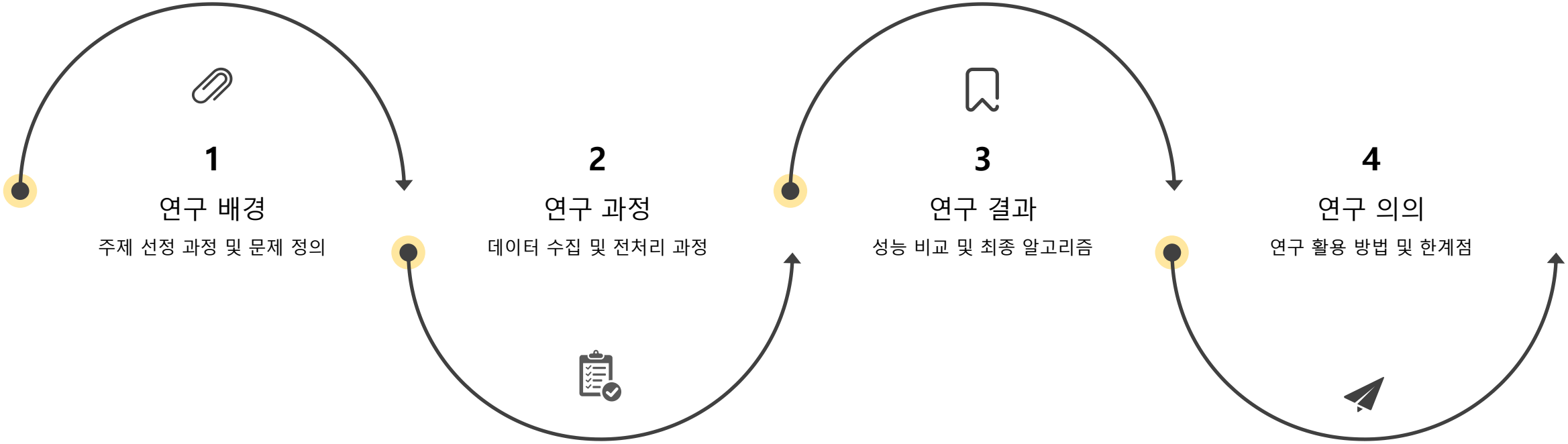
영화 포스터를 활용한 흥행 예측

클라우드 플랫폼 2022/06/20 이지현, 박초은, 김단은

01

CONTENTS



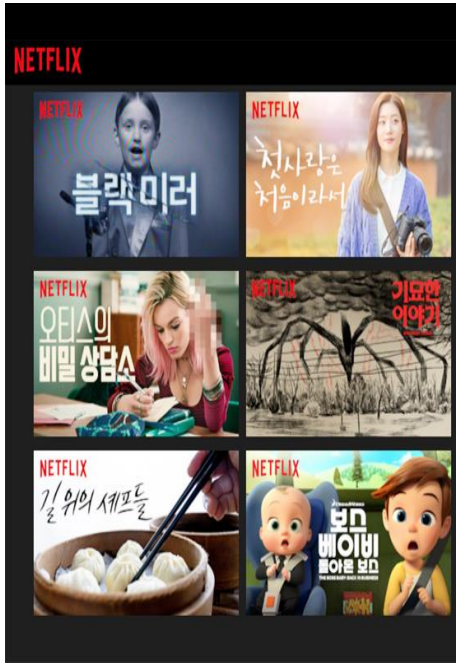
1
연구 배경
주제 선정 과정 및 문제 정의

2
연구 과정
데이터 수집 및 전처리 과정

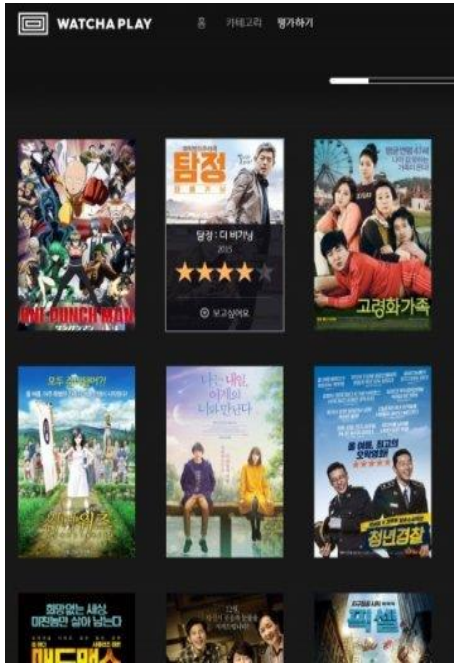
3
연구 결과
성능 비교 및 최종 알고리즘

4
연구 의의
연구 활용 방법 및 한계점

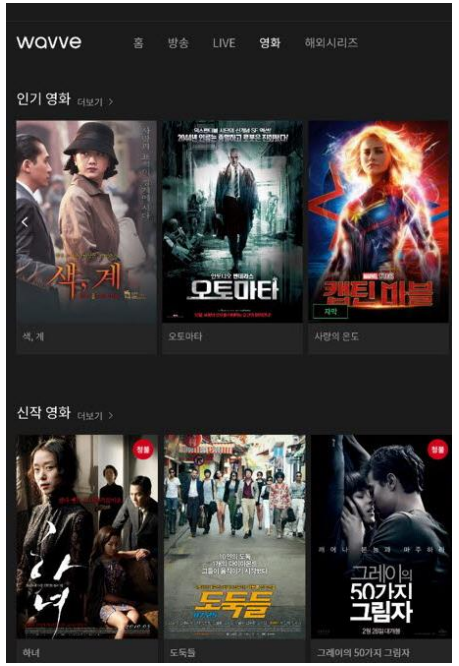
01 연구 배경



NETFLIX



WATCHA

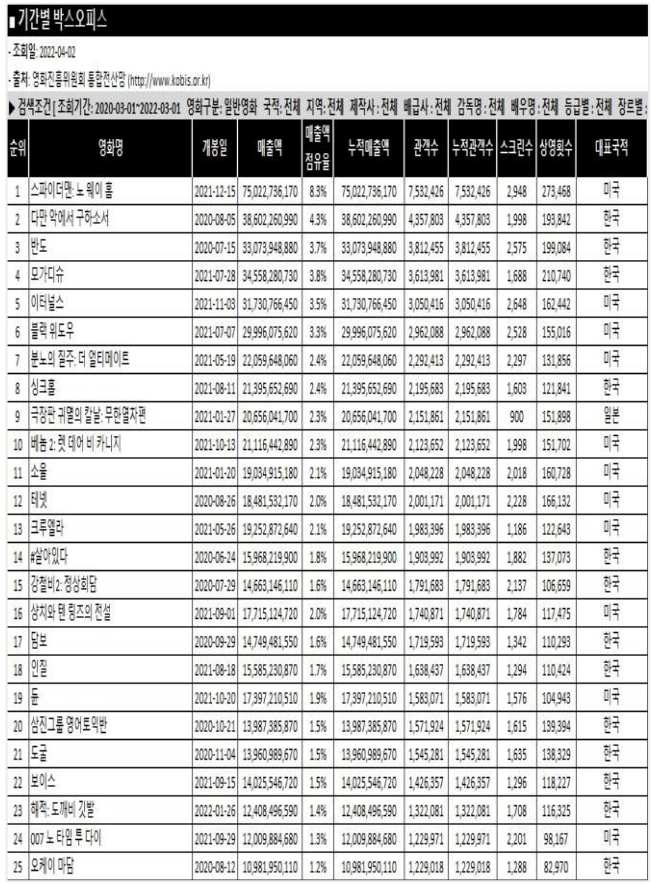


wavve

OTT 플랫폼은 시청 이력과 좋아요를 기반으로 사용자의 성향에 맞는 포스터를 노출시켜 선택 확률을 높임

02

연구 과정



기간별 박스오피스

- 조회일: 2022-04-02
- 출처: 영화진흥위원회 통합전산망 (http://www.kobis.or.kr)

검색조건 [조회기간: 2020-03-01~2022-03-01 영화구분: 일반영화 국적: 전체 지역: 전체 제작사: 전체 배급사: 전체 감독명: 전체 배우명: 전체 등급별: 전체 장르별:

순위	영화명	개봉일	매출액	매출액 점유율	누적매출액	관객수	누적관객수	스크린수	상영회수	대표국적
1	스파이더맨: 노 웨이 홈	2021-12-15	75,022,736,170	8.3%	75,022,736,170	7,532,426	7,532,426	2,948	273,468	미국
2	마지막 악에서 구하소서	2020-08-05	38,602,260,990	4.3%	38,602,260,990	4,357,803	4,357,803	1,998	193,842	한국
3	반도	2020-07-15	33,073,948,880	3.7%	33,073,948,880	3,812,455	3,812,455	2,575	199,084	한국
4	모가디슈	2021-07-28	34,558,280,730	3.8%	34,558,280,730	3,613,981	3,613,981	1,688	210,740	한국
5	이터널스	2021-11-03	31,730,766,450	3.5%	31,730,766,450	3,050,416	3,050,416	2,648	162,442	미국
6	블랙 위도우	2021-07-07	29,996,075,620	3.3%	29,996,075,620	2,962,088	2,962,088	2,528	155,016	미국
7	분노의 질주: 더 얼티메이트	2021-05-19	22,059,648,060	2.4%	22,059,648,060	2,292,413	2,292,413	2,297	131,856	미국
8	싱크홀	2021-08-11	21,395,652,690	2.4%	21,395,652,690	2,195,683	2,195,683	1,603	121,841	한국
9	극장판 귀멸의 칼날: 무한열차편	2021-01-27	20,656,041,700	2.3%	20,656,041,700	2,151,861	2,151,861	900	151,898	일본
