=out369)
summary(out369fit)

out369back<-step(out369fit, direction = "backward", trace = T)
summary(out369back)
out369back$anova

outlierTest(out369back)
par(mfrow = c(2,2))
plot(out369back)

#349번 제거
out349<-out369[-c(349),]
rownames(out349) <-NULL
out349fit<-lm(audiAcc~., data=out349)
summary(out349fit)

out349back<-step(out349fit, direction = "backward", trace = T)
summary(out349back)
out349back$anova

outlierTest(out349back)
par(mfrow = c(2,2))
plot(out349back)

#16번 제거
out16<-out349[-c(16),]
rownames(out16) <-NULL
out16fit<-lm(audiAcc~., data=out16)
summary(out16fit)
```

```
out16back<-step(out16fit, direction = "backward", trace = T)
summary(out16back)
out16back$anova

outlierTest(out16back)
par(mfrow = c(2,2))
plot(out16back)

#161번 제거
out161<-out16[-c(161),]
rownames(out161) <-NULL
out161fit<-lm(audiAcc~., data=out161)
summary(out161fit)

out161back<-step(out161fit, direction = "backward", trace = T)
summary(out161back)
out161back$anova

outlierTest(out161back)
par(mfrow = c(2,2))
plot(out161back)

#682번 제거
out682<-out161[-c(682),]
rownames(out682) <-NULL
out682fit<-lm(audiAcc~., data=out682)
summary(out682fit)

out682back<-step(out682fit, direction = "backward", trace = T)
summary(out682back)
out682back$anova
```

```
outlierTest(out682back)
par(mfrow = c(2,2))
plot(out682back)

#########################이상치 제거 끝
##최종 모델
lastfit<-step(out682fit, direction = "backward", trace = T)
summary(lastfit)

##잔차분석
install.packages('nortest')
library(nortest)

par(mfrow=c(2,2))
plot(lastfit)

pred<-predict(lastfit)
resid<-residuals(lastfit)

#등분산성
plot(pred,resid)
abline(c(0,0),col="red") #0을 기준으로 무작위 분포

#선형성
crPlots(lastfit)

#정규성
par(mfrow=c(1,2))
qqnorm((resid))
qqline((resid))
```

```
##################################
back_test=lm(audiAcc~actorscore+directorscore+news+firstscrnCnt+naverScoreA+naverScoreB, data=out682) #변수 그래프 그리기 위함. (범주형->수치형)



relweights<-function(fit){
  R <- cor(fit$model)
  nvar <- ncol(R)
  rxx <- R[2:nvar, 2:nvar]
  rxy <- R[2:nvar, 1]
  svd <- eigen(rxx)
  evec <- svd$vectors
  ev <- svd$values
  delta <- diag(sqrt(ev))
  lambda <- evec %*% delta %*% t(evec)
  lambdasq <- lambda^2
  beta <- solve(lambda) %*% rxy
  rsquare <- colSums(beta^2)
  rawwgt <- lambdasq %*% beta ^2
  import <- (rawwgt/rsquare)*100
  import <- as.data.frame(import)
  row.names(import) <- names(fit$model[2:nvar])
  names(import) <- "Weights"
  import <- import[order(import),1,drop=FALSE]
  dotchart(import$Weights,labels=row.names(import),
           xlab="% of R-Square",pch=19,
           main="Relative Importance of Predictor Variables",
           sub=paste("Total R-Square=",round(rsquare,digits=3)))
  return(import)
}
```

```
result=relweights(back_test)

#변수의 상대적 중요성 plot_r^2 : 0.895
plotRelWeights=function(fit){
  data<-relweights(fit)
  data$Predictors<-rownames(data)

p<-ggplot(data=data,aes(x=reorder(Predictors,Weights),y=Weights,fill=Predictors))+
    geom_bar(stat="identity",width=0.5)+
    ggtitle("Relative Importance of Predictor Variables")+
    ylab(paste0("% of R-square \n(Total R-Square=",attr(data,"R-square"),")"))+

geom_text(aes(y=Weights-0.1,label=paste(round(Weights,1),"%")),hjust=1)+
    guides(fill=FALSE)+
    coord_flip()
  p
}

plotRelWeights(back_test)
summary(lastfit)


##################예측
predicted2019 =
read.csv('C:/Users/yoonj/Desktop/regression/data/2019_predicted2.csv')

predicted2019<-predicted2019[,-c(9:11)]
trans2019 = predicted2019
```

```
trans2019$audiAcc = log(predicted2019$audiAcc)
trans2019$actorscore = log(predicted2019$actorscore)
trans2019$directorscore = log(predicted2019$directorscore)
trans2019$firstscrnCnt = sqrt(predicted2019$firstscrnCnt)
trans2019$news = log(predicted2019$news)

log_audiAcc = as.data.frame(predict(lastfit, trans2019))

names(log_audiAcc)[1] = c("predicted_audiAcc")



raw_audiAcc = trans2019[,c(1:2)]

result_reg = cbind(raw_audiAcc, log_audiAcc)
```

7. 2019년 예측 데이터 전처리(python)