10	베놈 2: 렛 데어 비 카니지	2021-10-13	21,116,442,890	2.3%	21,116,442,890	2,123,652	2,123,652	1,998	151,702	미국
11	소울	2021-01-20	19,034,915,180	2.1%	19,034,915,180	2,048,228	2,048,228	2,018	160,728	미국
12	테넷	2020-08-26	18,481,532,170	2.0%	18,481,532,170	2,001,171	2,001,171	2,228	166,132	미국
13	크루엘라	2021-05-26	19,252,872,640	2.1%	19,252,872,640	1,983,396	1,983,396	1,186	122,643	미국
14	#살아있다	2020-06-24	15,968,219,900	1.8%	15,968,219,900	1,903,992	1,903,992	1,882	137,073	한국
15	강철비 2: 정상회담	2020-07-29	14,663,146,110	1.6%	14,663,146,110	1,791,683	1,791,683	2,137	106,659	한국
16	샹치와 텐 링즈의 전설	2021-09-01	17,715,124,720	2.0%	17,715,124,720	1,740,871	1,740,871	1,784	117,475	미국
17	담보	2020-09-29	14,749,481,550	1.6%	14,749,481,550	1,719,593	1,719,593	1,342	110,293	한국
18	인질	2021-08-18	15,585,230,870	1.7%	15,585,230,870	1,638,437	1,638,437	1,294	110,424	한국
19	돈	2021-10-20	17,397,210,510	1.9%	17,397,210,510	1,583,071	1,583,071	1,576	104,943	미국
20	삼진그룹 영아토익반	2020-10-21	13,987,385,870	1.5%	13,987,385,870	1,571,924	1,571,924	1,615	139,394	한국
21	도굴	2020-11-04	13,960,989,670	1.5%	13,960,989,670	1,545,281	1,545,281	1,635	138,329	한국
22	보이스	2021-09-15	14,025,546,720	1.5%	14,025,546,720	1,426,357	1,426,357	1,296	118,227	한국
23	해적: 도깨비 깃발	2022-01-26	12,408,496,590	1.4%	12,408,496,590	1,322,081	1,322,081	1,708	116,325	한국
24	007 노 타임 투 다이	2021-09-29	12,009,884,680	1.3%	12,009,884,680	1,229,971	1,229,971	2,201	98,167	미국
25	오케이 마담	2020-08-12	10,981,950,110	1.2%	10,981,950,110	1,229,018	1,229,018	1,288	82,970	한국

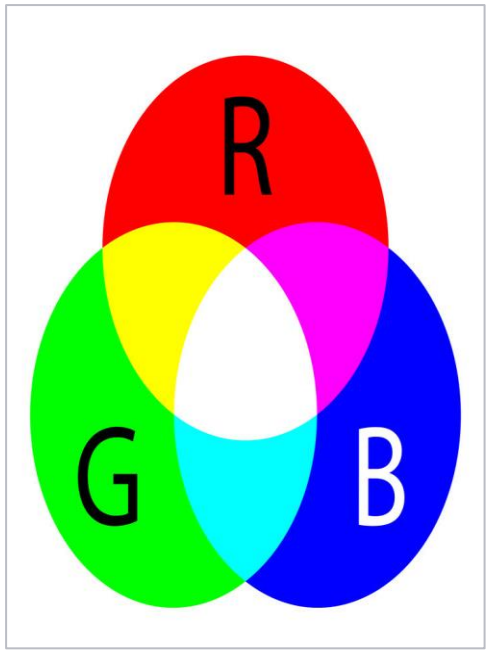
출처: KOBIS

- 개봉일자: 2020.1.1~2022.3.1 국내개봉 기준 (코로나 상황 반영을 위함)
- 영화 구분: 예술영화와 독립영화를 제외한 일반영화 기준
- 특수한 경우(에로 영화)를 제외하여 총 747개의 포스터 이미지 수집
- 747개의 포스터 중 포스터가 추출되지 않는 영화들은 삭제하여 총 650개의 영화 포스터 활용
- Selenium 라이브러리를 이용하여 크롤링
- 중간 발표회에서는 박스오피스 자료를 바탕으로 매출액 점유율에 따른 영화 순위로 흥행 여부 판단

02

연구 과정

포스터의 얼굴 수



포스터의 색감

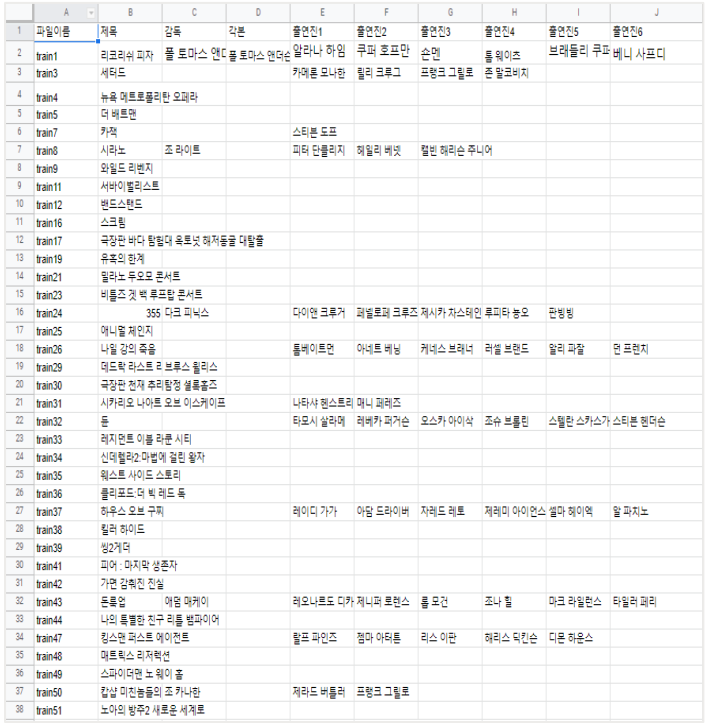
포스터의 글자

- openCV의 Haar Cascades는 측면 인식이 안되어 탈락
- MTCNN의 경우 눈,코,입의 위치를 표시해 정확도를 높일 수 있지만, confidence로 얼굴 인식의 신뢰도를 알 수 있다

counts	R	G	B
0.222222	0.409836	0.290749	0.234783
0.055556	0.000000	0.000000	0.000000
0.055556	0.389344	0.290749	0.230435
0.000000	0.471311	0.577093	0.586957
0.018519	0.459016	0.189427	0.073913
...	...	...	...
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000