```
# 2019년 예측 데이터 전처리
 from datetime import datetime
from dateutil.relativedelta import relativedelta
import pandas as pd
import numpy as np
import json
import os
import re
from bs4 import BeautifulSoup
import requests
from pandas import Series, DataFrame
#import datetime
import time
import calendar
score2019=pd.read_csv(r'C:\\Users\\yoonj\Desktop\\regression\\data\\2019_p
redicted.csv', encoding='CP949')
```

```
score2019_dummy=score2019
score2019
#다음 네티즌 평점 범주화
for i in range(0,28):
    if(score2019['daumScore'][i]>=8 and score2019['daumScore'][i]<=10):
        score2019_dummy['daumScore'][i] = 'A'
    elif(score2019['daumScore'][i]>=6 and score2019['daumScore'][i]<8):
        score2019_dummy['daumScore'][i] = 'B'
    else:
        score2019_dummy['daumScore'][i] = 'C'
score2019_dummy
#네이버 네티즌 평점 범주화
for i in range(0,28):
    if(score2019['naverScore'][i]>=8.5 and score2019['naverScore'][i]<=10):
        score2019_dummy['naverScore'][i] = 'A'
    elif(score2019['naverScore'][i]>=6.5 and
score2019['naverScore'][i]<8.5):
        score2019_dummy['naverScore'][i] = 'B'
    else:
        score2019_dummy['naverScore'][i] = 'C'
score2019_dummy
-------------------------
openDt = score2019_dummy['openDt']
type(openDt)
opendt_split = pd.DataFrame(columns=['year','month','day'])
for i in range(0,28):
    year = int(openDt[i].split('-')[0])
    month = int(openDt[i].split('-')[1])
    day = int(openDt[i].split('-')[2])

    opendt_split=opendt_split.append(pd.DataFrame([[year,month,day]],
columns=['year','month','day']), ignore_index=True)
```

```
opendt_split
startdt = []
for i in range(0,28):

my_date=datetime(opendt_split['year'][i],opendt_split['month'][i],opendt_split
['day'][i])
    weeks2 = my_date + relativedelta(days=-14)
    startdt.append(weeks2)

startDt = pd.DataFrame(startdt)
startDt.columns=['startDt']
startDt
news_date = pd.concat([startDt,openDt], axis=1)
news_date
for i in range(0,28):
    news_date['startDt'][i]                                                =
news_date['startDt'][i].strftime('%Y-%m-%d')
type(news_date['startDt'][0])
for i in range(0,28):
    news_date['startDt'][i] = news_date['startDt'][i].replace("-",".")
    news_date['openDt'][i] = news_date['openDt'][i].replace("-",".")
news_date
--------------------
Movies=[i for i in score2019['movieNm']]
MovieName = DataFrame(Movies)
MovieName
#검색리스트와 기간을 url에 넣을 수 있는 함수
def URLmaker(query,sdate,edate):
    try:
        base_url                                                           =
'https://search.naver.com/search.naver?where=news&query={}&sm=tab_opt
&sort=0&photo=0&field=0&reporter_article=&pd=3&ds={}&de={}'
```

```
            encoded_query=requests.utils.quote(query, encoding='MS949')
            base_url = base_url.format(encoded_query, sdate, edate)
        except:
            print(i)
            pass
        return base_url
#앞에 만든 함수에 검색과 기간을 넣어서 최종 url 생성
FinalUrl=[]
for i in range(len(MovieName)):
        url=URLmaker(MovieName[0][i], news_date['startDt'][i],
news_date['openDt'][i])
        FinalUrl.append(url)
FinalUrl
num=0
idx=0
def get_tag(i):
            rep = requests.get(i)
            source = rep.text
            soup = BeautifulSoup(source, 'html.parser')
            #print(soup.select("#main_pack > div > div.section_head >
div.title_desc.all_my > span"))
            tag = str(soup.select("#main_pack > div > div.section_head >
div.title_desc.all_my > span"))
            tag_split = tag[14:]
            tag_split = tag_split[0:-9]
            #print(tag_split)
            return tag_split

for i in FinalUrl:
        score2019_dummy['news'][idx]=get_tag(i)
        idx+=1
```

```
score2019_dummy
for i in range(0,28):
    score2019_dummy['news'][i] = score2019_dummy['news'][i].replace(',',
'')
score2019_dummy
score2019_dummy.to_csv('C:\\Users\\yoonj\Desktop\\regression\\data\\2019
_predicted.csv',index=False,encoding='CP949')
```