- PIL 라이브러리 이용
- 포스터의 각 픽셀의 색상을 RGB로 수치화한 후 평균값을 적용

연구 과정



	A	B	C	D	E	F	G	H	I	J
1	파일이름	제목	감독	각본	출연진1	출연진2	출연진3	출연진4	출연진5	출연진6
2	train1	리코리쉬 피자	폴 토마스 앤더슨	폴 토마스 앤더슨	알라나 하임	쿠퍼 호프만	숀 펜	톰 웨이츠	브래들리 쿠퍼	베니 사프디
3	train3	세터드			카메론 모나한	릴리 크루그	프랭크 그릴로	존 말코비치		
4	train4	뉴욕 메트로폴리탄 오페라								
5	train5	더 배트맨								
6	train7	카잭			스티븐 도프					
7	train8	시라노	조 라이트		피터 딘클리지	헤일리 베넷	켈빈 해리슨 주니어			
8	train9	와일드 리빈지								
9	train11	서바이벌리스트								
10	train12	밴드스탠드								
11	train16	스크림								
12	train17	극장판 바다 탐험대 옥토넛 해저동굴 대탈출								
13	train19	유혹의 한계								
14	train21	릴라노 두오모 콘서트								
15	train23	비틀즈 겟 백 루프탑 콘서트								
16	train24	355 다크 피닉스			다이앤 크루거	페넬로페 크루즈	제시카 자스테인 루피타 뇽오		판빙빙	
17	train25	애니멀 제인지								
18	train26	나일 강의 죽음			톰베이트먼	아네트 베닝	케네스 브래너	러셀 브랜드	엘리 과잘	던 프렌치
19	train29	데드락 라스트 리 브루스 윌리스								
20	train30	극장판 전자 주리탐정 셜록홈즈								
21	train31	시카리오 나아트 오브 이스케이프			나타샤 멘스트리	매니 패레즈				
22	train32	듄			타모시 살라메	레베카 퍼거슨	오스카 아이삭	조슈 브롤린	스텔란 스카스가	스티븐 헨더슨
23	train33	레지던트 이블 라푼 시티								
24	train34	신데렐라2 마법에 걸린 왕자								
25	train35	웨스트 사이드 스토리								
26	train36	클리포드 더 빅 레드 독								
27	train37	하우스 오브 구찌			레이디 가가	아담 드라이버	자레드 레토	제레미 아이언스	셀마 헤이엑	알 파치노
28	train38	컬러 하이드								
29	train39	씽2게더								
30	train41	피어 : 마지막 생존자								
31	train42	가면 갑춰진 진실								
32	train43	돈룩업	애덤 매케이		레오나르도 디카	제니퍼 로렌스	롭 모건	조나 힐	마크 라일런스	타일러 페리
33	train44	나의 특별한 친구 리틀 뱀파이어								
34	train47	킹스맨 퍼스트 에이전트			랄프 파인즈	정마 아터튼	리스 이판	해리스 딕킨슨	디몬 하운소	
35	train48	매트릭스 리저렉션								
36	train49	스파이더맨 노 웨이 홈								
37	train50	갑삽 미친놈들의 조 카나한			제라드 버틀러	프랭크 그릴로				
38	train51	노아의 방주2 새로운 세계로								

→

	R	G	B	faces	text	레이블
0	100.0	66.0	54.0	12.0	1.0	1.0
1	95.0	66.0	53.0	3.0	0.0	0.0
2	115.0	131.0	135.0	3.0	0.0	0.0
3	112.0	43.0	17.0	0.0	0.0	1.0
4	78.0	73.0	78.0	1.0	0.0	0.0
...	...	...	...	...	...	...
606	47.0	64.0	73.0	0.0	1.0	1.0
607	124.0	117.0	110.0	8.0	0.0	1.0
609	42.0	35.0	50.0	0.0	0.0	1.0
610	170.0	175.0	169.0	4.0	0.0	0.0
611	123.0	120.0	120.0	1.0	0.0	1.0

573 rows × 6 columns

- 수상/유명감독/박스 오피스 기록 표시를 text로 결합 후 하나라도 있다면 1 부여
- Faces는 얼굴 수를 나타냄

연구 과정

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.6383	0.6730	0.6173	0.6170	0.6136	0.2748	0.2778	0.173
nb	Naive Bayes	0.6211	0.6596	0.4746	0.6415	0.5326	0.2301	0.2435	0.020
qda	Quadratic Discriminant Analysis	0.6186	0.6358	0.5111	0.6115	0.5495	0.2270	0.2325	0.017
ridge	Ridge Classifier	0.6160	0.0000	0.4798	0.6256	0.5341	0.2204	0.2303	0.020
lr	Logistic Regression	0.6135	0.6294	0.4851	0.6175	0.5356	0.2160	0.2242	0.460
lda	Linear Discriminant Analysis	0.6135	0.6301	0.4690	0.6227	0.5260	0.2142	0.2242	0.021
et	Extra Trees Classifier	0.6134	0.6611	0.5371	0.6090	0.5629	0.2204	0.2280	0.530
ada	Ada Boost Classifier	0.6110	0.6461	0.5421	0.6000	0.5640	0.2159	0.2201	0.140
rf	Random Forest Classifier	0.6036	0.6473	0.5383	0.5858	0.5566	0.2014	0.2044	0.519
dt	Decision Tree Classifier	0.5612	0.5620	0.5164	0.5425	0.5224	0.1189	0.1226	0.030
knn	K Neighbors Classifier	0.5606	0.5934	0.4637	0.5412	0.4965	0.1120	0.1142	0.123
lightgbm	Light Gradient Boosting Machine	0.5586	0.6019	0.4901	0.5408	0.5055	0.1111	0.1152	0.078
dummy	Dummy Classifier	0.5312	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.014
svm	SVM - Linear Kernel	0.4835	0.0000	0.7784	0.4344	0.5343	0.0033	0.0063	0.022

- Pycaret 라이브러리를 활용하여 여러 모델의 성능평가를 단편적으로 확인
- 성능지표가 가장 우수한 Gradient Boosting Classifier를 이용

```
[130] from sklearn.ensemble import GradientBoostingClassifier

      gbc = GradientBoostingClassifier(random_state=0, max_depth=1)
      gbc.fit(X_train, y_train)

      score_train_pre = gbc.score(X_train, y_train) # train set 정확도
      print('{:.3f}'.format(score_train_pre))

      score_test_pre = gbc.score(X_test, y_test) # 일반화 정확도
      print('{:.3f}'.format(score_test_pre))

      0.646
      0.616
```

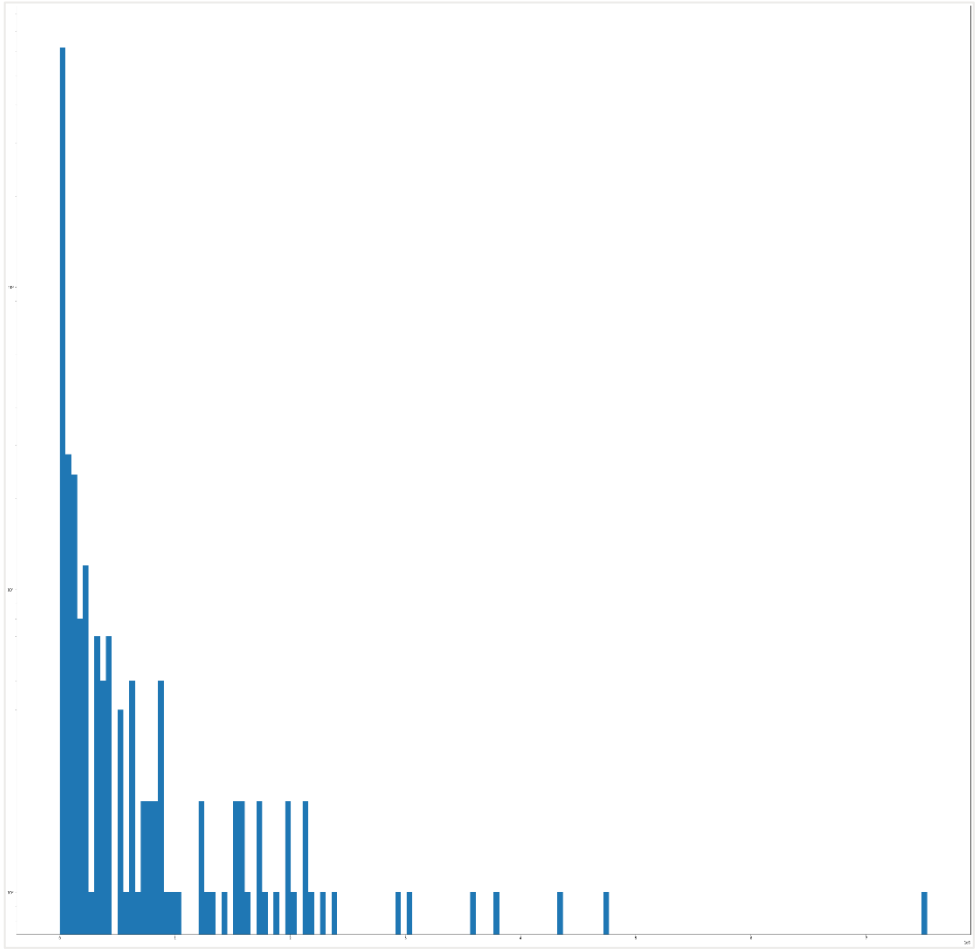
- Train set의 정확도: 65%
- Test set의 정확도: 62% 로 낮은 정확도를 보임

02

연구 과정

중간 연구 과정		최종 연구 과정
영화 흥행 여부를 50:50으로 분류	Step. 1	영화 흥행 여부를 0~4로 분류
포스터 속 홍보 문구를 text로 합쳐 하나라도 있다면 1 부여	Step. 2	Label, director, festival, award, staff로 세분화
정형 데이터만 이용 후 머신 러닝 학습	Step. 3	정형 데이터 및 이미지를 활용해 멀티모달 딥러닝으로 학습

연구 과정



전체 데이터셋 분포도

- 50:50으로 나눠 변별력이 없다는 피드백
- 논문 작성자에게 메일로 문의했지만 답변 X
- 흥행 여부를 0~4로 다중분류
- 클러스터링 기술을 활용하여 불균형한 레이블 생성
- 흥행 여부 판단에 있어 누적 관객수를 활용
- 관객수가 높은 흥행 영화는 비 흥행 영화에 비해 상대적으로 수가 적을 수 밖에 없어 불균형 데이터 일 수 밖에 없다

02 연구 과정

	R	G	B	faces	text	레이블
0	100.0	66.0	54.0	12.0	1.0	1.0
1	95.0	66.0	53.0	3.0	0.0	0.0
2	115.0	131.0	135.0	3.0	0.0	0.0
3	112.0	43.0	17.0	0.0	0.0	1.0
4	78.0	73.0	78.0	1.0	0.0	0.0
...	...	...	...	...	...	...
606	47.0	64.0	73.0	0.0	1.0	1.0
607	124.0	117.0	110.0	8.0	0.0	1.0
609	42.0	35.0	50.0	0.0	0.0	1.0
610	170.0	175.0	169.0	4.0	0.0	0.0
611	123.0	120.0	120.0	1.0	0.0	1.0

573 rows × 6 columns

	인덱스	레이블	counts	R	G	B	lable	director	festival	award	staff
0	1	3.0	12.0	100.0	66.0	54.0	1.0	0.0	1.0	1.0	0.0
1	2	3.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	3	3.0	3.0	95.0	66.0	53.0	0.0	0.0	0.0	0.0	0.0
3	4	3.0	0.0	115.0	131.0	135.0	0.0	0.0	0.0	0.0	0.0
4	5	1.0	1.0	112.0	43.0	17.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
767	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
768	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
769	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
770	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
771	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

772 rows × 11 columns

counts	R	G	B
0.222222	0.409836	0.290749	0.234783
0.055556	0.000000	0.000000	0.000000
0.055556	0.389344	0.290749	0.230435
0.000000	0.471311	0.577093	0.586957
0.018519	0.459016	0.189427	0.073913
...	...	...	...
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000

- 중간 연구 과정에서는 text로 모두 결합했지만 성능이 낮아 label, director, festival, award, staff로 세분화
- Counts = 얼굴 수
- RGB = 색감
- label/director/festival/award/staff = 홍보 문구 여부
- 연속적인 숫자 값은 0과 1 사이의 값으로 변환하여 모델 성능 향상

02 연구 과정

Random Forest

```
from keras.utils.np_utils import to_categorical

final = pd.concat([label, faces], axis=1)
final = pd.merge(final, color, how='outer', on='인덱스')
final = pd.merge(final, text, how='outer', left_on='인덱스', right_on='index')

final = final[['인덱스', '레이블', 'counts', 'R', 'G', 'B', 'lable', 'director', 'festival', 'award', 'staff']]
final = final.dropna()
final = final.reset_index(drop=True)

X = final[['counts', 'R', 'G', 'B', 'lable', 'director', 'festival', 'award', 'staff']]
y = final['레이블']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

forest = RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, y_train)
print("Accuarcy on training set: {:.3f}".format(forest.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(forest.score(X_test, y_test)))
```

```
Accuarcy on training set: 1.000
Accuracy on test set: 0.778
```

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier()
model.fit(X_train, y_train)

print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

```
0.9794344473007712
0.7544910179640718
```

- 7:3으로 분리 후 예측 진행
- Random Forest 정확도: 0.778
- Gradient Boosting Classifier 정확도: 0.754

VGG-16

```
checkpoint = ModelCheckpoint(filepath=model_path , monitor='val_loss', verbose=1, save_best_only=True)
early_stopping = EarlyStopping(monitor='val_loss', patience=6)

model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_22 (InputLayer)       [(None, 64, 64, 3)]       0

 block1_conv1 (Conv2D)       (None, 64, 64, 64)        1792

 block1_conv2 (Conv2D)       (None, 64, 64, 64)        36928

 block1_pool (MaxPooling2D)  (None, 32, 32, 64)        0

 block2_conv1 (Conv2D)       (None, 32, 32, 128)       73856

 block2_conv2 (Conv2D)       (None, 32, 32, 128)       147584

 block2_pool (MaxPooling2D)  (None, 16, 16, 128)       0

 block3_conv1 (Conv2D)       (None, 16, 16, 256)       295168

 block3_conv2 (Conv2D)       (None, 16, 16, 256)       590080

 block3_conv3 (Conv2D)       (None, 16, 16, 256)       590080

 block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0

 block4_conv1 (Conv2D)       (None, 8, 8, 512)         1180160

 block4_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_32 (Conv2D)          (None, 64, 64, 32)        896

 max_pooling2d_32 (MaxPoolin  (None, 32, 32, 32)       0
 g2D)
```

```
print("정확도 : %.4f" % (model.evaluate(X_test, y_test)[1]))
```

```
4/4 [==============================] - 0s 44ms/step - loss: 0.7561 - accuracy: 0.7714
정확도 : 0.7714
```

Pure CNN

```
if not os.path.exists(model_dir):
  os.mkdir(model_dir)

model_path = model_dir + 'multi_img_classification.model'
checkpoint = ModelCheckpoint(filepath=model_path , monitor='val_loss', verbose=1, save_best_only=True)
early_stopping = EarlyStopping(monitor='val_loss', patience=6)

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 64, 64, 32)        896

 max_pooling2d_6 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 dropout_9 (Dropout)         (None, 32, 32, 32)        0

 conv2d_7 (Conv2D)           (None, 32, 32, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 dropout_10 (Dropout)        (None, 16, 16, 64)        0

 flatten_3 (Flatten)         (None, 16384)             0

 dense_6 (Dense)             (None, 64)                1048640

 dropout_11 (Dropout)        (None, 64)                0

 dense_7 (Dense)             (None, 4)                 260

=================================================================
Total params: 1,068,292
Trainable params: 1,068,292
Non-trainable params: 0
_________________________________________________________________
```

```
print("정확도 : %.4f" % (model.evaluate(X_test, y_test)[1]))
```

```
4/4 [==============================] - 0s 37ms/step - loss: 0.9829 - accuracy: 0.7333
정확도 : 0.7333
```

- 총 이미지 수:85
- Augmentation 후: 420
- VGG-16 정확도: 0.771
- Train set 크기: 315
- Test set 크기: 105
- Pure CNN 정확도: 0.733

연구 과정

```
from sklearn.preprocessing import OneHotEncoder

label = pd.read_csv('/content/drive/MyDrive/클라우드  플랫폼 /데이터/기간별_박스오피스_전처리_3.csv')
color = pd.read_csv('/content/drive/MyDrive/클라우드  플랫폼 /데이터/색감 데이터.csv')
faces = pd.read_csv('/content/drive/MyDrive/클라우드  플랫폼 /데이터/얼굴인식.csv')
text = pd.read_csv('/content/drive/MyDrive/클라우드  플랫폼 /데이터/text_data.csv')

final = pd.concat([label, faces], axis=1)
final = pd.merge(final, color, how='outer', on='인덱스')
final = pd.merge(final, text, how='outer', left_on='인덱스', right_on='index')

final = final[['인덱스', '레이블', 'counts', 'R', 'G', 'B', 'lable', 'director', 'festival', 'award', 'staff']]
final = final.fillna(0)
final = final.reset_index(drop=True)

final
```

```
continuous = ['counts', 'R', 'G', 'B']
cs = MinMaxScaler()
Continuous = cs.fit_transform(final[continuous])
Continuous = pd.DataFrame(Continuous, columns=['counts', 'R', 'G', 'B'])

new = final[['인덱스', '레이블']].join(Continuous)
new = new.join(final[['lable', 'director', 'festival', 'award', 'staff']])
new
```

	인덱스	레이블	counts	R	G	B	lable	director	festival	award	staff
0	1	3.0	0.222222	0.409836	0.290749	0.234783	1.0	0.0	1.0	1.0	0.0
1	2	3.0	0.055556	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
2	3	3.0	0.055556	0.389344	0.290749	0.230435	0.0	0.0	0.0	0.0	0.0
3	4	3.0	0.000000	0.471311	0.577093	0.586957	0.0	0.0	0.0	0.0	0.0
4	5	1.0	0.018519	0.459016	0.189427	0.073913	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
767	0	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
768	0	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0

- 모든 정형데이터 파일 불러온 뒤 하나로 합침
- 빈 값은 0으로 대체
- 값의 범위가 불균형해 MinMaxScaler로 정규화

```
cnn = vgg16.VGG16(input_shape=(80, 60, 3), include_top= False, weights='imagenet')

global_average_layer = GlobalAveragePooling2D()
prediction_layer = Dense(4, activation ='softmax')

model = Sequential([cnn, global_average_layer, prediction_layer])
```

```
input1 = Input(shape=(9,))
x = Dense(8, activation='relu')(input1)
output1 = Dense(4, activation='relu')(x)

combinedInput = Concatenate()([output1, model.output])
x = Dense(4, activation="relu")(combinedInput)
output = Dense(4, activation="softmax")(x)
model_new = Model(inputs=[input1, model.input], outputs=output)
```

- 이미지 데이터 처리를 위해, 기존의 VGG16모델의 구조를 변형하여 활용
- 정형데이터 처리를 위한 DNN 모델을 간단하게 구현한 뒤 멀티모달 딥러닝을 활용하여 변형한 VGG16모델과 DNN 모델을 혼합

연구 과정

```
] #Train 이미지 데이터와 정형데이터 인덱스 맞추기

  train_data = []

  for j in range(4):
    train = os.listdir('/content/drive/MyDrive/클라우드  플랫폼 /데이터/포스터 데이터/포스터/포스터/train/'+str(j))
    for i in train:
      i = i.split('n')[1]
      i = i.split('.')[0]
      train_data.append(int(i))

  trainAttr = pd.DataFrame([])

  for k in train_data:
    trainAttr = pd.concat([trainAttr, new[new['인덱스']==k]])

  trainAttr = trainAttr[trainAttr['인덱스']<=733]
  trainAttr = trainAttr.sort_values(by=['인덱스'])

  trainAttrX = trainAttr[['counts', 'R', 'G', 'B', 'lable', 'director', 'festival', 'award', 'staff']]
  trainAttrY = trainAttr[['레이블']]
```

```
] #Test 이미지 데이터와 정형데이터 인덱스 맞추기

  test_data = []

  for j in range(4):
    test = os.listdir('/content/drive/MyDrive/클라우드  플랫폼 /데이터/포스터 데이터/포스터/포스터/test/'+str(j))
    for i in test:
      i = i.split('n')[1]
      i = i.split('.')[0]
      test_data.append(int(i))

  testAttr = pd.DataFrame([])

  for k in test_data:
    testAttr = pd.concat([testAttr, new[new['인덱스']==k]])

  testAttr = testAttr[testAttr['인덱스']<=727]
  testAttr = testAttr.sort_values(by=['인덱스'])

  testAttrX = testAttr[['counts', 'R', 'G', 'B', 'lable', 'director', 'festival', 'award', 'staff']]
  testAttrY = testAttr[['레이블']]
```

```
#one-hot encoding

from tensorflow.keras.utils import to_categorical
trainAttrY = to_categorical(trainAttrY)
testAttrY = to_categorical(testAttrY)
```

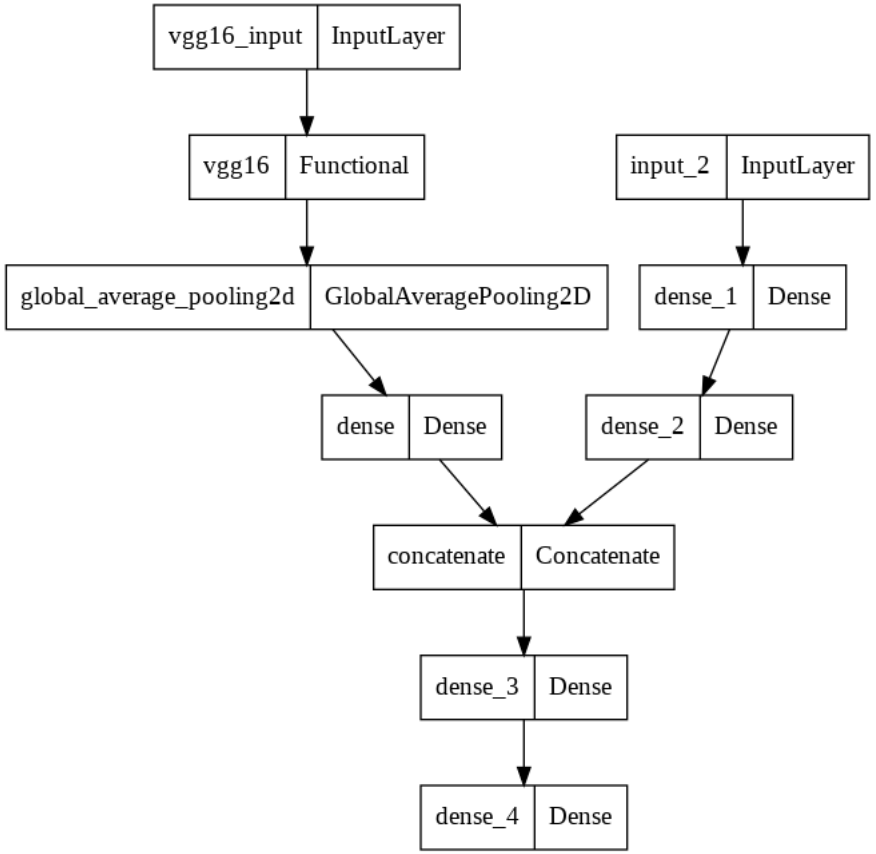
-->

```
0] #train 이미지 데이터셋 불러오기

  train_images = []

  for i in sorted(list(trainAttr['인덱스'])):
    image = cv2.imread('/content/drive/MyDrive/클라우드  플랫폼 /데이터/포스터 데이터/포스터/포스터/train/all/train'+str(i)+'.jpg의 사본')
    if image is None:
      pass
    else:
      image = cv2.resize(image, (60, 80))
      train_images.append(image)

  train_images = np.array(train_images)
```

```
1] #test 이미지 데이터셋 불러오기

  test_images = []

  for i in sorted(list(testAttr['인덱스'])):
    image = cv2.imread('/content/drive/MyDrive/클라우드  플랫폼 /데이터/포스터 데이터/포스터/포스터/test/all/train'+str(i)+'.jpg의 사본')
    if image is None:
      pass
    else:
      image = cv2.resize(image, (60, 80))
      test_images.append(image)

  test_images = np.array(test_images)
```

구글 드라이브에 저장된 이미지와 정형 데이터의 인덱스 순서가
맞지 않아 전처리를 통해 이를 맞춤

연구 과정


vgg16_input
InputLayer
vgg16
Functional
input_2
InputLayer
global_average_pooling2d
GlobalAveragePooling2D
dense_1
Dense
dense
Dense
dense_2
Dense
concatenate
Concatenate
dense_3
Dense
dense_4
Dense

연구 결과

Method	Accuracy
Pure CNN	0.7333
VGG-16	0.7714
Random Forest	0.778
Gradient Boosting Classifier	0.754
Multimodal Deep Learning	0.8173

04 연구 의의

의의

- 영화 한편 당 다수의 포스터가 제작되어 해당 연구를 통해 영화 관계자들은 포스터 제작 비용을 절감할 수 있다.
- 세계 최대 OTT 플랫폼인 넷플릭스의 경우 회원들의 시청 이력과 좋아요 표시 컨텐츠를 기반으로 성향에 맞는 포스터를 디스플레이 하여 선택 확률을 높여 해당 모델을 더하면 선택 확률이 더욱 높아질 것이라 기대한다.
- 영화 장면을 여러 포스터 중 하나로 제작하는 경우 해당 모델을 이용하면 최적의 포스터를 제작할 수 있을 것.
- 영화 분야 뿐만이 아닌 공연, 연극, 뮤지컬 등 다양한 엔터테인먼트 분야에 활용 가능한 잠재력을 가지고 있다.

한계점

- 데이터 수집기간이 2020년부터 여서 이전의 영화 포스터에 대한 예측은 불가능하다.
  (포스터 디자인은 시대별로 변화)
- 누적 관객 수에 따른 예측을 했지만, 전통적인 영화 흥행 기준은 손익분기점을 기준으로 본다.

참고 논문

- 조유정, 강경표 and 권오병. (2021). 공연예술에서 광고포스터의 이미지 특성을 활용한 딥러닝 기반 관객예측. 한국전자거래학회지, 26(2), 19-43.
- 김규연 and 윤성의. (2021). 이미지를 매개로 하는 멀티모달 반지도학습. 정보과학회 컴퓨팅의 실제 논문지, 27(12), 578-583.
- Y. Matsuzaki et al., "Could you guess an interesting movie from the posters?: An evaluation of vision-based features on movie poster database," 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), 2017, pp. 538-541, doi: 10.23919/MVA.2017.7986919.

감사합니다